Learning to Train a Point Cloud Reconstruction Network without Matching

Tianxin Huang[1], Xuemeng Yang[1], Jiangning Zhang[1,2], Jinhao Cui[1], Hao Zou[1], Jun Chen[1], Xiangrui Zhao[1], and Yong Liu[1,3 ⋆]

[1] APRIL Lab, Zhejiang University, HangZhou, China
[2] Tencent Youtu Laboratory
[3] Huzhou Institute of Zhejiang University
yongliu@iipc.zju.edu.cn

Abstract. Reconstruction networks for well-ordered data such as 2D images and 1D continuous signals are easy to optimize through element-wised squared errors, while permutation-arbitrary point clouds cannot be constrained directly because their points permutations are not fixed. Though existing works design algorithms to match two point clouds and evaluate shape errors based on matched results, they are limited by pre-defined matching processes. In this work, we propose a novel framework named PCLossNet which learns to train a point cloud reconstruction network without any matching. By training through an adversarial process together with the reconstruction network, PCLossNet can better explore the differences between point clouds and create more precise reconstruction results. Experiments on multiple datasets prove the superiority of our method, where PCLossNet can help networks achieve much lower reconstruction errors and extract more representative features, with about 4 times faster training efficiency than the commonly-used EMD loss. Our codes can be found in https://github.com/Tianxinhuang/PCLossNet.

Keywords: Learning to train; no matching; point cloud reconstruction;

1 Introduction

To reconstruct a series of data, a network such as auto-encoder is usually adopted to predict an output as similar to the original data as possible, which is usually trained with reconstruction errors between original data and the network output. Reconstruction errors for 2D images or 1D signals are quite easy to calculate directly with element-wised mean squared errors (MSE) because their elements such as pixels are arranged in a certain order. However, matching algorithms are required to synchronize different data when calculating reconstruction errors for point clouds because permutations of input and output point sets in reconstruction networks may be different.

Different matching algorithms match points between point clouds according to different rules. Fig. 1-(a) and (b) illustrate the matching processes adopted by two commonly used structural losses: the Chamfer Distance (CD) and the Earth Mover's Distance (EMD) [4]. CD matches points in one point set with their nearest neighbors in

⋆ indicates the corresponding author.

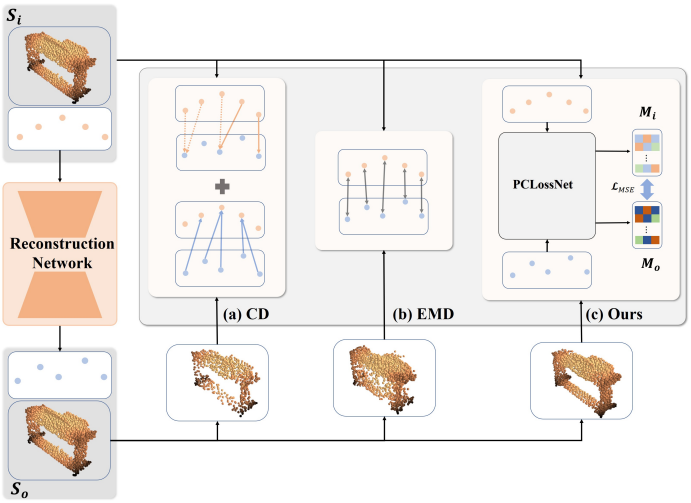

Fig. 1. (a) and (b) denote two matching processes of commonly-used CD and EMD losses, while (c) illustrates the PCLossNet framework.

the other set, while EMD optimizes to find a points bijection with the approximated minimum matching distance between point clouds. Many works [5,8,19,31,29,13] aims at improving the network structures to learn more representative features or construct better shapes, while all of them use the Chamfer Distance (CD) and the Earth Mover's Distance (EMD) as basic shape constraints. However, matching processes adopted by either CD or EMD are actually approximations for the shape differences. Converging well under matching may not mean the point cloud shapes are totally the same. Inevitable shape defects may exist due to biases between the pre-defined matching process and true shape differences. As illustrates in Fig. 1-(a) and (b), CD may create non-uniform surfaces because the its matching pays attention to the average neighbor distances, which allows one point to be neighbors of multiple points of another point set and lacks constraints for uniformity. Though EMD constructs bijection to create more uniform reconstructed results, the optimization processes to approximate the minimum matching distance may have biases and create distortions sometimes. Though some works introduce discriminator networks on point clouds [13,22,11] to enhance details, they still use CD or EMD to constrain basic structures of point clouds and are limited by the matching processes. In this condition, we want to get rid of the limitations of matching processes by replacing them with a differentiable network structure, in which way we can learn to train a reconstruction network without matching. In this work, we propose a novel struture named Point Cloud reconstruction Loss Network (PCLossNet) to train the reconstruction network without matching. As illustrated in Fig. 1-(c), PCLossNet extracts comparison matrices M_i and M_o from point clouds and evaluate their shape differences with distances between comparison matrices. To train the networks, parameters of the reconstruction network and PCLossNet are updated by turns

in a generative-adversarial process. Intuitively speaking, in each iteration, PCLossNet explores the regions with larger reconstruction errors by maximizing distances between M_i and M_o, while the reconstruction network is optimized by minimizing the distances. The reconstruction network and PCLossNet promote each other to stimulate the potential of networks. Note that our work is different from existing GAN-based discriminator constraints because we design PCLossNet not only to improve reconstruction performances simply, but also to replace the pre-defined matching process. More relative discussions can be found in Sec. 3. Our contributions can be summarized as follows:

- We propose a new differentiable structure named PCLossNet to transform shape differences between point clouds to errors between extracted comparison matrices.
- By training with a generative adversarial process, PCLossNet can dynamically search for shape differences between point clouds and constrain the reconstruction network without any pre-defined matching process;
- Experiments on multiple datasets demonstrate that networks trained with PCLossNet can achieve better reconstruction performances and extract more representative features with higher classification accuracy.

