Point-CERMIT: Point Cloud gEneration Robust Multi-Image neT

Anonymous WACV Algorithms Track submission

Paper ID *****

Abstract

Point clouds play a crucial role in defining the underlying structure of models. Generating high-quality point clouds from photographs or renderings captured from multiple perspectives is both a critical and challenging task, fundamental to advancing our capabilities in 3D modeling and perception. Currently, there are both single-step and multi-step approaches to point cloud generation. Multi-step methods typically start with an initial cloud and then refine its quality. The single-step methods, while generating point clouds, rely on non-recurrent processes, which restrict the ability to produce structures with a flexible number of points. Multi-step approaches, on the other hand, are often tailored to a fixed number of images or renderings, generating a predetermined set of points. This limits the scalability and adaptability of these traditional methods to arbitrary sets of inputs. Our approach enables the rapid generation of point clouds with an arbitrary number of points from a set of input images. We first produce an initial cloud with a random number of points, then iteratively refine it by increasing the point count, thus enhancing the quality of the cloud. By leveraging a novel attention-based architecture, alongside a custom loss function and efficient local feature extraction, we address common challenges in point cloud generation, such as point clustering and spherical distributions, which can distort the object's structure—a critical aspect for maintaining the integrity of point clouds.

1. Introduction

Humans possess an exceptional ability to perceive the world around them, a level of understanding that current machine learning algorithms and neural networks struggle to achieve. The challenge with 3D object processing is that neural networks often fail to fully grasp spatial relationships and intricate details of objects, scenes, and surfaces. In contrast, humans can effortlessly perceive even the finest nuances in their environment, a gap that highlights the limitations of current AI models in capturing the complexity of the real world.

Point clouds are just as crucial for representing 3D objects as voxels or polygons. While polygons primarily define texture and voxels offer a less precise structural representation, point clouds capture the qualitative structure of an object. Moreover, both polygons and voxels come with significant drawbacks: they increase memory consumption, complicate representation, and add computational overhead. However, there are some good solutions to generate them [10, 17]. In contrast, point clouds are more memory-efficient, provide rich local and global information, and are easier to process and transform, making them highly effective for downstream tasks.

One of the intriguing characteristics of point clouds is their invariance to the order of points within the tensor. Rearranging the points doesn't alter the structure of the cloud, which poses unique challenges for both the architectures used to generate point clouds and the associated loss functions and metrics. For instance, traditional CNN [15] architectures are highly sensitive to data ordering, making them ill-suited for handling point clouds directly, thus requiring the development of new approaches. As for losses and metrics, the unordered nature of point clouds limits the choice of effective evaluation functions. The F-score in radius is often the most suitable metric, as it measures accuracy in small-radius neighborhoods around each point. However, it too has limitations—duplicating all points in the cloud can yield an F-score close to 0.8 in small radii, effectively doubling the point count without any real improvement in the cloud's quality.

Our objective is to develop architectures capable of generating and improving the quality of point clouds, i.e., increasing the number of points to enhance cloud quality, with a small number of parameters. These architectures will deliver excellent generation quality and work seamlessly with arbitrary datasets, whether they consist of a varying number of images or points.

In our approach to point cloud generation, we propose a novel architecture inspired by the Visual Transformer (ViT) framework [4], incorporating an Encoder-Decoder model grounded in attention mechanisms [20] that effectively capture positional information. To ensure adaptability, we developed a flexible batch generation mechanism, enabling the ViT Encoder to handle an arbitrary number of input images. For the task of point cloud refinement and quality enhancement, we introduce an additional attention-based architecture, tailored to further elevate the precision of generated clouds.

To address common challenges in point cloud generation, we designed a robust method for local point feature extraction, which significantly mitigates point clustering and enhances the capture of fine-grained details in 3D models. Additionally, we implemented a custom loss function specifically for point clouds, preventing excessive clustering of points and preserving the cloud's structural fidelity across both the generation and refinement phases.

