DiffCLIP: Leveraging Stable Diffusion for Language Grounded 3D Classification

Sitian Shen¹, Zilin Zhu¹, Linqian Fan², Harry Zhang³, Xinxiao Wu¹
¹Beijing Key Laboratory of Intelligent Information Technology, School of Computer Science and Technology, Beijing Institute of Technology, Beijing, China,
²Beijing Jiaotong University, Beijing, China,
³Massachusetts Institute of Technology, Cambridge, USA,
sitian@bit.edu.cn, smzzlcn@gmail.com, LinqFan@bjtu.edu.cn, harryz@mit.edu, wuxinxiao@bit.edu.cn

Abstract

Large pre-trained models have revolutionized the field of computer vision by facilitating multi-modal learning. Notably, the CLIP model has exhibited remarkable proficiency in tasks such as image classification, object detection, and semantic segmentation. Nevertheless, its efficacy in processing 3D point clouds is restricted by the domain gap between the depth maps derived from 3D projection and the training images of CLIP.

This paper introduces DiffCLIP, a novel pre-training framework that seamlessly integrates stable diffusion with ControlNet. The primary objective of DiffCLIP is to bridge the domain gap inherent in the visual branch. Furthermore, to address few-shot tasks in the textual branch, we incorporate a style-prompt generation module.

Extensive experiments on the ModelNet10, ModelNet40, and ScanObjectNN datasets show that DiffCLIP has strong abilities for 3D understanding. By using stable diffusion and style-prompt generation, DiffCLIP achieves an accuracy of 43.2% for zero-shot classification on OBI_BG of ScanObjectNN, which is state-of-the-art performance, and an accuracy of 82.4% for zero-shot classification on ModelNet10, which is also state-of-the-art performance.

1. Introduction

Significant advancements in Natural Language Processing (NLP), particularly in the context of large pre-trained models, have considerably impacted computer vision, enabling multi-modal learning, transfer learning, and spurring the development of new architectures [6]. These pre-trained NLP models have been instrumental in the extraction of high-level features from text data, subsequently enabling the fusion with visual features from images or videos, which in turn improves performance on tasks such as image captioning and video classification. Recently, OpenAI’s Contrastive Language Image Pre-training (CLIP) [19] model has been particularly popular due to its superior performance in image classification, object detection, and semantic segmentation. These achievements underscore the potential of large vision-language pre-trained models across a spectrum of applications.

The remarkable performance of large pre-trained models on 2D vision tasks has stimulated researchers to investigate the potential applications of these models within the domain of 3D point cloud processing. Several approaches, including PointCLIP [39], PointCLIP v2 [43] and CLIP2Point [11] have been proposed over the years. While these methods demonstrate advantages in zero-shot and few-shot 3D point cloud processing tasks, experimental results on real-world datasets suggest that their efficacy in handling real-world tasks is limited. This limitation is primarily attributed to the significant domain gap between 2D depth maps derived from 3D projection and training images of CLIP. Consequently, the primary focus of our work is to minimize this domain gap and improve the performance of CLIP on zero-shot and few-shot 3D vision tasks.

Previous studies have made significant progress in addressing this challenge. For instance, [10] leverages prompt learning to adapt the input domain of the pre-trained visual encoder from computer-generated images of CAD objects to real-world images. The CLIP² [36] model aligns three-dimensional space with a natural language representation that is applicable in real-world scenarios, facilitating knowledge transfer between domains without prior training. To further boost the performance, we introduce a style feature extraction step and a style transfer module via stable diffusion [22] to further mitigate the domain gap.

In this paper, we propose a novel pre-training framework, called DiffCLIP, to minimize the domain gap within both the visual and textual branches. On the visual side, we design a multi-view projection on the original 3D point cloud, resulting in multi-view depth maps. Each depth map undergoes a style transfer process guided by stable diffusion.
and ControlNet [37], generating photorealistic 2D RGB images. The images generated from the style transfer module are then input into the frozen image encoder of CLIP. For zero-shot tasks, we design handcrafted prompts for both the stable diffusion module and the frozen text encoder of CLIP. For few-shot tasks, we further establish a style-prompt generation module that inputs style features extracted from a pre-trained point-cloud-based network and processed via a meta-net, and outputs the projected features as the style-prompt. Coupled with the class label, the entire prompt is then fed into the frozen text encoder of CLIP to guide the downstream process.

