IET Computer Vision

WILEY

The Institution of Engineering and Technology

ORIGINAL RESEARCH OPEN ACCESS

PAD: Detail-Preserving Point Cloud Reconstruction and Generation via Autodecoders

Yakai Zhang[1] | Ping Yang[1] | Haoran Wang[1] | Zizhao Wu[1] | Xiaoling Gu[1] | Alexandru Telea[2] | Kosinka Jiri[3]

[1]Hangzhou Dianzi University, Hangzhou, China | [2]Department of Information and Computing Science, Utrecht University, Utrecht, the Netherlands | [3]Faculty of Science and Engineering, University of Groningen, Groningen, the Netherlands

Correspondence: Zizhao Wu (wuzizhao@hdu.edu.cn)

Received: 27 February 2024 | **Revised:** 17 February 2025 | **Accepted:** 28 April 2025

Handling Editor: Antonio Robles-Kelly

Funding: This study was funded by Zhejiang Provincial Natural Science Foundation of China (LGF21F020012). This work was partially supported by the Shaoxing Science and Technology Project (Grant 2023B41007).

Keywords: computer graphics | computer vision | convolutional neural nets | feature extraction | shape recognition

ABSTRACT

High-accuracy point cloud (self-) reconstruction is crucial for point cloud editing, translation, and unsupervised representation learning. However, existing point cloud reconstruction methods often sacrifice many geometric details. Altough many techniques have proposed how to construct better point cloud decoders, only a few have designed point cloud encoders from a reconstruction perspective. We propose an autodecoder architecture to achieve detail-preserving point cloud reconstruction while bypassing the performance bottleneck of the encoder. Our architecture is theoretically applicable to any existing point cloud decoder. For training, both the weights of the decoder and the pre-initialised latent codes, corresponding to the input points, are updated simultaneously. Experimental results demonstrate that our autodecoder achieves an average reduction of 24.62% in Chamfer Distance compared to existing methods, significantly improving reconstruction quality on the ShapeNet dataset. Furthermore, we verify the effectiveness of our autodecoder in point cloud generation, upsampling, and unsupervised representation learning to demonstrate its performance on downstream tasks, which is comparable to the state-of-the-art methods. We will make our code publicly available after peer review.

1 | Introduction

In recent years, point cloud learning has become a mainstream research area in the field of computer graphics, including point cloud classification [1–11], segmentation [12], reconstruction [13, 14], and generation [15–19]. Extracting high-fidelity and low-dimensional point cloud features is crucial for these tasks and forms the basis for subsequent point cloud operations. Many deep learning models have been proposed to encode point cloud data and have achieved significant results. Yet, due to the irregular, sparse, and unordered nature of point cloud data, point cloud learning faces some unique challenges. As such, an effective method for encoding point cloud data is a core challenge in point cloud learning research.

A widely used method for the above goal is to train a point cloud encoder. Training can be supervised using classification and segmentation labels [1–5], or unsupervised [20, 21], as commonly done in point cloud reconstruction [22] (sometimes also called self-reconstruction to distinguish it from surface reconstruction from point clouds). Recent studies have introduced attention blocks [3, 4] and graph convolution networks (GCNs) [5] to make a point cloud encoder network better capture its inputs. It is worth noting that the evaluation of such

encoders mainly uses their performance in point cloud classification and segmentation. For such tasks, the cloud encoding must only capture the distinctive features of different classes in order to accurately distinguish among them. In contrast, for high-quality point cloud *reconstruction*, the encoding must preserve information as fully as possible, including all shape details. Only a few encoders are specifically designed to achieve better point cloud reconstruction results. Such detail-preserving reconstructions are critical to downstream tasks such as latent space editing and translation [23, 24].

To obtain high-quality point cloud reconstruction encoding, most unsupervised point cloud encoding networks adopt an autoencoder architecture where the encoder maps the point cloud to a latent code space and the decoder generates a point cloud from the latent code. A good representation is expected to guide the decoder to generate a point cloud as similar as possible to the input cloud. Some methods aim to generate better point clouds by designing a decoder with prior knowledge about the point cloud surface, including FoldingNet [13] and AtlasNet [14]. These methods rely on Chamfer Distance (CD) or Earth's mover distance (EMD) which have been shown to be challenging metrics for guiding the network to learn accurate point cloud reconstruction [25, 26]. Other methods model the process of generating point clouds using continuous normalising flow, variable gradient fields, or denoising diffusion models [15, 18].

Although decoders have been continuously improved and the metrics for point cloud reconstruction have been enhanced, high-accuracy capturing of shape details in point cloud reconstruction has been limited so far and presents a bottleneck in the current state-of-the-art methods (see Figure 1).

To address these limitations, inspired by Variational Auto-Decoder (VAD) [27] and DeepSDF [28], we have designed a point cloud autodecoder architecture, called PAD, that achieves high-quality point cloud reconstruction and low-dimensional representations (see Figure 2). The autodecoder—not to be confused with an autoencoder—has been used in image decoder [27] and signed distance function (SDF) decoder [28] training. However, as far as we know, no work has used an autodecoder to map encodings to the point cloud domain. The main difference of our method is that our architecture includes only one decoder, and the point cloud representations are initialised as a set of learnable parameters that are optimised jointly with the decoder during training. During inference, the parameters of the point cloud decoder are frozen and only the corresponding point cloud representations of the test set are iteratively updated.

To show our method's scalability and flexibility (the fact that our decoder is not limited to a specific architecture) we incorporate our decoder in DPM [15], ShapeGF [16], and AtlasNet [14]. Our results show significant improvements in reconstruction accuracy compared to previous methods. Qualitative analyses show that our method preserves more shape details, such as chair back structures and chair leg types in Figure 1 compared to previous methods.