This approach not only ensures scalability and flexibility but also consistently delivers high-quality results across a wide variety of datasets.

2. Related Work

Currently, no fast and scalable solutions exist for generating point clouds that can adapt to an arbitrary number of points or images. Several existing methods rely on parameter-heavy architectures, which make them less practical for widespread use [5]. While there are approaches that generate point clouds in a single step, these often suffer from low quality and are constrained by a fixed number of points [13]. Other methods, such as [1, 7, 14], deliver high-quality results, but their computational demands are significant, making them more suitable for large-scale datasets. Additionally, certain approaches [12, 21] generate a point cloud of 1024 points from a single image and then attempt to enhance its quality by increasing the number of points. However, these methods are also designed for a fixed number of points or images and still fall short of achieving the desired level of generation quality.

Many works don't have different versions of dataset normalization, but it is important when generating clouds. However, we decided to get rid of this bottleneck—to eliminate the need for painstaking data collection for generation, as well as to make it possible to generate a cloud of any scale and detail; also we tested our approach in different normalizations. In this paper, we used two-stage generation, training two different neural networks with their own architectures for this.

3. Method

We present a two-stage process for point cloud generation and refinement. First, a set of input images, of arbitrary quantity, is processed through a Visual Transformer (ViT) Encoder followed by an attention-based Decoder. This step produces an initial, low-resolution point cloud with n points. To enhance the cloud, this intermediate result is fed into a second attention-like architecture, comprising two Encoders and a Decoder, which outputs a higher-quality point cloud containing $n \times k$ points for any chosen k.

For both stages, we employ a loss function that combines the Chamfer loss [8] and a custom PointLoss. The Chamfer Distance, a well-established metric for measuring the closeness between two point clouds P and Q, is defined as:

$$d_{cd}(P,Q) = \sum_{p \in P} \min_{q \in Q} \|p-q\|^2 + \sum_{q \in Q} \min_{p \in P} \|q-p\|^2 \quad (1)$$

This distance metric provides an effective measure of the accuracy between point clouds, balancing computational efficiency with spatial proximity considerations. To further improve point distribution, we introduce a custom PointLoss function that penalizes clustering:

$$PointLoss = min(1, \frac{mean(\frac{0.002}{dist})}{5}) \quad (2)$$

where "dist" represents the tensor of pairwise distances between points in the two clouds, constrained within the interval $[0.00001, 0.01]$. This PointLoss Eq. (2) discourages excessive clustering by imposing stronger penalties when points converge too closely, thereby encouraging better spatial distribution. The loss formula (with the constant representing the point loss value, which is a tunable parameter) is as follows:

$$Loss(P,Q) = Chamfer + PointLoss \cdot constant \quad (3)$$

The combined use of Chamfer loss and PointLoss leads to a significant improvement in the overall quality and structure of the generated point clouds, effectively addressing issues such as point clustering, which frequently hinder traditional generation models.

3.1. Point Cloud Generation

The proposed architecture comprises a Visual Transformer Encoder and an attention-based Decoder. The ViT Encoder, built on the foundation of the Visual Transformer framework, effectively captures global contextual information from the input. The Decoder, utilizing an attention mechanism, plays a critical role in refining the generated point cloud by focusing on both local and global features, thereby ensuring a more accurate and detailed representation of the object structure. This combination allows for a high-quality and flexible generation of point clouds, adaptable to various input configurations.

3.2. ViT Encoder and Decoder

The ViT Encoder is built upon the Vision Transformer architecture, incorporating several key components: patch embeddings, which process an image as a sequence of patches—analogous to how sequences of words are processed in NLP; multi-head self-attention mechanisms; and rotational position embeddings [19], which allow for efficient encoding of spatial information. Additionally, the Encoder includes fully connected networks with SwiGLU activation [18] to enhance non-linear transformations. Layer Normalization [2] is applied to the attention layers to stabilize and optimize learning.