2 Related Works

2.1 Optimization-based Matching Losses

Point clouds are permutation-invariant, which means the data is irrelevant with the order of single points. Mean Square Error (MSE) commonly used in the reconstruction of 2D images or 1D signals is not appropriate for point clouds because the order of reconstructed point clouds may be quite different with the ground truths. Since the raise of PointNet [17] and PointNet++ [18], more and more works [19,31,14,9] have been developed to improve the reconstruction performances on point clouds. All of them are trained based on the Chamfer Distance or the Earth Mover's Distance [4]. The Chamfer Distance (CD) is defined as

$$\mathcal{L}_{CD}(S_1, S_2) = \frac{1}{2}\left(\frac{1}{|S_1|}\sum_{x\in S_1}\min_{y\in S_2}\|x-y\|_2 + \frac{1}{|S_2|}\sum_{x\in S_2}\min_{y\in S_1}\|x-y\|_2\right), \quad (1)$$

where S_1 and S_2 are two point sets. CD is actually the average distance from points in one set to their nearest neighbors in another set. A same nearest neighbor is allowed for multiple points for the calculation of CD. With the matching by nearest neighbors, CD concentrates on the differences between point clouds contours. But it usually constructs non-uniform surfaces because it lacks constraints for local uniformity.

The Earth Mover's Distance (EMD) can be presented as

$$\mathcal{L}_{EMD}(S_1, S_2) = \min_{\phi: S_1 \to S_2}\frac{1}{|S_1|}\sum_{x\in S_1}\|x-\phi(x)\|_2, \quad (2)$$

where S_1 and S_2 are two point sets. EMD aims to find an one-to-one optimal bijection ϕ from one point set to another by optimizing the minimum matching distances

between point sets. An optimization process is needed to construct the bijection in each iteration. In practice, exact computation of EMD is too expensive for deep learning, even on graphics hardware. In this work, we follow [4] to conduct a $(1+\epsilon)$ approximation scheme for EMD. The algorithm is easily parallelizable on GPU. EMD can create more uniform shapes by constructing bijection, while the optimized matching may cause distortions. Besides, EMD can only be applied to reconstructed output with the same number of points as input to solve the bijection, which limits its application. Some recent works [16,25] try different methods to improve CD or EMD, while their performances are still limited by the matching algorithms. Though DPdist [20] is a fully-network based training loss without any matching, it is proposed for registration instead of reconstruction, which is also inflexible due to the requirements of appropriate pre-training process.

2.2 Generative Adversarial Network

The Generative adversarial network [6] is an appropriate framework to train a network with another network. They are created to learn a transformation from a prior distribution such as the Gaussian distribution to the distribution of ground truths. In this way, sampling from the prior distribution can construct new data. A discriminator is adopted to judge if the generated data satisfy the same distribution as ground truths. The training strategy for a basic GAN can be presented as

$$\min_G \max_D V(G,D) = \mathbb{E}_{x\sim p_{data}(x)}[logD(x)] + \mathbb{E}_{z\sim p_z(z)}[log(1-D(G(z)))], \quad (3)$$

where p_{data} and p_z are the distributions of real data and input noise variables, respectively. G is a generation network to transform sampled noise to fake but realistic generated data, while D is a discriminator network to distinguish generated data and real data. Many GANs [3,7,15] have been developed based on the basic generative adversarial framework. Discriminator proposed in GANs [6] is sometimes introduced to improve the detail preserved ability on reconstruction-related tasks [22,11,21] of point clouds. Nevertheless, existing works only use the discriminator as a supplement for structural losses CD or EMD to improve the reconstruction performances. They are still influenced by the pre-defined matching processes.

3 Methodology

In this section, we introduce the core ideas of our method. The whole pipeline of our algorithm is shown in Fig. 2. We propose a framework named PCLossNet to extract comparison matrices M_i and M_o from point clouds S_i and S_o and evaluate their differences by errors between corresponding comparison matrices. Details of PCLossNet are presented in sec 3.1. It is trained together with the reconstruction network in a generative-adversarial process. The training process is further demonstrated in Alg. 1. Besides, we also conduct a simple theoretical analysis of our method in sec 3.3.