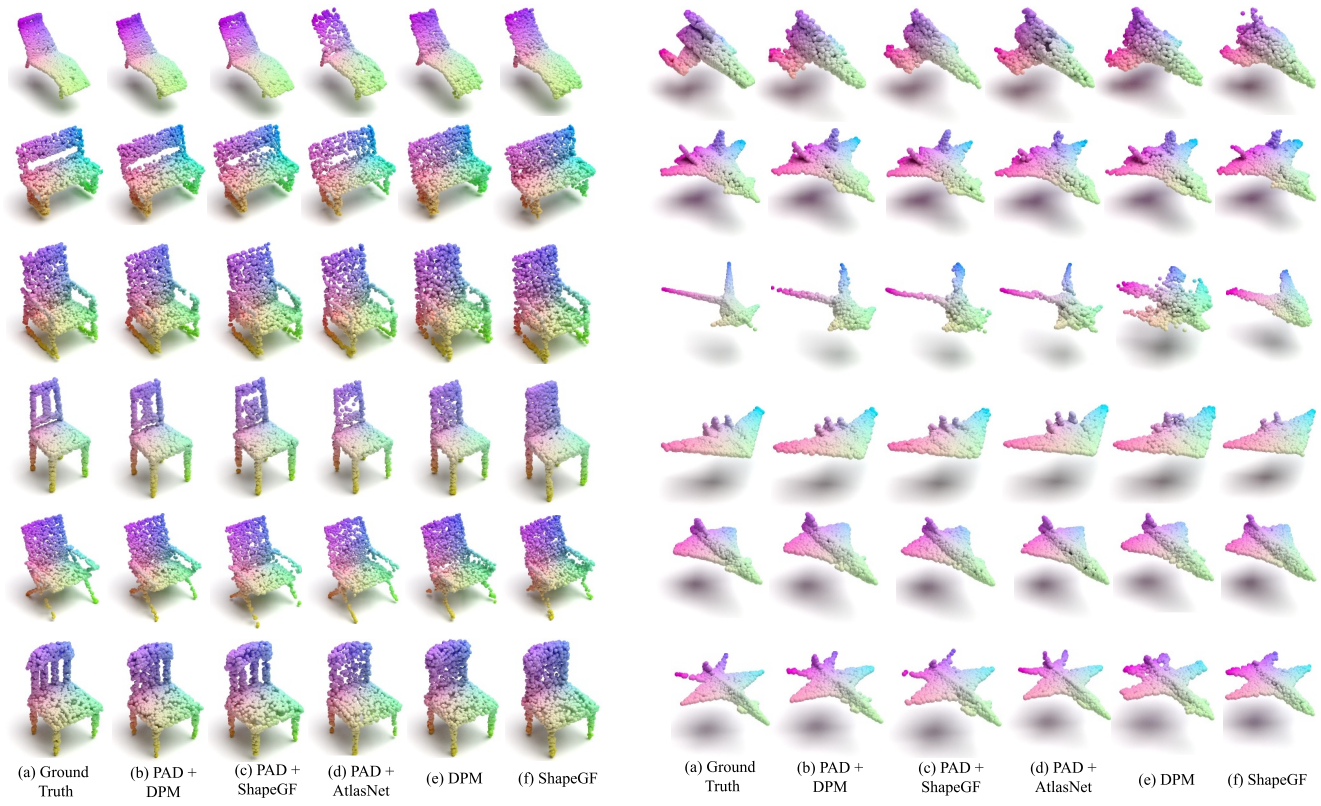


FIGURE 1 | Point cloud reconstruction results generated by PAD (ours) in combination with DPM, ShapeGF, and AtlasNet on chair and aeroplane datasets. It can be seen that our PAD method can preserve more shape details in the reconstruction of all decoder architectures, such as the hollow back of the chair and the protrusion of the tail. The shape edges reconstructed by PAD are generally smooth, while the autoencoder methods tend to generate point clouds including noise and outliers.

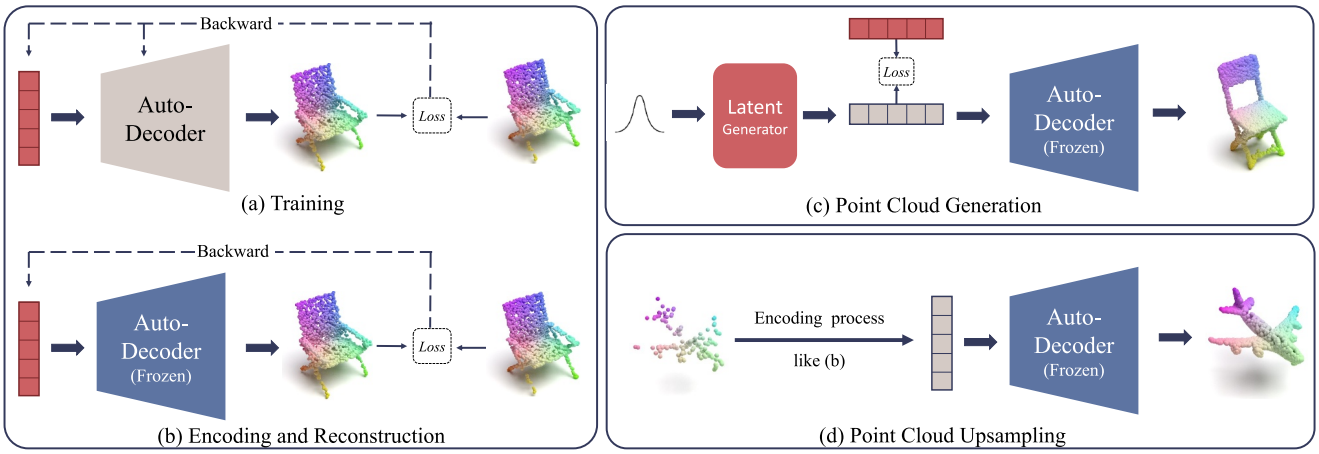


FIGURE 2 | The pipeline of our PAD method including (a) training, (b) reconstruction, (c) point cloud generation, and (d) upsampling. During the training process, the learnable latents and weights of the decoders are updated. For generating reconstruction results, the latents remain learnable while the decoders are frozen.

We show that our obtained point cloud representation improves downstream tasks by experiments on shape generation, point cloud upsampling, and unsupervised representation learning tasks using the pre-trained decoder. Our point cloud autodecoder achieves outcomes comparable to state-of-the-art methods.

In summary, our main contributions are as follows:

- We propose a detail-preserving point cloud reconstruction method, PAD, based on an autodecoder that does not require an encoder and can be integrated with existing decoders of various architectures.
- We use our autodecoder to design a framework for point cloud generation and upsampling, thus expanding the application scope of PAD.
- We show by quantitative and qualitative experiments that PAD has significantly higher reconstruction performance compared to previous methods and achieves evaluation results comparable to state-of-the-art methods in downstream tasks (point cloud generation, upsampling, and unsupervised representation learning).

2 | Related Work

We describe relevant work related to point cloud reconstruction (Section 2.1) and generation (Section 2.2).

2.1 | Point Cloud Autoencoders

Point cloud autoencoders have been widely studied in recent years and have shown great potential in various 3D computer vision tasks such as shape reconstruction [13–16, 18, 29] and unsupervised learning [20, 21]. The aim of a point cloud autoencoder is to learn a low-dimensional representation of the input point cloud while maintaining its essential geometric information. Most of such methods are based on encoder-decoder architectures, where the encoder maps the input point cloud to a low-dimensional latent code, and the decoder reconstructs the point cloud from this code.

2.1.1 | Decoders

Early point cloud decoders [23, 25, 26, 29, 30] treated point clouds as a fixed matrix. This setting meant that upsampling operations could not be directly applied and also ignored the permutation invariance of point clouds. To overcome these limitations, FoldingNet [13] and AtlasNet [14] proposed folding-based approaches that map 2D grids to the 3D point cloud surface. Both methods can select any number of points during shape generation, and the points on the surface patches are also unordered. Yet, in order to train the autoencoders, both FoldingNet and AtlasNet use CD or EMD as a metric to output the point cloud reconstruction compared to the input. As pointed out in refs. [18, 25], CD leads to local clustering of reconstructed points, while an accurate EMD computation is slow and its estimation may result in biased or noisy gradients.

Recent methods have approached point cloud decoding differently by modelling clouds as samples obtained from a 3D distribution. PointGrow [29] uses an auto-regressive model to recursively generate each point, but the generation process needs to determine the order of points. PointFlow models the point cloud distribution with normalising flows. Its generation process can be seen as gradually moving the sampled noise points to the shape surface. Similarly, ShapeGF [16] and DPM [15] propose methods for gradually restoring shapes. ShapeGF uses a score-based generative model where the network directly predicts the gradient of the logarithmic density field to compute the movement direction of the noise points. In contrast, DPM is based on a diffusion model that generates the target shape by gradually denoising the sampled points. These methods generally yield better reconstructions than methods that use CD or EMD for the reconstruction loss.