To demonstrate the effectiveness of DiffCLIP, we evaluate its zero-shot and few-shot classification ability on both synthetic and real-world datasets like ModelNet and ScanObjectNN. Our main contributions are summarized as follows:

1. We propose DiffCLIP, a novel neural network architecture that combines a pre-trained CLIP model with point-cloud-based networks for enhanced 3D understanding.

2. We develop a technique that effectively minimizes the domain gap in point-cloud processing, leveraging the capabilities of stable diffusion and style-prompt generation, which provides significant improvements in the task of point cloud understanding.

3. We conduct experiments on widely adopted benchmark datasets such as ModelNet and the more challenging ScanObjectNN to illustrate the robust 3D understanding capability of DiffCLIP and conduct ablation studies that evaluate the significance of each component within DiffCLIP’s architecture.

2. Related Work

2.1. 3D Shape Classification

3D point cloud processing methods for classification can be divided into three categories: projection-based [13; 33], volumetric-based [14; 21] and point-based [8; 40]. In projection-based methods, 3D shapes are often projected into multiple views to obtain comprehensive features. Then they are fused with well-designed weight. The most common instances of such a method are MVCNN [27] which employs a straightforward technique of max-pooling to obtain a global descriptor from multi-view features, and MHBN [34] that makes use of harmonized bilinear pooling to integrate local convolutional features and create a smaller-sized global descriptor. Among all the point-based methods, PointNet [16] is a seminal work that directly takes point clouds as its input and achieves permutation invariance with a symmetric function. Since features are learned independently for each point in PointNet [16], the local structural information between points is hardly captured. Therefore, PointNet++ [17] is proposed to capture fine geometric structures from the neighborhood of each point.

Recently, several 3D large pre-trained models are proposed. CLIP [19] is a cross-modal text-image pre-training model based on contrastive learning. PointCLIP [39] and PointCLIP v2 [43] extend CLIP’s 2D pre-trained knowledge to 3D point cloud understanding. Some other 3D processing methods based on large pre-trained model CLIP are proposed in recent years, including CLIP2Point [11], CLIP2 [36], and CG3D [10]. Similar to some of those methods, our model combines widely used point cloud learning architectures with novel large pre-trained models such as CLIP. The key difference here is that our point-based network hierarchically guides the generation of prompts by using direct 3D point cloud features, which largely reduces the domain gap between training and testing data for better zero-shot and few-shot learning. We also design a simpler yet more effective projection method, an intuitive multi-view fusion strategy, to make the projection processing as efficient as possible for the zero-shot task.

2.2. Domain Adaptation and Generalization in Vision-Language Models

Vision-language pre-trained models such as CLIP exhibit impressive zero-shot generalization ability to the open world. Despite the good zero-shot performance, it is found that further adapting CLIP using task-specific data comes at the expense of out-of-domain (OOD) generalization ability. [20; 31]. Recent advances explore improving the OOD generalization of CLIP on the downstream tasks by adapter learning [7; 38], model ensembling [31], test-time adaptation [25], prompt learning [41; 42], and model fine-tuning [26]. Specifically, in StyLIP [1], style features are extracted hierarchically from the image encoder of CLIP to generate domain-specific prompts. To the best of our knowledge, our work is the first to extract style features of 3D point clouds directly from point-based processing networks. With the extracted style features, we are able to generate domain-specific prompts for the text encoder of CLIP.