The Decoder consists of a linear transformation layer followed by attention layers. The input data first passes through the linear layer, which adjusts the dimensionality from the input size to the hidden representation, after which the attention layers refine and output the predicted point cloud. This setup ensures an accurate and high-quality generation of point clouds.

3.3. PreCERMIT

When the model receives a set of input images, it first processes them through the Encoder. The images are divided into fixed-size patches, which are subsequently converted into fixed-dimensional vector representations via a linear layer. These vectorized patches are then passed through the Decoder, which applies a series of transformations to the data, ultimately generating the output point cloud. This approach ensures an efficient and structured conversion from 2D image inputs to 3D point cloud representations, as illustrated at the top of Fig. 1a.

3.4. Quality Enhancement of Point Clouds

This architecture is composed of an Encoder, a Radius Encoder, and a Decoder, all built upon an attention-based framework.

3.5. Encoder, Decoder, Radius Encoder

The attention mechanism is utilized in both the Encoder and Decoder. While both Encoders share the same structure, the Radius Encoder is specifically designed to extract local features. In the multi-head attention mechanism employed, RMS normalization [22] is applied to normalize the query and key matrices, alongside the use of SwiGLU for enhanced activation. Furthermore, RoPE is integrated into each attention layer to effectively encode positional information and reduce the risk of model overfitting.

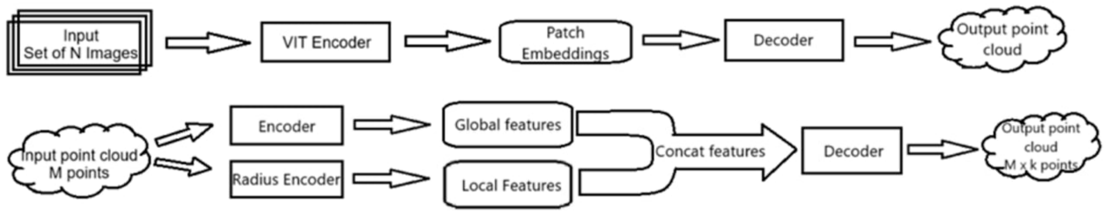
The Radius Encoder is designed similarly to the primary Encoder but plays a crucial role in capturing local features in addition to global features—particularly focusing on the spatial relationships between points. To accomplish this, for each point, the nearest `num` points within a specified radius `r` were identified, then concatenated with the original point cloud and processed through the Radius Encoder to extract these local features. Since CNN architectures interpret tensors like point clouds as structured data, the spatial arrangement of points becomes vital. This led to a key challenge during the nearest point search: how to handle cases where fewer points are found within the given radius than the required `num` nearest points. When the tensor was padded with zero vectors $[0, 0, 0]$, the quality of the generated point cloud (with an increased number of points) significantly deteriorated. This occurred because all point clouds were normalized within a unit cube, causing the architecture to generate a clustered cloud in the shape of an ellipse near the origin. Moreover, when the tensor containing the nearest points was shuffled, the points tended to cluster into a spherical formation.

To resolve this issue, several potential solutions were investigated. In cases where there were insufficient points within the specified radius, the tensor of nearest points could be augmented either by duplicating the point itself or by adding a point representing the average of each coordinate from the already identified points. After evaluation, the approach of adding the average of the selected points proved to be more effective than duplicating the point itself and was integrated into the final architecture. Additionally, incorporating tensor shuffling further improved the overall results and was included in the final solution.

The Decoder is composed of attention layers and linear layers. Initially, the input features are passed through a linear layer, which transforms them from their original dimensions to the hidden representation size. Following this, the data flows through a series of attention blocks. Finally, a second linear layer generates the predicted point cloud as the output.