2.1.2 | Encoders

While the architecture of point cloud decoders has been evolving, point cloud encoders for reconstruction purposes often only involve the direct application of classification or segmentation encoders or their simple modifications. AtlasNet, PointFlow, ShapeGF, and DPM all use a simple PointNet-based encoder. FoldingNet, on the other hand, adds graph-based max-pooling layers to PointNet to increase the use of neighbourhood information. LOGAN [23], a point cloud translation network, designed an over-complete encoder based on PointNet++ [2]. However, its design is aimed at facilitating the disentanglement of style features and its reconstruction performance is slightly worse than that of PointNet++. Overall, these encoders are usually modifications of previous methods, and there has not been sufficient quantitative comparison. Considering the difference between classification and reconstruction (regression) tasks, existing point cloud encoders may not be fully suitable for reconstruction tasks.

2.2 | Other Point Cloud Generative Methods

Most existing point cloud generation approaches are based on generative adversarial networks (GAN). However, these approaches suffer from limited generation capacity due to the difficulty of designing a discriminator that is sensitive enough to the distribution of points. Some methods, such as [31], use variational autoencoder (VAE)-based architectures [32] to achieve point cloud generation. With the rising popularity of diffusion models [33], researchers have explored their potential in point cloud generation. Luo and Hu [15] proposed DPM, a diffusion probability model for denoising the generation process of point clouds, using a flow-based model to sample codes from prior distributions that guide the point cloud diffusion. PVD [17] directly denoises using a point-voxel CNN [12] to generate point clouds. LION [34] combines three-dimensional spatial diffusion similar to PVD with shape latent code diffusion to generate point clouds. It achieves excellent generation results, albeit at a higher computational cost.

3 | PAD Method

We now detail our PAD method (see Figure 2) in terms of its architecture, training, inference and advantages(Section 3.1), three different types of decoders (Section 3.2), and point cloud generation (Section 3.3) and upsampling (Section 3.4).

3.1 | Preliminaries

Given a point cloud $x \in X$ from a collection X of such clouds, point cloud reconstruction aims to obtain an encoding z of x so that, when passing x through a decoder D_θ parameterised by θ, one obtains a reconstructed cloud $\tilde{x} = D_\theta(z)$ as close to x as possible.

In an autoencoder, obtaining z often requires training the autoencoder and then using the trained encoder E to encode the point cloud to get $z = E(x)$. In contrast, an autodecoder only contains a decoder that reads a one-dimensional encoding z as input to output the reconstructed point cloud $\tilde{x} = D_\theta(z)$. Since there is no encoder, a set of learnable codes z must given to the autodecoder, typically one for each cloud in the training set. The decoding loss L is given by the following:

$$L(x, z, \theta) = \text{dist}(x, D_\theta(z)),$$

where $\text{dist}(\cdot, \cdot)$ is a distance function determined by the type of the decoder, for example, Chamfer Distance (CD) for AtlasNet or the mean squared error (MSE) for DPM and ShapeGF. The decoder's weights θ and the learnable codes z are updated based on the loss L. During inference, the decoder weights θ are no longer updated. The same loss function L is used to obtain the encoding of the test set shapes by updating the learnable codes of the training-set point clouds. This encoding of the test set is then input into the decoder to create the reconstructed point clouds.

Autodecoders perform well in small sample learning and personalised optimisation, preserving fine details while handling sparse or irregular point clouds more effectively than traditional autoencoders. By assigning independent latent codes to each sample, they simplify model design, reduce computational overhead, and enable more accurate shape reconstruction.

3.2 | Decoder

In an autodecoder, the decoder is not limited to a particular type—in theory, any point cloud generator/decoder that takes latent codes as input can be used. To show the versatility of our method, we use decoder frameworks based on DPM [15], ShapeGF [16] and AtlasNet [14] networks. We now discuss them next in turn.

The **DPM** method treats point clouds as a particle set in a continuously evolving thermodynamic system and models the generation process using a diffusion model. During forward diffusion, noise is gradually added to the input point cloud x_0 to obtain a set of points x_T that approximate the noise points sampled from a given distribution. This process is modelled as a Markov chain

$$q(x_{1:T} \mid x_0) = \prod_{t=1}^{T} q(x_t \mid x_{t-1}).$$

In the reverse diffusion phase, a neural network uses the encoding z of the point cloud x_0 to estimate noise at each step and gradually remove the noise to form the point cloud.

In the autodecoder, there is no encoder block included. Hence, when using this as an autodecoder, only the decoder module is used, and the loss function needs to be modified accordingly, leading to

$$L(x, z, \theta) = \sum_{t=2}^{T} \text{KL}\big(q(x_{t-1} \mid x_t, x_0) \,\big\|\, p_\theta(x_{t-1} | x_t, z)\big),$$

where p_θ is one time-step forward in the decoder D_θ and KL is the Kullback–Leibler divergence. When training the decoder,

the above optimisation goal is reflected in computing the MSE loss between the predicted and prior noises. For the inference process, the decoder runs iteratively, from $t = T$ downwards, to generate the point cloud output.

In **ShapeGF**, sampling point clouds entails moving a set of points sampled from the prior distribution to the high probability region of the shape. The method learns a gradient field of the logarithmic density that changes over time to guide noise points to move to the shape surface. Similar to DPM, this is also an iterative denoising generative model. During training, the optimisation aims to fit the noise from the noisy point cloud. In the autodecoder, the loss function for training one step is given by the following:

$$L(\sigma, x, z, \psi) = \|g_\psi\left(\tilde{x}, z, \sigma\right) - \frac{x - \tilde{x}}{\sigma^2}\|_2^2 \quad \text{with} \quad \tilde{x} \sim N(x, \sigma^2 I),$$

where ψ is the standard deviation of the Gaussian noise added to the input cloud in that step and g_ψ is the decoder part responsible for computing noise in each denoising step. During point cloud generation, a point set is first sampled from a standard Gaussian distribution, and then the decoder is used to gradually remove noise under the guidance of the code z, thereby yielding the target shape.

The **AtlasNet** method approximates a target surface by locally mapping a set of square patches onto the input shape, thus representing the complete shape. To generate point clouds, the decoder randomly samples multiple point sets on two-dimensional square patches and maps them to the three-dimensional space to fit the surface of the original shape under the guidance of the latent code. This network can be directly used in an autoencoder and trained with CD as the loss function. Yet, since the network uses different network parameters for each patch, it lacks the ability to perceive the shape globally, which results in poor performance in subsequent tasks (as observed in experiments). To overcome this limitation, we modified the network to use the same set of parameters to map all patches and provide patch embeddings to the network simultaneously with input latent codes. This setting enhances the global perceptual ability of the AtlasNet decoder. The parameter count of the modified network does not change with the number of patches, and the loss function is given by the following:

$$L(x, z, \theta) = \mathrm{CD}(x, D_\theta(z)).$$

3.3 | Point Cloud Generation

Point cloud generation aims to map the encoding sampled from a prior distribution to the point cloud space in order to generate point cloud shapes that are as realistic as possible. A good point cloud generator should not only ensure the realism of the generated point clouds, but also be able to generate shapes that *differ* from its training data. Given the excellent performance of autodecoders in reconstruction tasks, we design a generation framework based on autodecoders to leverage their capabilities for generating more detailed shapes. This also extends the application domain of our autodecoder framework.