2.3. Style Transfer using Diffusion Model

Traditional style transfer is one of the domain generalization methods which helps reduce the domain gap between source and target domains by aligning their styles while preserving the content information [12]. For example, Domain-Specific Mappings [4] has been proposed to transfer the style of source domains to a target domain without losing domain-specific information. Fundamentally different from traditional methods of style transfer that operate on pre-existing images, text-to-image diffusion models generate an image based on a textual description that may include style-related information, thus embedding the style defined in textual descriptions into the generated images. These models are based
on a diffusion process, which involves gradually smoothing out an input image to create a stylized version of it, showing a strong ability in capturing complex texture and style information in images. For instance, Glide [15], Disco Diffusion [5], Stable Diffusion [23], and Imagen [24] all support text-to-image generation and editing via a diffusion process, which can further be used to change the image styles via textual prompts variations. We refer to the textual prompts in diffusion models as style prompts as they often encapsulate the style of generated images. In our work, we implement different text-to-image diffusion models and choose ControlNet [37] to help transfer depth map from 3D projection to a more realistic image style, minimizing its style gap with the training images of CLIP.

3. Method

DiffCLIP is a novel neural network architecture that enhances the performance of the CLIP model on 3D point processing tasks via style transfer and style-prompt generation. In Section 3.1, we first revisit PointCLIP [39] and PointCLIP v2 [43] that first introduce the CLIP model into the processing of 3D point clouds. Then in Section 3.2, we introduce the framework and motivation of each part of DiffCLIP, which minimizes the domain gap between 3D tasks and 2D pre-training data. In Section 3.3, we present the detailed usage of DiffCLIP on zero-shot and few-shot classification tasks.

3.1. Revisiting PointCLIP and PointCLIP v2

PointCLIP [39] is a recent extension of CLIP, which enables zero-shot classification on point cloud data. It involves taking a point cloud and converting it into multiple depth maps from different angles without rendering. The predictions from the individual views are combined to transfer knowledge from 2D to 3D. Moreover, an inter-view adapter is used to improve the extraction of global features and fuse the few-shot knowledge acquired from 3D into CLIP that was pre-trained on 2D image data. Specifically, for an unseen dataset of $K$ classes, PointCLIP constructs the textual prompt by placing all category names into a manual template, termed prompt, and then encodes the prompts into a $D$-dimensional textual feature, acting as the zero-shot classifier $W_t \in \mathbb{R}^{K \times D}$. Meanwhile, the features of each projected image from $M$ views are encoded as $f_i$ for $i = 1, ..., M$ by the visual encoder of CLIP. On top of this, the classification logits $s_i$ of view $i$ and the final logsit $s_p$ of each point are calculated by

$$\text{logits}_i = f_i W^T_t, \quad \forall i = 1, ..., M$$ (1a)

$$\text{logits}_p = \sum_{i=1}^{M} \alpha_i \text{logits}_i,$$ (1b)

where $\alpha_i$ is a hyper-parameter weighing the importance of view $i$.

PointCLIP v2 [43] introduces a realistic shape projection module to generate more realistic depth maps for the visual encoder of CLIP, further narrowing down the domain gap between projected point clouds and natural images. Moreover, a large-scale language model, GPT-3 [2], is leveraged to automatically design a more descriptive 3D-semantic prompt for the textual encoder of CLIP instead of the previous hand-crafted one.

3.2. DiffCLIP Framework

The DiffCLIP framework is illustrated in Fig. 1. We describe each component of DiffCLIP in detail in the following sections.

3.2.1 Multi-View Realistic Projection

In DiffCLIP, we use stable diffusion [22] to help transfer projected depth maps to a more realistic image style, minimizing the domain gap with the training images of CLIP. To generate realistic depth maps for better controlling the stable diffusion model, as well as to save computational resources for zero-shot and few-shot tasks, we design a multi-view realistic projection module, which has three steps: proportional sampling, central projection, and 2D max-pooling densifying.

**Proportional Sampling.** We randomly sample points on each face and edge of the object in the dataset, the number of which is proportional to the area of the faces and the length of the edges. Specifically, assuming that $k_1$ points are sampled on a line with length $l$ and $k_f$ points are sampled on a face with area $s$, we set sampling threshold values $\beta_1$ and $\beta_2$, so the number of sampled points can be calculated by $k_1 = \frac{l}{\beta_1}$ and $k_f = \frac{s}{\beta_2}$.