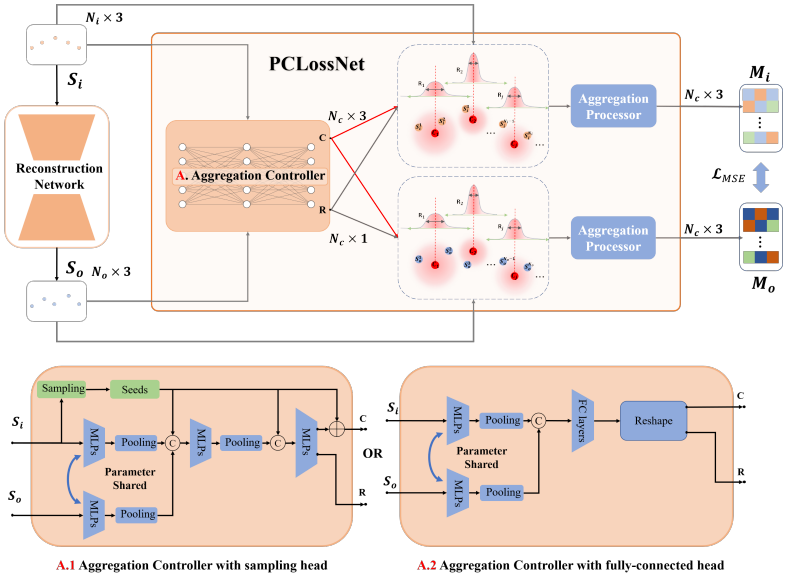

Fig. 2. The whole pipeline to train the reconstruction network with PCLossNet which transforms point clouds differences to the errors between comparison matrices M_i and M_o. N_i and N_o are the points numbers of point clouds S_i and S_o, while N_c denote the number of distributions to extract M_i and M_o, $j \in \{1, 2, ..., N_c\}$. A.Aggregation Controller and Aggregation processor are components of PCLossNet, while A.1 and A.2 are two different implements of A.Aggregation Controller. ⓒ and ⊕ denote the concatenation and element-wise addition, respectively.

3.1 The Architecture of PCLossNet

As illustrated in Fig. 2, PCLossNet plays the important role of extracting comparison matrices from point clouds. Existing point-based discriminators such as [13,22] predict scores to evaluate the similarities from point clouds with deep neural networks, which are totally non-linear structures. By running an adversarial training, the discriminators are expected to evaluate the shapes differences with the scores. However, same score may come from different outputs because the mapping from point clouds to scores is totally non-linear with unlimited searching spaces. Therefore, all existing point-based discriminators need a matching process to constrain reconstructed point clouds to similar shapes with original point clouds, which can reduce the searching spaces of discriminators and avoid the ambiguity of predicted scores as much as possible. They are actually limited by the biases between matching losses and true shapes differences.

In this condition, we decouple the non-linear discriminator structures into non-linear Aggregation Controller(AC) module to preserve the adversarial ability, and Aggregation processor (AP) module extracting comparison matrices M_i and M_o totally based on 3D Euclidean Space to naturally limit the searching spaces of comparison matrices. AP module extracts comparison codes M_i and M_o by weighting points with multiple distributions, while the centers and widths of these distributions are controlled with ag-

gregation centers C and decay radii R predicted by AC module with MLPs from S_i and S_o. The number of weighting distributions is defined as N_c in this work. During the training process, C and R are dynamically adjusted to search for differences between S_i and S_o. The operations to aggregate points/features by weights in PCLossNet are similar as those in NetVLAD [2], while they have obvious differences on the specific network structures. NetVLAD uses a separate group of parameters for the aggregation around each clustering center, while PCLossNet uses a group of parameters in AC module to adjust aggregations around all centers. Besides, NetVLAD is trained end-to-end by loss, while PCLossNet is adversarially optimized to search for shape differences. More discussions about AC module and AP module are presented below.

Aggregation Controller. In the Aggregation Controller (AC) module, features from input S_i and reconstructed S_o are extracted with parameter-shared Multi Layer Perceptrons (MLPs) and symmetric pooling operations. To search for the differences between point clouds, we introduce two implements of AC modules: AC module with the SAmpling head (**ACSA**) and AC module with the Fully Connected head (**ACFC**) to C and R controlling the positions and widths of distributions to extract comparison matrices. Structures of ACSA and ACFC are presented in Fig. 2-A.1 and A.2.

Let input be S_i, output be S_o, sampling operation be $sam(\cdot)$, the combination of MLPs and pooling operation be $f(\cdot)$, the concatenation operation be $g(\cdot)$.

We can present the ACSA as

$$\begin{cases} seeds = sam(S_i), \\ C = seeds + MLPs(g(f(g(f(S_i), f(S_o), seeds)), seeds)), \\ R = MLPs(g(f(g(f(S_i), f(S_o), seeds)), seeds)), \end{cases} \tag{4}$$

while the ACFC can be described as

$$C, R = FCs(g(f(S_i), f(S_o))), \tag{5}$$

where C and R are the predicted aggregation centers and corresponding decay radii, respectively. $FCs(\cdot)$ and $MLPs(\cdot)$ denote the fully-connected layers and MLPs, respectively. Note that we provide two simple organized networks ACSA and ACFC as examples of AC modules in this work. More effective networks can be designed to futher improve the performance of PCLossNet.

Aggregation processor. Aggregation processor (AP) module aggregates comparison matrices from point clouds on 3D Euclidean Space with multiple distributions controlled by aggregation centers C and decay radii R from AC module. Note that there is not any network structure in AP module. With provided centers C, decay radii R, and point cloud S, we can define AP module as follows.

(1) We build the graph $G = (V, E)$, where

$$V = C \cup S, E = \{< c, s > | \forall c \in C, \forall s \in S\}, \tag{6}$$

when $< c, s >= s - c$ exists for point clouds without direct edge information.