We implement point cloud generation using autodecoders by training a latent diffusion model [34, 35] for the decoder's latent space. This model has 11 multi-layer linear blocks, each having 3 linear layers, a dropout layer, and a residual connection. When training the diffusion model, the input consists of representations of the point clouds from the training set rather than using a pre-trained point cloud decoder. During the generation phase, the diffusion model samples latent codes from the prior distribution and maps them to the latent space of the autodecoder, and then generates the corresponding point clouds by the decoder.

3.4 | Point Cloud Upsampling

Point cloud upsampling methods aim to generate a dense point set with the same shape as a given sparse point set while achieving distribution uniformity. Since the resolution of the autodecoder is adaptive, it can find the latent code for point clouds of arbitrary resolution—that is, it can do point cloud upsampling similar to an autoencoder. The process of point cloud upsampling is consistent with the reconstruction process (see Figure 2d). The pre-trained autodecoder encodes the low-resolution point cloud and then generates a high-resolution point cloud based on the encoding.

4 | Experiments and Results

We next present the validation of our PAD method. Section 4.1 outlines the datasets and metrics used in our experiments. Section 4.2 illustrates the point cloud reconstruction results, Sections 4.3 and 4.4 show our results for point cloud generation and upsampling, respectively. Section 4.5 discusses the performance of unsupervised representation learning.

4.1 | Experimental Setup

4.1.1 | Dataset

We measure the performance of our autodecoder for both reconstruction and generation using the ShapeNet dataset [36] which has 51,127 shapes across 55 categories. It is one of the most commonly used datasets in the field of shape generation. Following previous work [15], we randomly split this dataset into an 80% training set, 15% test set, and 5% validation set. For reconstruction, we conduct experiments on the full dataset and three subsets (aeroplane, chair, and car) separately. For generation, we conduct experiments on two subsets (aeroplane and chair). For the unsupervised representation learning task, we pre-train the autodecoder on the ShapeNet training set and evaluate classification accuracy on ModelNet10 and ModelNet40 [37]. For each shape, we randomly sample 2048 points and normalise the point clouds to zero mean and unit variance.

4.1.2 | Metrics

Following prior reconstruction tasks [15, 16, 18], we use CD and EMD to evaluate our reconstruction results. To evaluate the generation task, we use popular metrics as the minimum matching distance (MMD), coverage score (COV), Jensen–Shannon divergence (JSD) [25] and 1-NN classifier accuracy (1-NNA) [18]. For a set of generated shapes, MMD computes the average distance between each point cloud in the test set and its nearest neighbour in the generated set. COV measures diversity by calculating the fraction of point clouds in the reference set that are the nearest neighbours of each generated set of point clouds in terms of CD or EMD. The JSD score evaluates the similarity of point distributions by computing the Jensen–Shannon divergence between the marginal point distributions of the generated and reference sets. Yang et al. [18] noticed that the above three metrics assigned a better score to generated shapes than to training shapes, indicating that these metrics are sub-optimal. In contrast, 1-NNA does not have this problem, as it was specifically designed to distinguish between generated and reference samples [38]. If the generated and reference samples have the same distribution, 1-NNA should converge to 50% given enough samples. The closer the accuracy is to 50%, the better the model is at learning the target distribution.

4.1.3 | Implementation Details

In our experiments, the latent code is initialised as a zero vector for a neutral optimisation start. We use the Adam optimiser with separate learning rate schedules: the decoder starts at 0.002, linearly decaying to 0.00005 over 500 steps after 1200 steps, while the latent code starts at 0.001, decaying to 0.0001 under the same schedule. Inspired by DPM, our autodecoder employs a U-Net-based denoising network to capture multi-scale features. The diffusion process includes 200 denoising steps, balancing efficiency and quality, with variance ranging from 0.0001 to 0.05 for stable training and high-quality reconstruction.

4.2 | Point Cloud Reconstruction

We compared state-of-the-art autoencoder architectures with our method on point cloud reconstruction tasks, including AtlasNet, PointFlow, ShapeGF, and DPM.

For the autodecoder, we tested three types of decoders, based on AtlasNet, ShapeGF, and DPM. Additionally, we also used the theoretical lower bound 'oracle' of [15] which measures the distance between two different point cloud samples from the same shape. As Table 1 shows, our method achieves significantly lower CD and EMD than other methods across the datasets and is already close to the theoretical lower bound.

Following previous work [15, 16], we sample 4096 points for each shape and divide them equally into two different point clouds, one as the reconstruction input and the other as the reference shape for evaluation. This setting avoids the reconstruction accuracy being overly fitted to the input points and

TABLE 1 | Comparison of point cloud reconstruction performance. CD is multiplied by 10^4 and EMD is multiplied by 10^2. Bold figures indicate optima per dataset-and-metric.

Dataset	Metric	AtlasNet (S1)	AtlasNet (P25)	PointFlow	ShapeGF	DPM	PAD + ShapeGF	PAD + AtlasNet	PAD + DPM	Oracle
Aeroplane	CD	2.000	1.795	2.420	2.102	2.118	1.893	1.781	**1.754**	1.016
	EMD	4.311	4.366	3.311	3.508	2.876	3.179	4.067	**2.556**	2.141
Car	CD	6.906	6.503	5.828	5.468	5.493	5.052	4.938	**4.904**	3.917
	EMD	5.617	5.408	4.390	4.489	3.937	4.004	4.791	**3.616**	3.246
Chair	CD	5.479	4.980	6.795	5.146	5.677	4.029	4.896	**3.880**	3.221
	EMD	5.550	5.282	5.008	4.784	4.153	4.100	10.359	**3.619**	3.281
ShapeNet	CD	5.873	5.420	7.550	5.725	5.252	4.396	5.840	**3.880**	3.074
	EMD	5.457	5.599	5.172	5.049	3.783	4.057	9.166	**3.527**	3.112

results in reconstruction metrics lower than those provided by the oracle. If we use the same point cloud as both input and reference shape, the EMD accuracy in reconstruction will generally be lower than the lower bound. Figure 1 compares the reconstruction results with previous methods. We see that our method generates a reconstructed point cloud that better captures the features of the original point cloud, resulting in more realistic details (such as the chair backs and the aeroplane tails). We show more reconstruction results on other shape categories of the ShapeNet dataset in Figure 3. These results are generated by an autodecoder trained on *all 55 categories* of ShapeNet.

Although our method demonstrates superior precision in reconstructing certain fine-grained details, it may encounter