**Central Projection.** For each projection viewpoint $n \in \{1, 2, ..., N\}$, we select a specific projection center and projection plane, and use affine transformation to calculate the coordinates of the projection point on the projection plane for each sampling point. The pixel intensity $d$ is calculated as $d = \frac{1}{dis}$, where $dis$ is the distance between the sampling point and our projection plane. On the 2D plane, the projected point may not fall exactly on a pixel, so we use the nearest-neighbor pixel to approximate its density.

**2D Max-pooling Densifying.** In the central projection step, the nearest-neighbor pixel approximation may yield many pixels to be unassigned, causing the projected depth map to look unrealistic, sparse, and scattered. To tackle this problem, we densify the projected points via a local max-pooling operation to guarantee visual continuity. For each projected pixel of the depth map, we choose the max density $d_{\text{max}}$ among four points: the pixel itself and the pixels to its right, bottom, and bottom right. We then assign the max density to the neighboring four pixels.
3.2.2 Stable-Diffusion-Based Style Transfer

To better use stable diffusion [22] in our task-specific domain, where the input is depth maps, we use ControlNet [37], a robust neural network training method to avoid overfitting and to preserve generalization ability when large models are trained for specific problems. In DiffCLIP, ControlNet employs a U-Net architecture identical to the one used in stable diffusion. It duplicates the weights of an extensive diffusion model into two versions: one that is trainable and another that remains fixed. The linked trainable and fixed neural network units incorporate a specialized convolution layer known as "zero convolution." In this arrangement, the weights of the convolution evolve from zero to optimized parameters in a learned manner. ControlNet has several implementations with different image-based conditions to control large diffusion models, including Canny edge, Hough line, HED boundary, human pose, depth, etc. We use the frozen pre-trained parameters of ControlNet under depth condition, which is pre-trained on 3M depth-image-caption pairs from the internet. The pre-trained ControlNet is then used to generate our own realistic images of depth maps using different class labels as prompts. We illustrate the style-transfer effects of ControlNet in Fig. 2.

3.2.3 Style-Prompt Generation

Optionally, when we want to do few-shot learning, we set up a style prompt generation module to embed style into prompts as is shown in the left part of Fig. 3. We make use of “style features” in point cloud processing analogous to the usage in 2D vision. In 2D vision tasks, ConvNets act as visual encoders that extract features at dif-
different layers: higher-level layers tend to learn more abstract and global features while lower-level layers detect simple and local features such as edges, corners, and blobs [9; 35]. We conjecture that in the 3D point cloud case, the combination of features extracted from multiple levels could also improve the generalizability of the model, similar to how style features improve learning on RGB images in 2D vision tasks.

DiffCLIP leverages CLIP’s frozen vision encoder $f_v$ and text encoder $f_t$. Additionally, a pre-trained point-based network, Point Transformer $f_p$ [40], is used in DiffCLIP. The Point Transformer operates by first encoding each point in the point cloud into a high-dimensional feature vector using a multi-layer perception network [30], and then processing the encoded feature vectors are then processed by a series of self-attention layers, similar to the ones used in the Transformer [29] architecture.

We pre-train the Point Transformer network using our customized dataset, ShapeNet37, which consists of sampled data from ShapeNetCore [3]. The network comprises several blocks, each generating feature maps at different scales. To incorporate domain-specific characteristics, we project the output features from each block into a token and then concatenate them together with a class label to generate the style prompt as the input of CLIP’s text encoder.

### 3.3. Using DiffCLIP

#### 3.3.1 Zero-shot 3D classification

For each 3D point cloud input $x$, after being projected into $M$ views and densified, we obtain their smooth depth maps $x_i'$, $\forall i \in \{1, ..., M\}$. These depth maps are then fed into ControlNet as conditions to guide the image generation of stable diffusion. We use the prompt “a photo of a [class], best quality, extremely detailed” as the default style prompt input to the stable diffusion model, denoted as $t_j$, where $j \in \{1, ..., K\}$. The generated realistic images are represented as $x_{ij}' = d(x_i', t_j)$, where the function $d(\cdot)$ denotes the stable diffusion combined with ControlNet. Based on the realistic images generated by the projection and style transfer stages, we employ CLIP to extract their visual features $f_{ij} = f_v(x_{ij}')$. In the textual branch, we use “a photo of a [class]” as the prompt and encode their textual features as the zero-shot classifier $W_t \in \mathbb{R}^{K \times D}$. Furthermore, the classification logit $s_{ij}$ for each view $i$ and each style transfer text guidance $j$ are calculated separately as follows:

$$\text{logit}_{s_{ij}} = f_{ij}W_t^T, \forall i \in \{1, ..., M\}, j \in \{1, ..., K\}$$