3.6. CERMIT

When the model receives a point cloud as input, it first processes it through the Encoder, where dimensionality is expanded and the cloud is upsampled by the factor corresponding to the desired increase in quality (i.e., the number of points). This step captures the global features of the cloud. Simultaneously, point features are derived from the original cloud by adding dimensionality and upsampling. Local features are then extracted by identifying the nearest neighbors at three different radii, which are concatenated with the point cloud and passed through the Radius Encoder. Here, dimensionality is further expanded and upsampling occurs. Finally, all global and local features are concatenated and fed into the Decoder, producing a refined, higher-quality point cloud, as illustrated at the bottom of Fig. 1a.



(a) The diagram above illustrates the architecture used for generating a point cloud from an arbitrary set of N input images(i.e. PreCERMIT), while the diagram below shows the architecture responsible for refining a given point cloud of M points, producing an enhanced cloud with M $\times$ k points(i.e. CERMIT).

4. Implementation Details

4.1. Implementation of Generation Architecture

In this implementation, the Encoder processes a batch where each element is a concatenation of an arbitrary number of image tensors. To ensure proper functionality of the architecture, the embeddings produced by the ViT Encoder are averaged.

The ViT Encoder is composed of 12 attention layers, each with 8 heads, and includes RMS normalization within the attention blocks. The input image size is set to 512, with a patch size of 16. Each layer is further enhanced by incorporating RoPE to effectively encode positional information and reduce the risk of model overfitting. SwiGLU is for activation.

Similarly, the Decoder features 12 attention layers with 8 heads and uses RMS normalization within the attention blocks. RoPE and SwiGLU are also included to improve performance. A dropout rate of 0.2 is applied to prevent overfitting and optimize the learning process.

PreCERMIT is composed of a ViT Encoder and a Decoder. It is designed to handle a batch containing an arbitrary number of concatenated images corresponding to a single point cloud and is capable of generating an arbitrary number of points. However, the primary objective of the model is to initially generate a point cloud with a defined number of points (e.g., 1024 points) and subsequently upscale the number of points to enhance the cloud's quality.

4.2. Implementation of the Quality Enhancement Architecture

In this architecture, multi-head attention with 8 heads is utilized, with an embedding size of 64 for the Encoders and 128 for the Decoder. Both the Encoders and Decoder consist of 6 classic attention layers. Each attention layer incorporates RoPE for universal and high-quality generation of point clouds of different sizes, RMS normalization for stabilization, and SwiGLU for activation. Dropout is set to 0 to facilitate optimal learning.

The local feature extraction process is divided into several stages. In the first stage, the `num` nearest points within a specific radius `r` for each point are identified using the NearestNeighbors algorithm, which offers a speed advantage due to its KD-tree-based implementation. If fewer than `num` points are found, the average point (in each coordinate) of those already identified within radius `r` is computed and added to fill the remaining spots in the tensor. Afterward, the tensor is shuffled, a step that led to improved performance compared to not shuffling.

Next, the tensors of nearest points were concatenated with the original point cloud such that in the resulting tensor, the coordinates of the nearest points were placed next to the coordinates of the original points. Simultaneously, the resulting tensor was fed into the Radius Encoder, which extracted local features from it.

CERMIT consists of an Encoder, a Radius Encoder, and a Decoder. The features obtained from the Encoder were concatenated with the features obtained from the Radius Encoder and then fed into the Decoder. Additionally, a prediction function was written for the architecture, allowing for the generation of a point cloud of size $n \times \alpha^k$ without training a new instance of the class—this was achieved by applying the forward upsampling operation k times(α is upsampling parameter).

5. Data

For both architectures, the ShapeNetRendering dataset [3] was used because of the large amount of diverse data available. Before training, all data was normalized: images were resized to 512×512, and point clouds were scaled to fit within a cube with coordinates $[0, 1]$ or $[-1, 1]$ with mean coordinates equal to zero. The normalization of point clouds affects the choice of the radius parameter when evaluating model quality using the F-score. For example, if point clouds were scaled to fit within a cube with coordinates $[0, 1]$ and the radius parameter of the F-score is equal to 0.01 or 0.02, respectively, then if point clouds were scaled to fit within a cube with coordinates $[-1, 1]$, the radius parameter was considered equal to 0.02 and 0.04 respectively (since the cloud expanded 2 times).