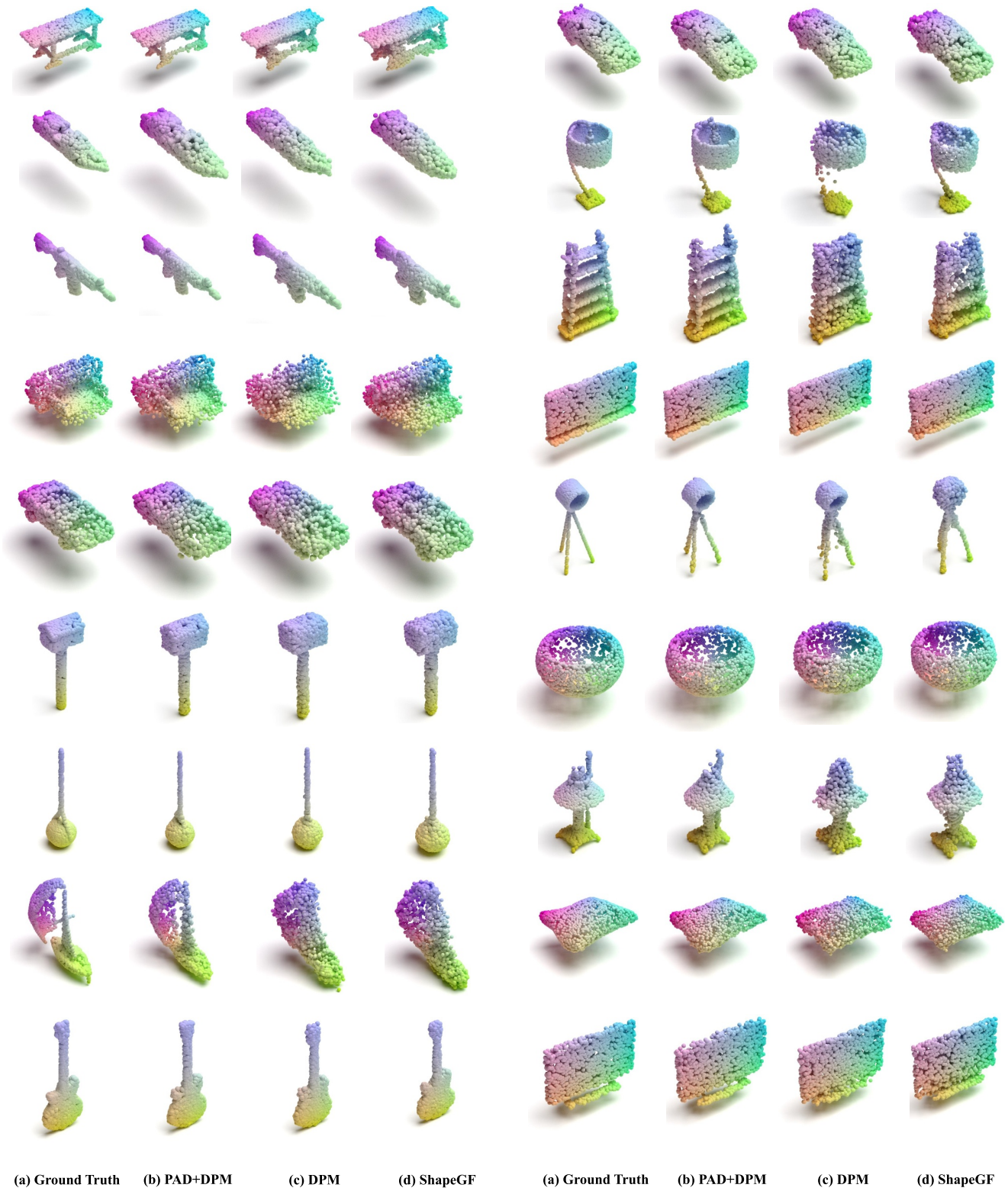


FIGURE 3 | More point cloud reconstruction results generated by PAD with DPM on the ShapeNet dataset.

limitations when handling highly complex models or regions with extremely fine and concentrated details. We attribute this to the inherent challenges of optimising local geometric features under a global Chamfer Distance (CD) loss, which may not sufficiently prioritise localised structural accuracy. To provide a comprehensive analysis, we illustrate several failure cases in Figure 4, highlighting scenarios where the method struggles to capture intricate local details.

Our autodecoder uses iterative updates during encoding. This inevitably consumes more time than an autoencoder. For example, on an RTX3090 GPU, with a batch size of 128, DPM-based PAD takes about 15.8 s to reconstruct one batch of shapes (1000 iterations) while DPM takes 4.1 s. Note, however, that the extra time comes from the *encoding* stage. There is almost no performance gap in downstream tasks that only use the decoder such as point cloud editing and generation. Also, since we do not need to train an encoder, our method improves training speed—we need approx. 0.06 s per iteration in contrast to 0.19 s for DPM. In terms of computational resources, our autodecoder omits encoder architectures such as PointNet or PointNet++, but we need to maintain a latent code for each input, so it is similar to previous encoder methods in terms of computational resources.

4.3 | Point Cloud Generation

We compare several state-of-the-art point cloud generation models, including PC-GAN [25], GCNGAN [39], TreeGAN [40], PointFlow [18], ShapeGF [16] and DPM [15]. Following ShapeGF and DPM, we normalise the generated point clouds and reference point clouds to the bounding box of $[-1, 1]^3$ to emphasise the features of shapes rather than the scale. Table 2 shows our quantitative evaluation results. PAD with the DPM-based decoder exceeds previous methods in some metrics, especially in 1-NN scores on two datasets. Additionally, we also visualise the qualitative results to demonstrate the advantages of our method in generating shapes. Our generated results typically exhibit smoother surfaces and finer details as shown in Figure 5.

4.4 | Point Cloud Upsampling

We compare our autodecoder based on DPM with two autoencoders (ShapeGF and DPM) in the task of upsampling. Both the autoencoder and the autodecoder are pre-trained on the training dataset and then used to encode-and-upsample the test samples. We randomly sample 128 points from the point cloud shapes of the ShapeNet dataset as inputs, including chairs and aeroplanes, and generate upsampling results of 2048 points. Table 3 shows the quantitative evaluation results, which include the CD and EMD values between the ground truth shapes and the upsampled shapes generated by different methods. To show that the improvement of upsampling performance is due to the better shape-awareness of PAD, rather than the effect of the iterative encoding method, we also test the upsampling results using iterative encoding on a pre-trained autoencoder architecture.

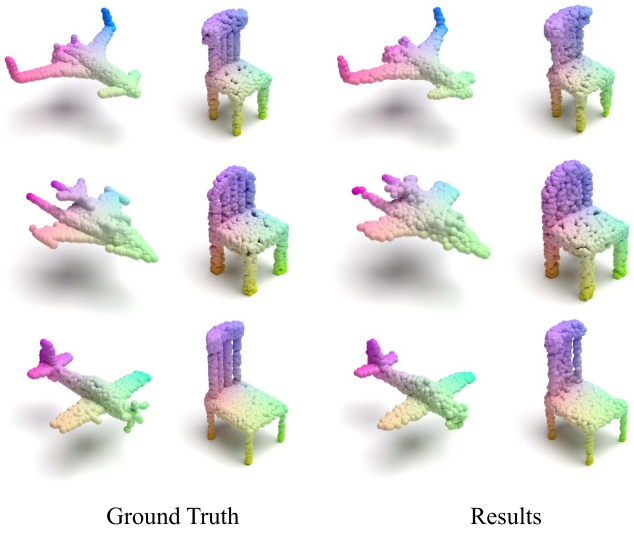


FIGURE 4 | Failed results generated by PAD + DPM on aeroplane and chair datasets.

The results show that although iterative encoding improves upsampling performance (as shown by DPM + iter. and ShapeGF + iter. in Table 3), it still falls behind our PAD method. Figure 6 shows the results of these methods. We see that our method generates upsampling results that are clearly closer to the real shape. These results show that the autodecoder can better leverage prior information and thus achieve a better surface reconstruction from low-resolution point clouds. This holds even in extreme cases when the output resolution is about 200 times larger than the input one, see Figure 7.