(2) Then we can aggregate the comparison matrices M from G by the distributions decided by aggregation centers C and decay radii R as

$$M = \{m | m = \sum_{k=1}^{N} \frac{exp(-\frac{\|e_k^j\|_2}{R_j + \sigma})}{\sum_{k=1}^{N} exp(-\frac{\|e_k^j\|_2}{R_j + \sigma}) + \delta} \cdot e_k^j, j \in \{1 \cdots N_c\}\}, \tag{7}$$

Algorithm 1 Training Process

Input: Dataset $\mathbf{S}_i$, the number of iterations $iter$, the reconstruction network $RecNet(\cdot)$
for $n = 1$ **to** $iter$ **do**
  Calculate output of the reconstruction network:
  $\mathbf{S}_o^n = RecNet(\mathbf{S}_i^n)$.
  Fix the parameter of reconstruction network and update PCLossNet by descending gradient:
  $\nabla_{\theta_L} L_{PCLossNet}(\mathbf{S}_o^n, \mathbf{S}_i^n)$.
  Fix PCLossNet and update the reconstruction network by descending gradient:
  $\nabla_{\theta_T} T_{PCLossNet}(\mathbf{S}_o^n, \mathbf{S}_i^n)$.
end for

where $e_k^j =< c_j, s^k >$ and $e_k^j \in E$. N is the points number of S, while N_c is the number of distributions aggregating comparison matrices, also the number of centers in C. σ is a tiny constant to ensure the equation still works when $R_j \to 0$. δ is a tiny constant to protect weights from going out of bounds when the denominator is small.

To ensures each point gets enough weight from distributions, we use the uniformity constraint to shorten distances between points and their nearest aggregation centers as

$$L_{UC} = \underset{\forall s \in S}{mean} \underset{<c,s> \in E}{min} \|c - s\|_2. \tag{8}$$

Besides, we restrain the decay radius R to reduce the decay radii around each center as

$$L_R = \|R\|_2^2. \tag{9}$$

3.2 Training of the Reconstruction Network

To train a point cloud reconstruction network, we update parameters of the reconstruction network and PCLossNet in a generative-adversarial process. Our training algorithm is presented in Alg. 1. θ_L and θ_T are parameters of PCLossNet and the reconstruction network, respectively. Loss function for the updating of the reconstruction network is

$$T_{PCLossNet}(S_i, S_o) = \|M_i, M_o\|_2 + \epsilon * L_{UC}, \tag{10}$$

while loss function for the updating of PCLossNet is

$$L_{PCLossNet}(S_i, S_o) = -log(\|M_i, M_o\|_2 + \sigma_L) + L_{UC} + \epsilon_1 * L_R, \tag{11}$$

where M_i and M_o, L_{UC} and L_R are comparison matrices, uniformity constraint and decay radii constraint mentioned in Sec 3.1. σ_L is a tiny constant to prevent the gradient of $L_{PCLossNet}$ from explosion when $\|M_i, M_o\|_2 \to 0$. ϵ and ϵ_1 are weights for components. L_{UC} is defined in Eq. 10 to ensure each point in reconstructed point clouds get enough weights to be constrained.

Loss function for the reconstruction network is designed to reduce the error between comparison matrices, while the loss function for PCLossNet tries to explore more differences by increasing that error. We adopt $log(\cdot)$ to adjust the updating of PCLossNet

dynamically, in which case the gradient would decrease as $\|M_i, M_o\|_2$ increases. In this way, PCLossNet is updated slowly when the reconstruction network is weak, fast when the reconstruction network works well and reaches a small reconstruction error.

3.3 Algorithm Analysis

To intuitively analyze our method, the training process can be modeled as a process of solving equations. As illustrated in Sec. 3.1, reconstructed output and the ground truths are abstracted into comparison matrices through Eq. 7. Let $s_i^k \in S_i$ and $s_o^k \in S_o$ be *k-th* point in input and output, $c \in C$ and $r \in R$ be aggregation center and decay radii. Then, for the input and reconstructed point clouds in each iteration, we have

$$\begin{cases} \| \sum_{k=1}^{N_i} w_{(s_i^k, c_1, r_1)} \cdot s_i^k - \sum_{k=1}^{N_o} w_{(s_o^k, c_1, r_1)} \cdot s_o^k \|_2 = \sigma_n^1 \\ \| \sum_{k=1}^{N_i} w_{(s_i^k, c_1, r_1)} \cdot s_i^k - \sum_{k=1}^{N_o} w_{(s_o^k, c_1, r_1)} \cdot s_o^k \|_2 = \sigma_n^2 \\ \vdots \\ \| \sum_{k=1}^{N_i} w_{(s_i^k, c_{N_c}, r_{N_c})} \cdot s_i^k - \sum_{k=1}^{N_o} w_{(s_o^k, c_{N_c}, r_{N_c})} \cdot s_o^k \|_2 = \sigma_n^{N_c}, \end{cases} \tag{12}$$

where $w_{(s_i^k, c_j, r_j)} = \frac{exp(-\frac{\|s_i^k - c_j\|_2}{r_j + \sigma})}{\sum_{k=1}^{N_i} exp(-\frac{\|s_i^k - c_j\|_2}{r_j + \sigma}) + \delta}$ as Eq. 7. N_c is the number of aggregation centers, while N_i and N_o are the number of input and reconstructed points, respectively. σ_n^j is the corresponding distance between comparison matrices around *j-th* aggregation center after *n-th* iteration. We can see that Eqs. 12 is undetermined in a single iteration because we usually have $N_c < N_i$ and $N_c < N_o$ to reduce the computational cost.

In each later iteration, a new group of equations are added. For $L_{PCLossNet}$ in Eq. 11, $-log(\|M_i, M_o\|_2)$ searches for equations as independent as possible from former ones during subsequent iterations, while L_{UC} and L_R improves local independence in the group of equations. L_{UC} would like to provide a near aggregation center for each point, while L_R tends to shrink the decay radii and concentrates bigger weights on fewer points. They will lead to uniform spatial positions of aggregation centers and smaller intersections between their neighbors, which will improve the local independence in each group of equations. As a result, the set of equations will approach to be determined after multiple iterations, which can constrain all points without matching.