For each style transfer text guidance $j$, the classification logit $s_j \in \mathbb{R}^{1 \times K}$ is defined as $MaxP(\text{logit}_{s_{1j}}, ..., \text{logit}_{s_{Mj}})$, where $MaxP(\cdot)$ represents taking the maximum value of each column in the matrix. Then, the probability matrix $P \in \mathbb{R}^{K \times K}$ is generated by $P = [\text{softmax}(\text{logit}_{s_{1j}}); ..., \text{softmax}(\text{logit}_{s_{Kj}})]$.

We design two strategies to calculate the final probability vector $p = [p_1, p_2, ..., p_K] \in \mathbb{R}^K$ for each classification. Each of them includes a global logits part and a local one in order to make full use of raw probability distributions of each diffusion result from each view on all the classes. The first strategy is given by

$$p = \beta_1p_{\text{glo}} + \beta_2p_{\text{loc}} \quad (3)$$

where

$$p_{\text{glo}} = \left[ \sum_{i=1}^{K} P_{i1} \mathbb{I}(P_{i1} \leq P_{11}), ..., \sum_{i=1}^{K} P_{iK} \mathbb{I}(P_{iK} \leq P_{KK}) \right] \quad (4)$$

$$p_{\text{loc}} = [P_{11}, ..., P_{KK}] \quad (5)$$

and $\beta_1$ and $\beta_2$ are hyper-parameters. $p_{\text{glo}}$ represents summing all values in the matrix that are no more than the diagonal by column, which represents the global information, and $p_{\text{loc}}$ returns the diagonal entries of $P$ that represent the probabilities of the realistic images generated by text guidance $j$ being classified into category $j$, which provides local information. We illustrate the detailed computation in our supplementary material. The second strategy to aggregate the matrix $P$ to calculate the final logits is as follows:

$$p = \text{norm}(p_{\text{glo}}) \ast p_{\text{loc}} \quad (6)$$

where

$$p_{\text{glo}} = \left[ \sqrt[\kappa]{\prod_{i=1}^{K} P_{1i}}, ..., \sqrt[\kappa]{\prod_{i=1}^{K} P_{Ki}} \right]$$

$$p_{\text{loc}} = \left[ \max_{i \in \{1,2,...,K\}} P_{ii}, ..., \max_{i \in \{1,2,...,K\}} P_{Ki} \right]$$

and

$$\text{norm}(p) = (p - \min(p)) / (\max(p) - \min(p))$$

Each entry in $p_{\text{glo}}$ represents the global feature of each category, which is defined as the geometric mean of the probabilities of that category across all style transfer results. Each entry in $p_{\text{loc}}$ represents the local feature for each category and is obtained from the diffusion result that is most similar to that category itself. Detailed clarification of these equations are shown in the appendix. The experiment results demonstrate that these two calculation strategies have their own strengths and weaknesses on different datasets.

To better illustrate the calculation process of probability matrix $P$ mentioned in equation 3 and equation 5, we give the matrix in Fig. 4 as an example. In equation 6, while calculating $p_{\text{glo}}$, numbers in blue boxes, which are bigger than numbers in orange boxes, are ignored.