4.5 | Unsupervised Representation Learning

In our method, the effectiveness of unsupervised classification can measure the semantic information obtained by the latent. If we can achieve better unsupervised classification results while obtaining better reconstruction performance, this means that the learnt latent space preserves not only more shape details but also more semantic information. Similar to previous work [15, 16, 18], we train an autodecoder on all subcategories of ShapeNet. Using the pre-trained decoder, we obtained the encoding of ModelNet40 and ModelNet10. We used the encoding of the training-set shapes to train a linear SVM (support vector machine) classifier and tested it on the encoding of the test set shapes.

Since our method needs to define a one-to-one correspondence between the shape and the representation, input point clouds cannot directly benefit from rotation augmentation, unlike in existing methods [15, 16, 25]. As such, for a fair comparison, we tested the classification accuracy of ShapeGF and DPM without data augmentation. Even without data augmentation, the results in Table 4 show that our method still outperforms the *augmented* methods on ModelNet40. On ModelNet10, our autodecoder also achieves comparable results to previous methods.

TABLE 2 | Comparison of point cloud generation performance. MMD-CD and JSD figures are multiplied by 10^3; MMD-EMD is multiplied by 10^1. The arrow ↓ means the lower the better; ↑ means the higher the better. Bold figures indicate optima per dataset-and-metric.

		MMD (↓)		COV (%, ↑)		1-NNA (%, ↓)		JSD (↓)
Shape	**Model**	**CD**	**EMD**	**CD**	**EMD**	**CD**	**EMD**	—
Aeroplane	PC-GAN	3.819	1.810	42.17	13.84	77.59	98.52	6.188
	GCN-GAN	4.713	1.650	39.04	18.62	89.13	98.60	6.669
	TreeGAN	4.323	1.953	39.37	8.40	83.86	99.67	15.646
	PointFlow	3.688	1.090	44.98	44.65	66.39	69.36	1.536
	ShapeGF	3.306	1.027	50.41	47.12	**61.94**	70.51	**1.059**
	DPM	3.276	1.061	48.71	45.47	64.83	75.12	1.067
	PVD	3.713	1.086	46.77	48.42	75.32	70.07	1.291
	PointTrans-G	**3.126**	1.073	**51.02**	**48.58**	75.21	75.40	1.062
	PAD + AtlasNet	4.059	1.091	38.72	34.10	75.65	81.96	3.762
	PAD + ShapeGF	3.948	1.139	37.23	34.43	81.96	86.16	4.385
	PAD + DPM	3.609	**0.994**	44.81	46.29	62.77	**57.50**	1.539
Chair	PC-GAN	13.436	3.104	46.23	22.14	69.67	100.00	6.649
	GCN-GAN	15.354	2.213	39.84	35.09	77.86	95.80	21.708
	TreeGAN	14.936	3.613	38.02	6.77	74.92	100.00	13.282
	PointFlow	13.631	1.856	41.86	43.38	66.13	68.40	12.474
	ShapeGF	13.175	1.785	48.53	46.71	**56.17**	62.69	5.996
	DPM	**12.276**	1.784	**48.94**	**47.52**	60.11	69.06	7.797
	PVD	13.073	1.763	47.19	47.13	58.44	60.02	8.472
	PointTrans-G	13.247	1.774	47.17	46.37	58.55	60.74	8.837
	PAD + AtlasNet	13.758	3.224	44.08	32.24	61.88	100.00	12.500
	PAD + ShapeGF	14.696	1.862	43.78	44.19	68.00	70.88	15.810
	PAD + DPM	13.328	**1.728**	43.98	44.39	60.40	**59.71**	**7.727**

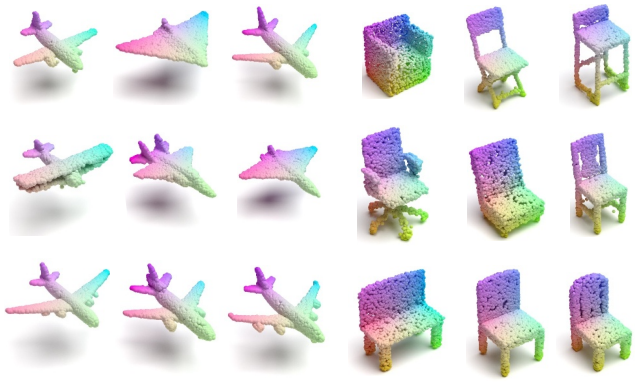
It is worth mentioning that if we train an autodecoder from scratch on the ModelNet40 dataset, and use the encodings of the training shapes and their corresponding class labels as inputs to train a classifier module consisting of two linear layers and one activation function (similar to the supervised point cloud classification without augmentation), then the classification accuracy on the test set can reach 92.0%, which is close to popular point cloud classification networks [2, 5], especially considering the fact that our decoder did not use any classification data for training.

4.6 | Ablation Study

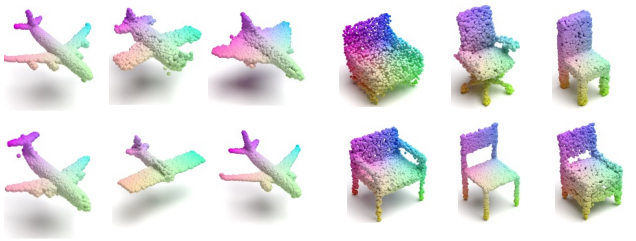
Given the limited number of hyperparameters in our model, we conduct an ablation study focussing on the dimension of the latent code. We evaluate the model's performance across different latent code dimensions by comparing the number of trainable parameters, as well as the Chamfer Distance (CD) and Earth Mover's Distance (EMD) metrics. Here, the parameter count refers to the total trainable parameters during model training, while CD and EMD represent the average metrics computed over the entire ShapeNet dataset. For a comprehensive comparison, detailed results are provided in Table 5.

5 | Conclusion

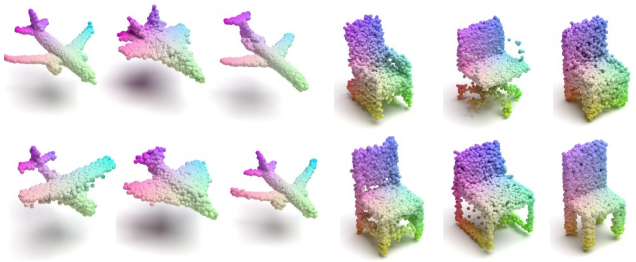
We have presented PAD, a point cloud reconstruction framework based on an autodecoder architecture without any encoder. Our experimental results demonstrate that the proposed method significantly outperforms previous state-of-the-art methods, achieving an average reduction of 24.62% in Chamfer Distance on the ShapeNet dataset, which underscores its superior reconstruction accuracy. Furthermore, PAD achieves a 68.42% reduction in computation time for downstream tasks, demonstrating its ability to enhance both reconstruction quality and efficiency. Additionally, we have validated the effectiveness of PAD by constructing a generative model and testing an unsupervised classification method on the autodecoder. Our PAD method can be directly applied to existing point cloud decoders to improve their performance and training speed. PAD is scalable and capable of preserving detailed shapes which is expected to aid downstream tasks such as point cloud editing and completion. Despite these



(a) PAD + DPM



(b) ShapeGF



(c) DPM

FIGURE 5 | Point cloud generation results generated by PAD, DPM and ShapeGF on chair and aeroplane datasets.

