Gaussian Splatting Projection in PointCLIP

anonymous

July 23th 2025

Abstract

Recent studies have advanced PointCLIP with variants achieving high accuracy, even when projected 2D images from 3D point clouds differ significantly from CLIP's original training distribution. While techniques such as GPT-based prompt engineering have improved performance, incorporating large language models (LLMs) remains computationally expensive. Additionally, existing projection methods often fail to preserve the geometric structure of 3D data, limiting classification accuracy. Effective 2D projections must render object shapes accurately and preserve fine-grained internal features to distinguish between visually similar classes.

We propose a method that integrates 3D Gaussian splatting into the 2D projection process to preserve key 3D geometric characteristics and enhance the representational power of the resulting 2D images. As far as we know, this is the first work that utilizes Gaussian splatting in the 2D projection process for this purpose. Gaussian splatting offers a compact and expressive way to represent point distributions and geometry by learning Gaussian parameters such as position, scale, and orientation. Our model is lightweight compared to transformer-based alternatives and does not require LLMs.

The proposed method achieves 96.15% top-1 accuracy on ModelNet40 and 99.91% on ScanObjectNN under a 16-shot setting, using only about 1.56 million parameters.

1 Introduction

(Zhang et al. 2021f) enables a 2D pretrained model like CLIP to process 3D point clouds by projecting them into 2D images. Since (Radford et al. 2021b) is trained on large-scale 2D natural images, directly applying it to 3D data leads to a domain gap that degrades performance. Just with a limited 3D dataset, training inevitably leads to poor feature representations. Cheraghian et al. (2019b) attempted to address this issue by introducing an additional loss to prevent the predicted class distribution from being skewed. However, this approach still falls short of being a fundamental solution. And (Bahng et al., 2022a) demonstrated that applying appropriate transformations to the input through visual prompting enables pretrained CLIP to be highly robust to distribution

gaps between the input and its training data. Thus, the projection strategy plays a key role in PointCLIP's effectiveness. Additionally, as CLIP relies on contrastive learning between image and text pairs, the choice of textual input also affects performance. PointCLIP v2 demonstrated that GPT-based prompt engineering can improve alignment between text and projected images, enhancing accuracy. However, relying on large language models (LLMs) introduces significant computational cost.

Gaussian Splatting is a recent method for modeling 3D scenes using multivariate Gaussians defined by parameters such as covariance, scale, and opacity. Unlike NeRF, which uses MLPs for volumetric rendering, Gaussian Splatting enables faster and more efficient rendering. Prior works have focused on photorealistic rendering, but in this paper, we propose using Gaussian Splatting to improve 3D-to-2D projection quality for CLIP-based classification. Specifically, we determine the opacity of each Gaussian using local linearity and density in the point cloud, allowing the projection to preserve both object boundaries and fine-grained internal features.

Many prior projection methods, such as silhouette and depth maps, capture overall shape but often fail to preserve internal geometry, which limits their compatibility with CLIP. And (Ghose et al. 2024b) introduced translation from sparse points to 2d image. But still performance limit exists. To mitigate this, recent works have introduced additional visual cues or used voxel-based filtering to reduce the domain gap. Similarly, GPT-generated prompts have been used to improve text embeddings, but at the cost of increased resource usage.

In contrast, our method leverages Gaussian Splatting to generate projection images that more accurately reflect 3D structure without requiring LLMs or Transformer-based components. This leads to improved compatibility with CLIP's pretrained encoder, while remaining lightweight and computationally efficient.

2 Related Works

(Kerbl et al. 2023b) is a recent and novel approach in the field of 3D vision. This method exhibits highly expressive reconstruction capabilities. It generates Gaussian distributions characterized by learnable parameters such as opacity, scale, spherical harmonics (SH), covariance, position, and rotation. These parameters are optimized via backpropagation, allowing the model to increase the weight of important points, enlarge their scale, and represent how broadly a point is distributed in 3D space through the covariance. (Lee et al. 2025c) focus on covariance regularization as a central strategy to mitigate the trade-off between blurry renderings from large covariance values and sparsity from overly small ones. Huang et al. (2024a), in particular, proposed an optimized projection strategy on a separate projection plane to reduce the mathematical approximation error introduced during the projection process of conventional Gaussian Splatting, based on a Taylor expansion.