For the PreCERMIT architecture, the batch generation process was implemented with $[0, 1]$ normalization. Given our requirement for a network that could handle an arbitrary number of input images, we developed a custom dataloader. This dataloader generates a random num $\in [1, 24]$ at the start of each iteration for every batch. The batch generation process then outputs a tensor containing num concatenated images along with one corresponding point cloud. The custom dataloader ensures that all samples within a batch share the same random num value, allowing for correct concatenation of the images across the batch.

For the CERMIT architecture, batch generation was similarly implemented with $[0, 1]$ normalization because of better results, and a standard `torch.dataloader` [16] was employed for data loading.

6. Results Discussion

6.1. Losses and Metrics

As previously discussed, we employed a combination of a custom loss function alongside Chamfer loss for training. The evaluation metric was the F-score, calculated within small radii around the points. Both the F-score and Chamfer loss were implemented using the Kaolin library [6].

In the PreCERMIT model, the best Chamfer loss achieved was 0.2, with a PointLoss of 4.15. For the CERMIT model, the best Chamfer loss reached 2.11, while the PointLoss was 2.47, and the F-score was 47.93. These results are detailed in Tab. 1.

6.2. Hyperparameters

The best performance was achieved using the following hyperparameters:

For the point cloud generation model (PreCERMIT):

The model underwent training for a total of 20 epochs, with the highest performance recorded at the 20 epoch. The point loss value was set to 0.01, with Adam [9] as the optimizer. A Cosine scheduler [11] was used, with updates occurring every 150 iterations. The initial learning rate was 0.0001, and both the training and validation batch sizes were 32.

For the quality enhancement model (CERMIT):

The model underwent training for a total of 20 epochs, with the highest performance recorded at the 18 epoch. The PointLoss was integrated into the loss function with a coefficient of 0.055. Adam served as the optimizer, with an initial learning rate of 0.0001, and A Cosine scheduler was applied with updates every 200 iterations. The batch size for both training and validation was 32.

Both models employed multi-head attention with 8 heads, and the embedding size was set to 64 for the Encoders and 128 for the Decoder. Each Encoder and the Decoder consisted of 6 attention layers. In CERMIT, local feature extraction was carried out by identifying the 10 nearest points within radii of 0.03, 0.04, and 0.05.

7. Conclusion

In this paper, we introduce two innovative architectures: one for point cloud generation and another for enhancing point cloud quality, both designed to handle an arbitrary number of input images or points. Our proposed approach, which first generates a point cloud and then refines its quality, demonstrates superior effectiveness compared to direct generation methods. Central to our quality enhancement strategy is an attention-based model, augmented with combination of two loss functions and a robust local feature extraction technique. This combination achieves high performance and effectively mitigates common challenges seen in traditional CNN or MLP-based architectures, such as point clustering, or the formation of undesirable spherical or elliptical shapes.

References

[1] Panos Achlioptas, Olga Diamanti, Ioannis Mitliagkas, and Leonidas Guibas. Learning representations and generative models for 3d point clouds, 2018. 2

[2] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization, 2016. 3

[3] Angel X. Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, Jianxiong Xiao, Li Yi, and Fisher Yu. ShapeNet: An Information-Rich 3D Model Repository. Technical Report arXiv:1512.03012 [cs.GR], Stanford University — Princeton University — Toyota Technological Institute at Chicago, 2015. 4

[4] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale, 2021. 1

[5] Haoqiang Fan, Hao Su, and Leonidas Guibas. A point set generation network for 3d object reconstruction from a single image, 2016. 2