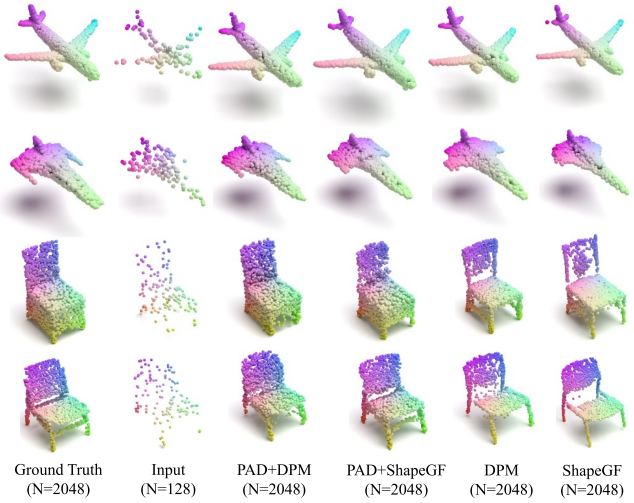


FIGURE 6 | Point cloud upsampling results generated by PAD, DPM, and ShapeGF on chair and aeroplane datasets. Pre-trained networks encode point clouds with 128 points to obtain latent codes, and then reconstruct them into point clouds with 2048 points. The upsampling results of PAD are closer to the ground truth than those of the autoencoder methods. The latter fails to capture the complete shape features—see for instance the missing spaceship wing-tips and holes in the chair backpiece for the results of DPM and ShapeGF.

TABLE 3 | Comparison of upsampling accuracy among DPM, ShapeGF, and PAD. *DPM + iter.* and *ShapeGF + iter.* denote using PAD's iterative method to find encodings in the pre-trained autoencoder architectures.

Shape Decoder	Aeroplane		Chair	
	CD	EMD	CD	EMD
ShapeGF	7.022	5.748	16.666	8.738
DPM	4.808	4.771	12.979	6.726
ShapeGF + iter.	4.447	4.923	8.804	6.312
DPM + iter.	4.406	4.324	12.315	6.343
PAD + ShapeGF	**3.101**	4.455	**8.62**	6.029
PAD + DPM	3.153	**3.467**	9.196	**5.055**

Note: The bold values indicate the best performance.

advantages, our method has two primary limitations. First, it exhibits weak generalisation capability because it requires training individual latent codes for each model, which compromises its performance on unseen shapes. Second, the inference speed is relatively slow due to the necessity of optimising latent codes during the testing phase.

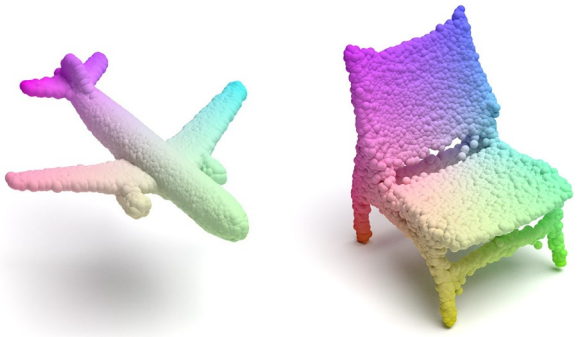


FIGURE 7 | Point cloud upsampling results generated by PAD + DPM. Pre-trained networks encode a 128-point cloud and then reconstruct it into a 20K-point cloud.

There are many avenues for future research in point cloud reconstruction and generation. First, the autodecoder is not constrained by the encoder architecture, it is possible to explore new encoding formats other than one-dimensional vectors to better represent point cloud information. Moreover, finding how to use the latent space of the autodecoder to improve encoder performance is another promising research avenue. Finally, integrating other cross-modal information as input conditions such as text, images, etc. to control the generation of point cloud is a recent research hotspot. We believe that combining our PAD with the cross-model conditioned point cloud generation can effectively improve the generation quality from many aspects.

TABLE 4 | Unsupervised representation learning. Models are pre-trained on ShapeNet and then evaluated on ModelNet40 and ModelNet10 by comparing the accuracy of linear SVMs trained using only the encodings of the training dataset. The +/− aug. entries indicate whether the random rotation augmented strategy is used in the pre-training process (+) or not (−).

	Model	ModelNet10	ModelNet40
+ aug.	AtlasNet	91.9	86.6
	PC-GAN (CD)	**95.4**	84.5
	PC-GAN (EMD)	**95.4**	84.0
	PointFlow	93.7	86.8
	ShapeGF	90.2	84.6
	DPM	94.2	87.6
− aug.	ShapeGF	90.1	83.2
	DPM	91.7	84.8
	Ours	92.6	**88.8**

Note: The bold values indicate the best performance.

TABLE 5 | Quantitative metrics of different latent dimension.

Latent dimension	Params	CD	EMD
128	672.28K	4.324	3.613
256	1000.72K	4.134	3.596
512	1657.62K	3.880	3.527
1024	2971.42K	3.868	3.523

Author Contributions

Yakai Zhang: methodology, software, validation, writing – original draft. **Ping Yang**: investigation, methodology, validation, writing – original draft. **Haoran Wang**: software, validation, data curation, methodology, visualization, writing – original draft. **Zizhao Wu**: conceptualization, methodology, software, supervision, writing – review and editing. **Xiaoling Gu**: methodology, resources, software. **Alexandru Telea**: methodology, writing – review and editing. **Jiri Kosinka**: methodology, writing – review and editing.

Acknowledgements

This work was partially supported by the Shaoxing Science and Technology Project (Grant 2023B41007).

Conflicts of Interest

The authors declare no conflicts of interest.

Data Availability Statement

The data that support the findings of this work are openly available at: https://shapenet.org/ and https://modelnet.cs.princeton.edu/.

References

1. C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation," in *Proc. IEEE CVPR* (2017), 77–85.

2. C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space," in *NIPS* (2017), 5099–5108.

3. H. Zhao, L. Jiang, J. Jia, P. H. S. Torr, and V. Koltun, "Point Transformer," in *2021 IEEE/CVF International Conference on Computer Vision, ICCV 2021, Montreal, QC, Canada, October 10-17, 2021* (IEEE, 2021), 16239–16248.

4. M. Guo, J. Cai, Z. Liu, T. Mu, R. R. Martin, and S. Hu, "PCT: Point Cloud Transformer," *Computational Visual Media* 7, no. 2 (2021): 187–199, https://doi.org/10.1007/s41095-021-0229-5.

5. Y. Wang, Y. Sun, Z. Liu, S. E. Sarma, M. M. Bronstein, and J. M. Solomon, "Dynamic Graph CNN for Learning on Point Clouds," *ACM Transactions on Graphics* 38, no. 5 (2019): 146:1–146:12, https://doi.org/10.1145/3326362.

6. Z. Fang, B. Xiong, and F. Liu, "Sparse Point-Voxel Aggregation Network for Efficient Point Cloud Semantic Segmentation," *IET Computer Vision* 16, no. 7 (2022): 644–654, https://doi.org/10.1049/cvi2.12131.

7. Y. Guo, H. Wang, R. Clark, S. Berretti, and M. Bennamoun, "Deep Learning for 3D Vision," *IET Computer Vision* 16, no. 7 (2022): 567–569, https://doi.org/10.1049/cvi2.12141.

8. C. Zhang, H. Wan, X. Shen, and Z. Wu, "Patchformer: An Efficient Point Transformer With Patch Attention," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR* (IEEE, 2022), 11789–11798.

9. X. Deng, C. Zhang, J. Shi, and Z. Wu, "PU-GAT: Point Cloud Upsampling With Graph Attention Network," *Graphical Models* 130 (2023): 101201, https://doi.org/10.1016/j.gmod.2023.101201.