(Huang et al. 2022c) proposed a method to transfer a 2D pretrained CLIP

model to the 3D vision domain by using depth maps for transfer learning. (Su et al. 2015a) propose a CNN architecture to better extract features from 2D images rendered from 3D shapes. It was an attempt to leverage well-pretrained 2D architectures. (Zhu et al. 2022k) improves 2D projection realism by retaining only the minimum depth value when multiple points fall into the same grid cell, preserving occlusion consistency. To enhance spatial continuity and produce CLIP-friendly renderings, the method applies densification through min pooling and smooths the result using Gaussian filtering to remove noise.

There have also been studies that utilize 3D and 2D data in parallel. (Li et al. 2024a) leverage CLIP to obtain image and text features, and use Gaussian Splatting to align 3D data with 2D image and text representations. (Guo et al. 2023a) employ both 3D and 2D data, using a masking technique to reconstruct both modalities simultaneously.

3 Method

Formula (1) is part of the 3D Gaussian distribution, where the covariance matrix serves as a weighting factor in the distance computation.

$$\exp\left(-\frac{1}{2}(\mathbf{x}-\mu)^\top \Sigma^{-1}(\mathbf{x}-\mu)\right) \tag{1}$$

The covariance matrix can be decomposed as in Equation (2):

$$\Sigma = RS^2R^\top \tag{2}$$

To ensure that the covariance matrix remains symmetric positive definite (SPD), and to impose the constraint that each scale value is strictly positive, the covariance is decomposed into a rotation and scale formulation. The rotation and scale matrices are processed through separate MLP blocks.

3.1 Gaussian Splatting to 3D Point Cloud

We first compute virtual centers by averaging each point in the point cloud with its neighboring points. These centers serve as inputs to the rotation prediction module. We use a two-layer MLP with GELU activation to predict the axis-angle rotation vector:

$$\textbf{axis angle} = W_2 \cdot \phi(W_1 \cdot \mathbf{x} + \mathbf{b}_1) + \mathbf{b}_2 \tag{3}$$

$\mathbf{x} \in R^{B\times N\times 3}$, where N is the number of points in the point cloud for each sample. The predicted axis-angle vector is further processed using the Rodrigues' rotation formula to obtain the Rotation matrix. The scale vector is computed as follows:

$$\mathbf{s} = W_2 \cdot \phi(W_1 \cdot \mathbf{x} + \mathbf{b}_1) + \mathbf{b}_2 \tag{4}$$

$$\mathbf{s} = \mathit{Hardtanh}_{[a,b]}(\mathit{Softplus}(\mathbf{s})) \tag{5}$$

The resulting vector $\mathbf{s} \in R^{B\times N\times 3}$ is then converted into a diagonal matrix $Scale \in R^{B\times N\times 3\times 3}$ by embedding it on the diagonal, where a and b are hyperparameters that define the clipping range of the Hardtanh activation. The function that returns opacity takes individual points from the point cloud and their neighboring points as input. This function operates in two directions. The first involves assessing the local linearity of the point cloud using the singular value decomposition (SVD) method. The second evaluates the density of the point cloud by counting the number of neighboring points within a certain radius around each point.

3.2 Projecting 3D Point Cloud to 2D Image

In our method, the 3D covariance matrix (*3D Cov*) is projected onto three 2D planes—front, top, and left—resulting in 2D covariance matrices. These 2D covariances are used to compute the exponent term in the 2D Gaussian distribution. This exponent captures the influence of each Gaussian on a given pixel.

$$weight = opacity \cdot importance\_score \cdot exponent \tag{6}$$

The *importance score* is a learnable parameter optimized during training, whereas the *exponent* term originates from the 2D Gaussian kernel and modulates each pixel value based on its distance from the projected Gaussian center. Given a 3D point cloud with N points and a batch size of B, the resulting tensor has shape $B \times N \times H \times W$.