4 Experiments

4.1 Datasets and Implementation Details

In this work, three point cloud datasets: ShapeNet [28], ModelNet10 (MN10) and ModelNet40 (MN40) [26] are adopted. Each model consists of 2048 points randomly sampled from the surfaces of original mesh models.

We train networks on train splits of ShapeNet part dataset following FoldingNet [27] and evaluate performances on both the test split of ShapeNet and MN40 to provide a robust and exhaustive evaluation. We optimize the reconstruction network by Adam Optimizer [12] with a learning rate of 0.0001, while PCLossNet is trained with a learning rate of 0.005. We compare commonly-used matching losses Chamfer Distance (CD)

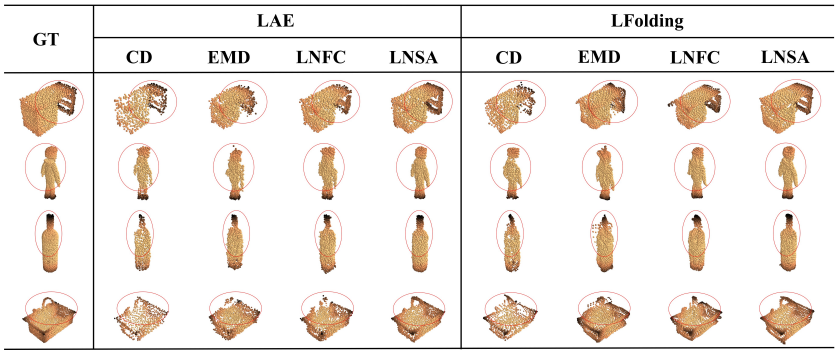

Fig. 3. Qualitative Comparison with matching losses on point clouds. We can see that our methods have great improvements on reconstruction details over CD and EMD with matching priors.

and Earth Mover's Distance (EMD) mentioned in Sec. 2.1 with two implements of PCLossNet: **PCLossNet with ACFC** (**LNFC**) and **PCLossNet with ACSA** (**LNSA**), where ACFC and ACSA are defined in Sec. 3.1.

Metrics. To provide a clear and accurate evaluation for the reconstruction performance, we adopt multi-scale Chamfer Distance (MCD) proposed by [9] and Hausdorff distance (HD) following [24] as metrics in this work. Let input point cloud be S_i, reconstructed output point cloud be S_o, MCD can be defined as

$$MCD = \xi \cdot CD(S_i, S_o) + \frac{1}{|K|} \sum_{\forall k \in K} \frac{1}{|C|} \sum_{\forall c \in C} CD(S_i^{c,k}, S_o^{c,k}), \tag{13}$$

where C denotes centers of evaluated local regions, which is acquired with farthest point sampling (FPS) [18] from S_i, S_o. $CD(\cdot)$ denotes the Chamfer Distance. K is a list including multiple k values to control the local region scales. $S_i^{c,k}$ means the local region on S_i with k points around center c. We can see that MCD evaluates both local and global reconstruction errors with Chamfer Distance, while ξ is a parameter to control their weights. Here, we have $K = \{4, 8, 16, 32, 64\}$ around 256 sampled C. HD can be defined as

$$HD = \frac{1}{2}(\max_{x \in S_i} \min_{y \in S_o} \|x - y\|_2^2 + \max_{x \in S_o} \min_{y \in S_i} \|x - y\|_2^2). \tag{14}$$

We can see that HD measures the global worst reconstruction distortions. We use MCD to comprehensively consider global and local structural differences, and HD to compare the most obvious shape distortions. In this work, all metrics are multiplied with 10^2.

Reconstruction Networks. AE [1] and FoldingNet [27] are two classic and commonly used point cloud reconstruction networks, which have been used in many works [19,10,13,30]. In this work, We apply PointNet [17], PointNet++ [18] and DGCNN [23] to the encoder parts of AE [1] and FoldingNet [27] to construct diverse reconstruction networks. We use 128-dim bottleneck layers in the encoder parts following AE [1]. Besides, to build stronger reconstruction networks, we divide point clouds into multiple

Data		ShapeNet								ModelNet40							
RecNet		AE	Folding	AE (PN++)	Folding (PN++)	AE (DGCNN)	Folding (DGCNN)	LAE	LFolding	AE	Folding	AE (PN++)	Folding (PN++)	AE (DGCNN)	Folding (DGCNN)	LAE	LFolding
CD	MCD	0.32	0.42	0.37	0.34	0.30	0.52	0.31	0.28	0.75	0.83	0.88	0.79	0.76	0.75	0.44	0.39
	HD	1.87	4.20	2.50	3.37	1.88	3.84	1.02	1.20	6.08	7.35	7.38	7.55	6.40	7.03	1.69	2.16
EMD	MCD	0.25	-	0.26	-	0.21	-	0.23	0.21	0.61	-	0.66	-	0.56	-	0.33	0.32
	HD	2.23	-	2.51	-	2.09	-	2.48	2.40	6.18	-	6.47	-	5.66	-	3.82	3.88
LNFC	MCD	0.23	**0.32**	0.25	0.33	0.21	0.69	0.15	0.15	**0.58**	0.75	0.68	0.76	**0.56**	1.04	0.23	0.22
	HD	1.66	2.71	1.98	2.84	1.65	4.26	1.27	1.22	5.43	6.80	6.32	6.94	5.35	8.08	2.05	1.98
LNSA	MCD	**0.23**	0.33	**0.24**	**0.31**	**0.20**	**0.43**	**0.13**	**0.14**	0.59	**0.75**	**0.66**	**0.74**	0.60	**0.74**	**0.17**	**0.19**
	HD	**1.66**	**2.57**	**1.87**	**2.50**	**1.51**	**3.10**	**0.65**	**0.76**	**5.30**	**6.65**	**6.04**	**6.75**	**5.30**	**6.55**	**1.05**	**1.37**