10. C. Zhang, H. Chen, H. Wan, P. Yang, and Z. Wu, "GRAPH-PBN: Graph-Based Parallel Branch Network for Efficient Point Cloud Learning," *Graphical Models* 119 (2022): 101120, https://doi.org/10.1016/j.gmod.2021.101120.

11. C. Zhang, H. Wan, X. Shen, and Z. Wu, "PVT: Point-Voxel Transformer for Point Cloud Learning," *International Journal of Intelligent Systems* 37, no. 12 (2022): 11985–12008, https://doi.org/10.1002/int.23073.

12. Z. Liu, H. Tang, Y. Lin, and S. Han, "Point-Voxel CNN for Efficient 3D Deep Learning," in *NeurIPS* (2019), 963–973.

13. Y. Yang, C. Feng, Y. Shen, and D. Tian, "Foldingnet: Point Cloud Auto-Encoder via Deep Grid Deformation," in *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018* (Computer Vision Foundation/IEEE Computer Society, 2018), 206–215.

14. T. Groueix, M. Fisher, V. G. Kim, B. Russell, and M. Aubry, "AtlasNet: A Papier-Mâché Approach to Learning 3D Surface Generation," in *Proc. IEEE CVPR* (2018).

15. S. Luo and W. Hu, "Diffusion Probabilistic Models for 3D Point Cloud Generation," in *CVPR* (Computer Vision Foundation/IEEE, 2021), 2837–2845.

16. R. Cai, G. Yang, H. Averbuch-Elor, et al., "Learning Gradient Fields for Shape Generation," in *ECCV, Volume 12348 of Lecture Notes in Computer Science* (Springer, 2020), 364–381.

17. L. Zhou, Y. Du, and J. Wu, "3D Shape Generation and Completion through Point-Voxel Diffusion," in *Proc. IEEE ICCV* (2021), 5806–5815.

18. G. Yang, X. Huang, Z. Hao, M. Liu, S. J. Belongie, and B. Hariharan, "PointFlow: 3D Point Cloud Generation With Continuous Normalizing Flows," in *Proc. IEEE ICCV* (2019), 4540–4549.

19. R. Li, X. Li, K. Hui, and C. Fu, "SP-GAN: Sphere-Guided 3D Shape Generation and Manipulation," *ACM Transactions on Graphics* 40, no. 4 (2021): 151:1–151:12, https://doi.org/10.1145/3450626.3459766.

20. X. Yu, L. Tang, Y. Rao, T. Huang, J. Zhou, and J. Lu, "POINT-BERT: Pre-Training 3D Point Cloud Transformers With Masked Point Modeling," in *IEEE/CVF Conference on Computer Vision and Pattern*

Recognition, CVPR 2022, New Orleans, LA, USA, June 18-24, 2022 (IEEE, 2022), 19291–19300.

21. Y. Pang, W. Wang, F. E. H. Tay, W. Liu, Y. Tian, and L. Yuan, "Masked Autoencoders for Point Cloud Self-Supervised Learning," in *Computer Vision - ECCV 2022 - 17th European Conference, Tel Aviv, Israel, October 23-27, 2022, Proceedings, Part II, Volume 13662 of Lecture Notes in Computer Science* (Springer, 2022), 604–621.

22. A. Xiao, J. Huang, D. Guan, and S. Lu, "Unsupervised Representation Learning for Point Clouds: A Survey," in *CoRR abs/2202.13589* (2022).

23. K. Yin, Z. Chen, H. Huang, D. Cohen-Or, and H. Zhang, "LOGAN: Unpaired Shape Transform in Latent Overcomplete Space," *ACM Transactions on Graphics* 38, no. 6 (2019): 198:1–198:13, https://doi.org/10.1145/3355089.3356494.

24. H. Wang, J. Li, A. Telea, J. Kosinka, and Z. Wu, "USTNet: Unsupervised Shape-To-Shape Translation via Disentangled Representations," *Computer Graphics Forum* 41, no. 7 (2022): 141–152, https://doi.org/10.1111/cgf.14664.

25. P. Achlioptas, O. Diamanti, I. Mitliagkas, and L. J. Guibas, "Learning Representations and Generative Models for 3D Point Clouds," in *Proc. ICML, Volume 80 of PMLR* (2018), 40–49.

26. H. Fan, H. Su, and L. J. Guibas, "A Point Set Generation Network for 3D Object Reconstruction From a Single Image," in *Proc. IEEE CVPR* (2017), 2463–2471.

27. A. Zadeh, Y. C. Lim, P. P. Liang, and L. Morency, "Variational Auto-Decoder," in *CoRR abs/1903.00840* (2019).

28. J. J. Park, P. Florence, J. Straub, R. A. Newcombe, and S. Lovegrove, "DeepSDF: Learning Continuous Signed Distance Functions for Shape Representation," in *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019* (Computer Vision Foundation/IEEE, 2019), 165–174.

29. Y. Sun, Y. Wang, Z. Liu, J. E. Siegel, and S. E. Sarma, "Pointgrow: Autoregressively Learned Point Cloud Generation With Self-Attention," in *IEEE Winter Conference on Applications of Computer Vision, WACV 2020, Snowmass Village, CO, USA, March 1-5, 2020* (IEEE, 2020), 61–70.

30. M. Zamorski, M. Zieba, R. Nowak, W. Stokowiec, and T. Trzcinski, "Adversarial Autoencoders for Generating 3D Point Clouds," in *CoRR abs/1811.07605* (2018).

31. M. Gadelha, R. Wang, and S. Maji, "Multiresolution Tree Networks for 3D Point Cloud Processing," in *Computer Vision - ECCV 2018 - 15th European Conference, Munich, Germany, September 8-14, 2018, Proceedings, Part VII, Volume 11211 of Lecture Notes in Computer Science* (Springer, 2018), 105–122.

32. D. P. Kingma and M. Welling, "Auto-Encoding Variational Bayes," in *ICLR* (2014).

33. J. Ho, A. Jain, and P. Abbeel, "Denoising Diffusion Probabilistic Models," in *NeurIPS* (2020).

34. X. Zeng, A. Vahdat, F. Williams, et al., "LION: Latent Point Diffusion Models for 3D Shape Generation," in *CoRR abs/2210* (2022).06978

35. R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer, "High-Resolution Image Synthesis With Latent Diffusion Models," in *CVPR* (IEEE, 2022), 10674–10685.

36. A. X. Chang, T. A. Funkhouser, L. J. Guibas, et al., "ShapeNet: An Information-Rich 3D Model Repository," in *CoRR abs/1512.03012* (2015).

37. Z. Wu, S. Song, A. Khosla, et al., "3D Shapenets: A Deep Representation for Volumetric Shapes," in *CVPR* (IEEE Computer Society, 2015), 1912–1920.

38. D. Lopez-Paz and M. Oquab, "Revisiting Classifier Two-Sample Tests," in *ICLR* (OpenReview.net, 2017).

39. D. Valsesia, G. Fracastoro, and E. Magli, "Learning Localized Generative Models for 3D Point Clouds via Graph Convolution," in *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019* (OpenReview.net, 2019).

40. D. W. Shu, S. W. Park, and J. Kwon, "3D Point Cloud Generative Adversarial Network Based on Tree Structured Graph Convolutions," in *Proc. IEEE ICCV* (IEEE, 2019), 3858–3867.