3600
3.3.2 Few-Shot 3D Classification

In the context of few-shot classification tasks, a primary innovation of our model lies in the generation of style prompts. The image branch of the model remains identical to the one used in zero-shot classification tasks. As detailed in Section 3.2.3, we assume that the point-based network consists of $B$ blocks. To incorporate domain-specific characteristics, we establish $B$ meta-nets as style projectors $P_b(\cdot; \theta_b)$, $\forall b \in \{1, ..., B\}$ to encode domain characteristics into $B$ prefix features $c_b$, where $\theta_b$ represents the parameters of the $b$-th meta-net. Assuming that the textual feature from the $b$-th block is $D_b$-dimension, the total dimension of our textual encoder is $D = \sum_{b=1}^{B} D_b$. Specifically, $c_b \in \mathbb{R}^{D_b}$ is computed by $c_b(x) = P_b(F_b(x); \theta_b)$.

To generate a style-prompt for a given $(x, y)$ pair, where $x \in \mathbb{R}^{2048 \times 3}$ is an original 3D point cloud input, we define the full style prompt $t_y$ by

$$t_y = [c_1(x)][c_2(x)][c_B(x)]|CLSy|,$$

where $|CLSy|$ is the word embedding of label $y$. Subsequently, a zero-shot classifier $W_f \in \mathbb{R}^{D \times K}$ can be generated.

Following the computation of logits, as in zero-shot classification, we establish another trainable module, using a convolutional neural network to fuse logits $s_i$ of each view $i \in \{1, ..., M\}$, illustrated in the right portion of Fig. 3.

### 4. Experiments and Results

#### 4.1. Zero-Shot Classification

We evaluate the zero-shot classification performance of DiffCLIP on three well-known datasets: ModelNet10 [32], ModelNet40 [32], and ScanObjectNN [28]. For each dataset, we require no training data and adopt the full test set for evaluation. For the pre-trained CLIP model, we adopt ResNet-50 [9] as the visual encoder and transformer [29] as the textual encoder by default, and then try several other encoders in the ablation studies. We initially set the prompt of stable diffusion as “a photo of a [class], best quality, extremely detailed”, and then we try multiple prompts to adapt the three different datasets later.

**Performance.** In Table 1, we present the performances of zero-shot DiffCLIP for three datasets with their best performance settings. Without any 3D training data, DiffCLIP achieves an accuracy of 43.2% on OBJ_BG of ScanObjectNN dataset, an accuracy of 82.4% for zero-shot classific-
Ablation Studies. We conduct ablation studies on zero-shot DiffCLIP concerning the importance of stable diffusion plus ControlNet module, as well as the influence of different projection views and projection view numbers on ModelNet10. For zero-shot DiffCLIP structure without style transfer stages, in which the depth maps are directly sent into the frozen image encoder of CLIP, the best classification accuracy reaches 58.8%, while implementing ViT/16 as the image encoder of CLIP. Hence, we can see the style transfer stage increases the accuracy by 21.8%. Detailed ablation results are shown in Table 2.

In terms of the number of projected views, we try 1 (randomly selected), 1 (elaborately selected), 4, and 8 projection view(s) on ModelNet10. In Table 3, records are taken from a bird’s-eye view angle of 35 degrees, and the angles in the table represent the number of degrees of horizontal rotation around the Z-axis. The result of one elaborately selected view from a -135° angle outperforms others. Due to the characteristics of ModelNet10 dataset that all the objects are placed horizontally, projection from 4 views does not show obvious advantages. But the performance of the method on ModelNet40 is improved from multi-view projection because objects have different spatial arrangements, as shown in Table 5. We also ablate on the choice of different visual backbones, shown in Table 4. The results suggest that ViT/16 yields the best results in our framework.

Prompt Design. In DiffCLIP, we design several different prompts as both the textual branch of the CLIP model and the style prompt input of stable diffusion. For ModelNet10 and ModelNet40, we implement a default prompt for stable diffusion. The performance of multiple prompt designs for CLIP on ModelNet10 is shown in Table 6. We try to place the class tag at the beginning, middle, and end of sentences, among which class tag located at the middle of the sentence performs best. Specifically, real-world objects in ScanObjectNN are incomplete and unclear, we elaborately design its diffusion style prompt as “a photo of a [class], behind the building, best quality, extremely detailed”, in order to generate an obstacle in front of the target object, which is shown to be more recognizable for CLIP’s image encoder.
4.2. Few-shot Experiments

We experiment DiffCLIP with the trainable Style-Prompt Generation Module and the trainable Multi-view Fusion Module under 1, 2, 4, and 8 shots on ModelNet10. For K-shot settings, we randomly sample K point clouds from each category of the training set. We set the learning rate as 1e-4 initially to train the meta-nets in the style prompt generation module, and then turn the learning rate of the meta-net into 1e-7 and train the multi-view fusion block. Due to the limitation of memory of our GPU, the training batch size is set as 1. Default prompts are used for stable diffusion.