We then perform a summation along the N dimension to aggregate the contributions of all Gaussians at each pixel. The resulting *weighted map* reflects the cumulative influence of all N Gaussians, modulated by their individual importance scores and opacity values. This map represents the appearance of the object indicated by the original 3D point cloud from a specific viewpoint, to the extent that it can be visually recognized at a glance.

3.3 postprocessing

2D images generated through Gaussian splatting often appear sparsely populated with discrete points. When the scale values used in splatting are too large, fine structures can become blurred, and the boundaries between internal features of the object may become indistinct. To address this, the scale is treated as a hyperparameter and adjusted accordingly.

From the weighted contributions of all Gaussians described in Projection section, we obtain a weighted map that reflects the aggregated influence of all projected 3D Gaussians on each 2D pixel. This weighted map is then post-processed to enhance visual coherence. Specifically, a Gaussian blur is applied to smooth the sparse and discrete patterns, making the rendered image more continuous and similar in appearance to natural images.

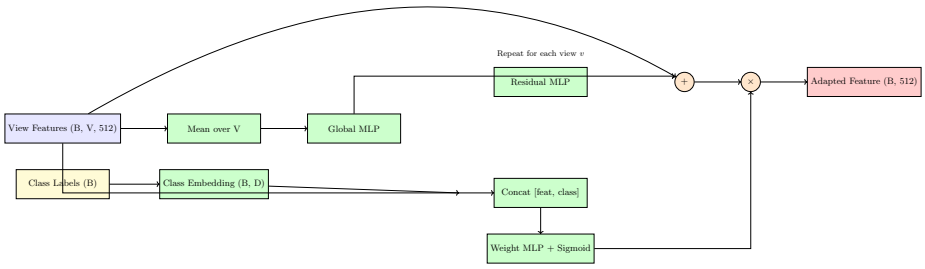

Figure 1: Architecture of our Class-Conditional Inter-View Adapter. Each view is conditioned on the class label to compute adaptive weights, which modulate the transformed features.

3.4 class view adapter

As shown in Figure 1, Each 2D image corresponding to a different view is passed through an MLP block and then fused into a single image. During this process, view-specific weights are applied, and these weights are assigned differently for each class. The MLP block is implemented as a ResNet block by incorporating residual connections.

3.5 Loss

The global feature obtained after passing through the class view adapter is used as the input to the CLIP image encoder. The corresponding class label is fed into the CLIP text encoder to obtain a text feature. The similarity logits are then computed as:

$$logits = 100 \cdot \mathbf{ImageFeature} \cdot \mathbf{TextFeature}^{\top} \tag{7}$$

Finally, the logits and class labels are used to compute the cross-entropy loss.

4 Experiments

In order to enhance generalization not only on the test sets of the two datasets but also in real-world scenarios, it is crucial to prevent overfitting caused by the class adapter. This is particularly important because the distribution between the train and test datasets is often overly similar and unrealistic compared to real-world settings. To address this, the projected map must preserve the inherent characteristics of the original 3D point cloud that are visually indicative of its class. Furthermore, to distinguish between highly similar classes, even subtle features—which are often lost during the projection—must be retained. To this end, it is important to prevent the 3D covariance and scale of the Gaussian distributions derived from individual points from becoming excessively

Dataset	Backbone	Method	4-shot	8-shot	16-shot
4*ModelNet40	ViT-B/16	PointCLIP v2 (10 views)	–	–	89.55
		Ours (3 views)	86.22	92.75	**96.15**
	ViT-B/32	PointCLIP v2 (10 views)	–	–	88.05
		Ours (3 views)	78.20	87.60	**92.91**
4*ScanObjectNN	ViT-B/16	PointCLIP v2 (10 views)	–	–	55.81
		Ours (3 views)	88.63	99.78	**99.91**
	ViT-B/32	PointCLIP v2 (10 views)	–	–	51.01
		Ours (3 views)	93.61	94.14	**99.87**

Table 1: Few-shot classification accuracy (%) comparison between PointCLIP v2 (10 views) and **Ours (3 views)** under different backbones and shot settings.

large, which would result in a blurry image. Conversely, if these values are too small, excessive Gaussian blurring can lead to feature distortion, effectively causing the same degradation as large-scale smoothing. Thus, appropriate value adjustments and clamping are necessary to maintain clarity and feature integrity in the projected images.