Table 1. Quantitative Comparisons with matching-based losses on reconstruction networks. AE($f(\cdot)$) or Folding($f(\cdot)$) denotes reconstruction networks replacing the encoder parts of AE or Folding with $f(\cdot)$. For example, AE(PN++) denotes AE network with PN++ encoder. **Bold** values denote the best values.

local regions following PointNet++ [18] and apply AE and FoldingNet in each region to construct Local AE (LAE) and Local FoldingNet (LFolding) networks. To make a fair evaluation, we retrain all networks under same settings with different training losses.

4.2 Comparisons with Basic Matching-based Losses

In this section, we conduct comparisons on multiple kinds of reconstruction networks trained with two basic matching-based losses CD and EMD. We do not introduce extra constraint in this part to make a comparison only between basic structural losses with matching and our method without matching.

Comparisons on reconstruction errors. The qualitative and quantitative results are presented in Fig. 3 and Table 1, respectively. We can see that our methods can achieve lowest reconstruction errors in most conditions and reconstruct models details much better than CD and EMD, which confirms that they can better constrain shape differences between point clouds. LNSA usually has better performances than LNFC because the sampling head adopted in LNSA may be easier to acquire aggregation centers around complicated shapes than the fully-connected head used in LNFC.

Besides, LNSA has greater improvements on LAE and LFolding networks, which means that it can tap into greater potential when the reconstruction network is stronger.

Comparisons on unsupervised classification. To further explore the effectiveness of PCLossNet, we also conduct a comparison on unsupervised classification following AE [1] and FoldingNet [27]. Specifically speaking, we train multiple auto-encoders with different loss methods and use the encoder parts to extract features from point clouds. Features extracted from train splits of MN10 and MN40 are adopted to train Supported Vector Machines (SVMs), whose classification accuracies are measured on test splits to evaluate the distinguishability of features. The results are presented in Table 2. We can see that our LNSA and LNFC can cover all best performed cases with higher classification accuracies, which means that PCLossNet can help the reconstruction network learn more representative features.

RecNet		AE		Folding		AE(PN++)		Folding(PN++)		AE(DGCNN)		Folding(DGCNN)	
Dataset		MN10	MN40	MN10	MN40	MN10	MN40	MN10	MN40	MN10	MN40	MN10	MN40
Methods	CD	90.60	85.92	91.03	85.22	90.38	88.03	91.48	87.01	91.04	87.95	90.81	87.18
	EMD	89.49	85.47	-	-	90.15	88.07	-	-	90.48	87.78	-	-
	LNFC	91.15	86.08	89.93	84.82	90.71	**88.19**	91.04	**87.26**	91.81	87.91	91.26	86.12
	LNSA	**91.48**	**86.36**	**91.70**	**85.35**	**92.04**	87.54	**91.48**	86.73	**92.37**	**88.11**	**91.81**	**87.50**

Table 2. Classification comparisons on MN10 and MN40.

Data	SP								MN40							
RecNet	AE		Folding		LAE		LFolding		AE		Folding		LAE		LFolding	
Metrics	MCD	HD	MCD	HD	MCD	HD	MCD	HD	MCD	HD	MCD	HD	MCD	HD	MCD	HD
PUGAN*	19.83	50.37	20.84	54.01	1.61	2.62	1.59	2.67	23.12	50.95	23.92	56.27	2.41	3.89	2.40	3.94
PFNet*	20.15	50.66	20.18	50.65	1.62	2.63	1.59	2.63	23.48	51.28	23.51	51.33	2.43	3.89	2.40	3.89
CRN*	14.36	50.63	20.18	50.65	1.44	2.65	3.10	5.72	17.22	49.26	23.51	51.33	2.16	3.89	3.69	6.92
PUGAN	0.32	1.88	0.36	3.83	0.32	1.02	0.27	1.11	0.73	5.85	0.77	7.29	0.45	1.71	0.38	1.97
PFNet	0.32	1.87	0.41	4.14	0.31	0.99	0.26	0.97	0.74	6.28	0.88	7.55	0.44	1.69	0.35	1.69
CRN	0.31	1.86	0.34	3.17	0.31	1.00	0.26	0.99	0.71	5.66	0.76	7.24	0.44	1.69	0.35	1.76
LNFC	0.23	1.66	**0.32**	2.71	0.15	1.27	0.15	1.22	**0.58**	5.43	0.75	6.80	0.23	2.05	0.22	1.98
LNSA	**0.23**	**1.66**	0.33	**2.57**	**0.13**	**0.65**	**0.14**	**0.76**	0.59	**5.30**	**0.75**	**6.65**	**0.17**	**1.05**	**0.19**	**1.37**

Table 3. Comparison with existing GAN discriminator losses on point clouds reconstruction. The symbol * denotes training with only the discriminator and no matching loss.

4.3 Comparisons with Discriminators-based Losses

Some recent works such as PFNet [11], CRN [22] and PUGAN [13] also use discriminators to enhance the reconstruction performances. However, these works are quite different from PCLossNet because they still use matching-based CD or EMD loss to constrain basic structural differences. In this section, we make a comparison with discriminator-based losses from these works in Table 3. We can see that existing discriminator-based losses show inferior performances over our methods, while they cannot work without matching-based losses. Our work, especially LNSA can achieve obvious improvements over even these existing discriminator-based losses, which confirms it can break the limitation of existing matching-based reconstruction losses.