[6] Clement Fuji Tsang, Maria Shugrina, Jean Francois Lafleche, Towaki Takikawa, Jiehan Wang, Charles Loop, Wenzheng Chen, Krishna Murthy Jatavallabhula, Edward Smith, Artem Rozantsev, Or Perel, Tianchang Shen, Jun Gao, Sanja Fidler, Gavriel State, Jason Gorski, Tommy Xiang, Jianing Li, Michael Li, and Rev Lebaredian. Kaolin: A pytorch library for accelerating 3d deep learning research. https://github.com/NVIDIAGameWorks/kaolin, 2022. 5

[7] Zixuan Guo, Yifan Xie, Weijing Xie, Peng Huang, Fei Ma, and Fei Richard Yu. Gaussianpu: A hybrid 2d-3d upsampling framework for enhancing color point clouds via 3d gaussian splatting, 2024. 2

[8] Tianxin Huang, Qingyao Liu, Xiangrui Zhao, Jun Chen, and Yong Liu. Learnable chamfer distance for point cloud reconstruction, 2023. 2

[9] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017. 5

[10] Ruoshi Liu, Rundi Wu, Basile Van Hoorick, Pavel Tokmakov, Sergey Zakharov, and Carl Vondrick. Zero-1-to-3: Zero-shot one image to 3d object, 2023. 1

[11] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts, 2017. 5

[12] Priyanka Mandikal and R. Venkatesh Babu. Dense 3d point cloud reconstruction using a deep pyramid network, 2019. 2

[13] Priyanka Mandikal, K L Navaneet, Mayank Agarwal, and R. Venkatesh Babu. 3d-lmnet: Latent embedding matching for accurate and diverse 3d point cloud reconstruction from a single image, 2019. 2

[14] K L Navaneet, Ansu Mathew, Shashank Kashyap, Wei-Chih Hung, Varun Jampani, and R. Venkatesh Babu. From image collections to point clouds with self-supervised shape and pose networks, 2020. 2

[15] Keiron O'Shea and Ryan Nash. An introduction to convolutional neural networks, 2015. 1

[16] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019. 5

[17] Guocheng Qian, Jinjie Mai, Abdullah Hamdi, Jian Ren, Aliaksandr Siarohin, Bing Li, Hsin-Ying Lee, Ivan Skorokhodov, Peter Wonka, Sergey Tulyakov, and Bernard Ghanem. Magic123: One image to high-quality 3d object generation using both 2d and 3d diffusion priors, 2023. 1

[18] Noam Shazeer. Glu variants improve transformer, 2020. 3

[19] Jianlin Su, Yu Lu, Shengfeng Pan, Ahmed Murtadha, Bo Wen, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding, 2023. 3

[20] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2023. 2

[21] Lemeng Wu, Dilin Wang, Chengyue Gong, Xingchao Liu, Yunyang Xiong, Rakesh Ranjan, Raghuraman Krishnamoorthi, Vikas Chandra, and Qiang Liu. Fast point cloud generation with straight flows, 2022. 2

[22] Biao Zhang and Rico Sennrich. Root mean square layer normalization, 2019. 3

HyperParameters		**Results**		
point loss value	number of near points	Chamfer loss $\times 10^2$	PointLoss $\times 10^2$	F-score $\times 10^2$
0.03	10	5.22	4.28	10.26
0.04	10	3.01	3.95	15.31
0.045	10	2.66	3.54	17.28
0.05	10	2.19	3.03	32.97
0.055	10	2.11	2.47	47.93
0.06	10	2.95	2.82	37.64
0.07	10	2.74	3.60	21.78
0.045	20	5.21	5.25	14.32
0.05	20	4.38	4.72	19.18
0.055	20	3.46	4.18	24.55
0.04	30	4.46	4.85	10.58
0.045	30	3.82	4.63	13.45
0.05	30	3.09	4.25	17.02
0.04	50	5.29	5.30	7.87
0.045	50	4.45	4.75	12.43
0.05	50	3.92	4.53	14.67

Table 1. HyperParameters and Results of CERMIT. Losses and metric are scaled by 100.