Performance. As is shown in Table 7, we compare few-shot classification performance of DiffCLIP to PointNet [16] and PointNet++ [17] in terms of K-shot classification accuracy. DiffCLIP surpasses PointNet in performance and approximately approaches PointNet++. It reaches an accuracy of 79.0% under 8-shot training.

5. Discussion and Limitations

In zero-shot 3D classification tasks, DiffCLIP achieves state-of-the-art performance on OBJ_BG of ScanObjectNN dataset and accuracy on ModelNet10 which is comparable to state-of-the-art. In few-shot 3D classification tasks, DiffCLIP surpasses PointNet in performance and approximately approaches PointNet++. However, our model does have some notable room for improvement. First, DiffCLIP’s performance on zero-shot tasks needs more exploration, such as projecting 3D point cloud into more views, designing better functions to calculate the logits vector $p$. Second, DiffCLIP’s performance on few-shot tasks are still not that good which needs further improvement, such as fine-tuning on initial CLIP encoders as well as ControlNet. Moreover, the ability of DiffCLIP processing some other 3D tasks including segmentation and object detection is expected to be explored. We plan to leave these for future work.

Limitations. First, the implementation of stable diffusion introduces a time-intensive aspect to both the training and testing of the model due to the intricate computations required in the diffusion process, which can elongate the time for model execution. We also acknowledge that the scale of our pre-training dataset for the point transformer is presently limited. This constraint impacts the performance of DiffCLIP. A larger, more diverse dataset would inherently provide a richer source of learning for the model, thereby enhancing its capability to understand more data.

6. Conclusion

In conclusion, we propose a new pre-training framework called DiffCLIP that addresses the domain gap in 3D point cloud processing tasks by incorporating stable diffusion with ControlNet in the visual branch and introducing a style-prompt generation module for few-shot tasks in the textual branch. The experiments conducted on ModelNet10, ModelNet40, and ScanObjectNN datasets demonstrate that DiffCLIP has strong abilities for zero-shot 3D understanding. The results show that DiffCLIP achieves state-of-the-art performance with an accuracy of 43.2% for zero-shot classification on OBJ_BG of ScanObjectNN dataset and a comparable accuracy with state-of-the-art of 82.4% for zero-shot classification on ModelNet10. These findings suggest that the proposed framework can effectively minimize the domain gap and improve the performance of large pre-trained models on 3D point cloud processing tasks.

<table>
<thead>
<tr>
<th>Method</th>
<th>ModelNet10</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1-shot</td>
</tr>
<tr>
<td>PointNet [16]</td>
<td>42.9</td>
</tr>
<tr>
<td>PointNet++ [17]</td>
<td>53.2</td>
</tr>
<tr>
<td>DiffCLIP</td>
<td>45.8</td>
</tr>
</tbody>
</table>

Table 7. Few-shot classification accuracy (%) of DiffCLIP on ModelNet10 with different shot numbers.

<table>
<thead>
<tr>
<th>Step for diffusion</th>
<th>Time consumption</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>94min</td>
<td>71.7</td>
</tr>
<tr>
<td>5</td>
<td>304min</td>
<td>80.7</td>
</tr>
<tr>
<td>10</td>
<td>549min</td>
<td>82.4</td>
</tr>
<tr>
<td>15</td>
<td>862min</td>
<td>82.2</td>
</tr>
<tr>
<td>20</td>
<td>1061min</td>
<td>80.6</td>
</tr>
</tbody>
</table>

Table 8. Time consumption of zero-shot inference and accuracy(%) of DiffCLIP with various diffusion steps on ModelNet10.
References