4.4 Comparisons on Training Efficiency

In this section, we evaluate the training efficiencies of different loss functions on AE networks [1], which are measured by the time consumed training a single iteration. The results are presented in Table 4. We can see that our method LNSA can achieve the best reconstruction performances while getting nearly 4 times faster than EMD. Though CD and PFNet loss are faster than LNSA, they get the worst reconstruction results.

To explore the efficiency potential of our method, we also conduct a fast implement of LNSA named FLNSA. From Alg. 1 we can see that there are two back propagation processes in an iteration step of LNSA, which actually decreases the training efficiency.

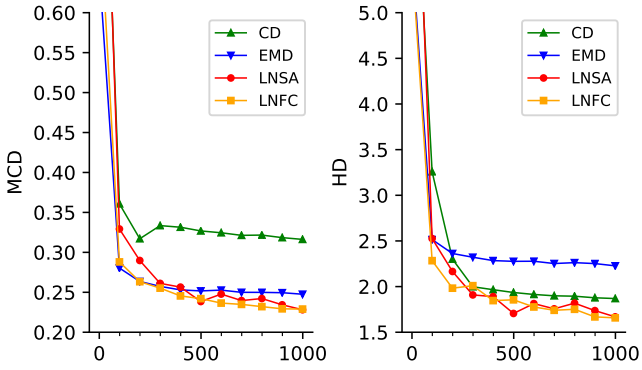

Fig. 4. Comparisons of the training processes.

In this condition, we randomly give single back propagation process in each iteration to improve the efficiency. In details, we assign 0.7 probability to train PCLossNet and 0.3 to train the reconstruction network. We can see that FLNSA would get same time efficiency with CD while getting much better reconstruction performances, which shows our potential on training efficiency.

Methods	CD	EMD	PUGAN	PFNet	CRN	LNFC	LNSA	FLNSA
Time(ms)	**23**	216	77	45	97	56	57	**23**
MCD↓	0.32	0.25	0.32	0.32	0.31	0.23	**0.23**	0.24
HD ↓	1.87	2.23	1.88	1.87	1.86	1.66	**1.66**	1.80

Table 4. Training efficiency comparison on AE network. The comparisons are conducted based on an NVIDIA 2080ti with a 2.9GHz i5-9400 CPU.

4.5 How is the Training Process Going?

To observe the convergence process of PCLossNet, we compare the reconstruction errors on ShapeNet and AE [1] during whole training process between different methods. The results are presented in Fig. 4. We can see that LNSA and LNFC have relatively large errors at the beginning of iterations because they need to search for the differences through training. But they will get lower errors than CD and EMD after enough iterations, which confirms that PCLossNet can help the reconstruction network converge better through the adversarial process.

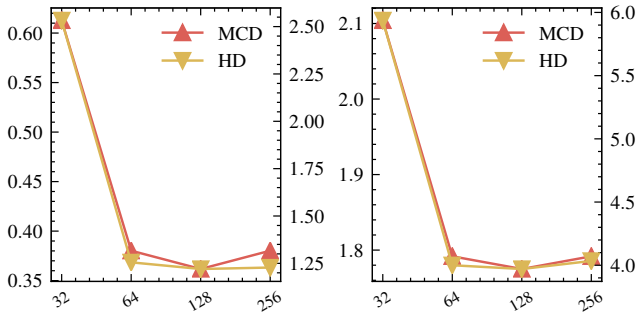

Fig. 5. Ablation study on the number of aggregation centers N_c. left and right vertical axises measure MCD and HD, while left and right pictures demonstrate results evaluates on ShapeNet and ModelNet40, respectively.

4.6 Ablation Study

Ablation for components in PCLossNet. To clarify the function of each component in the loss, we conduct an ablation study here. $L_{log\|\cdot\|}$ denotes log operation mentioned in Sec. 3.2, while L_{UC} and L_R are components of the loss function defined in Sec. 3.1. We can see each module makes sense. Removing anyone will reduce the performances.

$L_{\|\cdot\|}$	$L_{log\|\cdot\|}$	L_{UC}	L_R	SP		MN40	
				MCD	HD	MCD	HD
✓	✗	✗	✗	0.26	1.91	0.66	5.95
✓	✓	✗	✗	0.25	1.90	0.62	5.80
✓	✓	✓	✗	0.23	1.72	0.59	5.52
✓	✓	✓	✓	**0.23**	**1.66**	**0.59**	**5.30**

Table 5. Ablation study for components in PCLossNet.

Influence of aggregation centers number. The number of aggregation centers N_c is actually a very important hyper-parameter in PCLossNet, which decides the number of distributions to extract comparison matrices. To choose an appropriate value for N_c, we conduct an related ablation study based on AE [1] and LNSA, whose results are presented in Fig. 5. We use logarithmic coordinates for vertical axises to show the differences clearer. We can see that the reconstruction network can reach the smallest reconstruction error when N_c is 128. Though larger N_c has close results, the computational cost also increase.