4.1 Settings

All experiments were conducted on a single NVIDIA A100 GPU. The datasets used were ModelNet40 and ScanObjectNN, both of which provide officially separated training and test splits. For few-shot training, we randomly sampled the specified number of examples (e.g., 4, 8, or 16) from the training set and evaluated the model's performance on the test set. The text prompts used for CLIP were directly derived from the class labels provided in each dataset, allowing the evaluation to rely purely on the expressiveness of the image projection process. For the visual encoder that receives the projected 2D images, we used ViT-B/16 and ViT-B/32. Only three views front, top, and left were used for projection. And CLIP remains frozen to prevent it from being updated during training.

4.2 2D visualiation

4.3 Performance

As shown in **Figure 2**, our proposed method demonstrates significant improvement across all presented views. In particular, it achieves near-perfect accuracy on challenging datasets like ScanObjectNN, which many models struggle with. This improvement is not only due to the clearer delineation of external boundaries, but also because Gaussian splatting effectively preserves internal features that are often lost yet critical for class discrimination. In addition, it shows almost perfect accuracy on ScanObjectNN using ViT-B/16, which is known to be more difficult than ModelNet40 due to noise and background clutter. Even

when switching to ViT-B/32, the performance remains robust, suggesting that our approach closely follows the natural image structure.

4.4 Ablation

As shown in Table 1, when the backbone CLIP model is changed from ViT-B/16 to ViT-B/32 under the 16-shot setting, the performance on ModelNet40 slightly decreases. However, our method still outperforms PointCLIP v2 by more than 3% even with fewer views.

5 Conclusion

Our model demonstrates significant performance improvements over existing PointCLIP V2 models, regardless of whether ViT-B/16 or ViT-B/32 is used. Notably, it achieves strong performance using only three views, leading to two key implications. First, an appropriate projection method can preserve the geometric structure of data even with a limited number of views. Second, 3D data can be processed and inferred with reduced computational resources. Moreover, since our method does not rely on any prompt engineering, it suggests the potential for further performance gains and improved generalization to additional datasets. Additionally, our results indicate that Gaussian splatting is not only effective for realistic reconstruction but also excels at preserving essential object features. As future work, we plan to extend our approach from classification to segmentation tasks.

6 References

reference can not be written such as have to change.

what about modelnet40 scanobject data citation?

clip (Radford et al. 2021b)

3D Gaussian Splatting for Real-Time Radiance Field Rendering (Kerbl et al. 2023b)

pointclip v2 (Zhu et al. 2022k)

[2112.02413] PointCLIP: Point Cloud Understanding by CLIP (Zhang et al. 2021f)

CLIP-based Point Cloud Classification via Point Cloud to Image Translation (Ghose et al. 2024b)

Multi-view Convolutional Neural Networks for 3D Shape Recognition (Su et al. 2015a)

Micro-splatting: Maximizing Isotropic Constraints for Refined Optimization in 3D Gaussian Splatting (Lee et al. 2025c)

mitigating the hubness problem for zero shot learning of 3d objects. (Cheraghian et al. 2019b)

exploring visual prompts for adapting large scale models (bahng et al. 2022a)

2d-3d joint masked autoencoders for 3d point cloud pre training (Guo et al. 2023a)

clip2point: transfer clip to point cloud classification with image depth pre training (Huang et al. 2022c)

On the Error Analysis of 3D Gaussian Splatting and an Optimal Projection Strategy (Huang et al. 2024a)

GS-CLIP: Gaussian Splatting for Contrastive Language-Image-3D Pretraining from Real-World Data (Li et al. 2024a)