Influence of PCLossNet learning rate. The learning rate of PCLossNet is also an interesting hyper-parameter. It should be higher than the learning rate of the reconstruc-

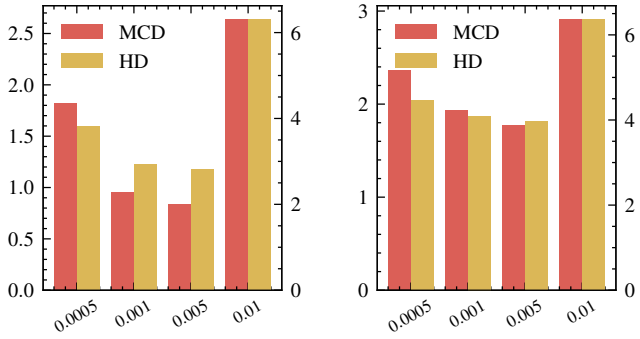

Fig. 6. Ablation study on the PCLossNet Learning rate. left and right vertical axises measure MCD and HD, while left and right pictures demonstrate results evaluates on ShapeNet and ModelNet40, respectively.

tion network to ensure that PCLossNet can find regions with greater differences in each iteration, which will help the reconstruction network converge well. However, too large learning rate may also block the convergence of PCLossNet. To determine the learning rate of PCLossNet, we conduct an ablation experiment and present it in Fig. 6. We can see that 0.005 is an proper option for PCLossNet.

5 Conclusion

In this work, we propose a novel learning-based framework named PCLossNet to help the point cloud reconstruction network to get rid of the limitation of commonly used matching processes. PCLossNet transforms differences between point clouds to the errors between extracted comparison matrices which are aggregated from point clouds with multiple distributions dynamically. By decoupling the extraction process into nonlinear Aggregation Controller module and 3D Euclidean space-based Aggregation processor, PCLossNet can get over the limitations of existing point-based discriminators and naturally supports adversarial training. By training in a generative-adversarial process together with the reconstruction network, PCLossNet can search for the main differences between reconstructed results and original point clouds and train the reconstruction network without any matching. Experiments on multiple datasets and reconstruction networks demonstrate reconstruction networks trained with PCLossNet can outperform those trained with matching-based losses, carrying smaller reconstruction errors and higher feature classification accuracy.

Acknowledge

We thank all authors, reviewers and the chair for the excellent contributions. This work is supported by the National Science Foundation 62088101.

References

1. Achlioptas, P., Diamanti, O., Mitliagkas, I., Guibas, L.: Learning representations and generative models for 3d point clouds. In: International conference on machine learning. pp. 40–49. PMLR (2018)
2. Arandjelovic, R., Gronat, P., Torii, A., Pajdla, T., Sivic, J.: Netvlad: Cnn architecture for weakly supervised place recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 5297–5307 (2016)
3. Arjovsky, M., Chintala, S., Bottou, L.: Wasserstein gan. arXiv preprint arXiv:1701.07875 (2017)
4. Fan, H., Su, H., Guibas, L.J.: A point set generation network for 3d object reconstruction from a single image. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 605–613 (2017)
5. Gadelha, M., Wang, R., Maji, S.: Multiresolution tree networks for 3d point cloud processing. In: Proceedings of the European Conference on Computer Vision (ECCV). pp. 103–118 (2018)
6. Goodfellow, I.J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y.: Generative adversarial networks. arXiv preprint arXiv:1406.2661 (2014)
7. Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V., Courville, A.C.: Improved training of wasserstein gans. In: Advances in neural information processing systems. pp. 5767–5777 (2017)
8. Han, Z., Wang, X., Liu, Y.S., Zwicker, M.: Multi-angle point cloud-vae: Unsupervised feature learning for 3d point clouds from multiple angles by joint self-reconstruction and half-to-half prediction. In: 2019 IEEE/CVF International Conference on Computer Vision (ICCV). pp. 10441–10450. IEEE (2019)
9. Huang, T., Liu, Y.: 3d point cloud geometry compression on deep learning. In: Proceedings of the 27th ACM International Conference on Multimedia. pp. 890–898 (2019)
10. Huang, T., Zou, H., Cui, J., Yang, X., Wang, M., Zhao, X., Zhang, J., Yuan, Y., Xu, Y., Liu, Y.: Rfnet: Recurrent forward network for dense point cloud completion. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 12508–12517 (2021)
11. Huang, Z., Yu, Y., Xu, J., Ni, F., Le, X.: Pf-net: Point fractal network for 3d point cloud completion. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 7662–7670 (2020)
12. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)
13. Li, R., Li, X., Fu, C.W., Cohen-Or, D., Heng, P.A.: Pu-gan: a point cloud upsampling adversarial network. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 7203–7212 (2019)
14. Liu, M., Sheng, L., Yang, S., Shao, J., Hu, S.M.: Morphing and sampling network for dense point cloud completion. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 34, pp. 11596–11603 (2020)
15. Mao, X., Li, Q., Xie, H., Lau, R.Y., Wang, Z., Paul Smolley, S.: Least squares generative adversarial networks. In: Proceedings of the IEEE international conference on computer vision. pp. 2794–2802 (2017)
16. Nguyen, T., Pham, Q.H., Le, T., Pham, T., Ho, N., Hua, B.S.: Point-set distances for learning representations of 3d point clouds. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 10478–10487 (2021)
17. Qi, C.R., Su, H., Mo, K., Guibas, L.J.: Pointnet: Deep learning on point sets for 3d classification and segmentation. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 652–660 (2017)

18. Qi, C.R., Yi, L., Su, H., Guibas, L.J.: Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In: Advances in neural information processing systems. pp. 5099–5108 (2017)
19. Rao, Y., Lu, J., Zhou, J.: Global-local bidirectional reasoning for unsupervised representation learning of 3d point clouds. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 5376–5385 (2020)
20. Urbach, D., Ben-Shabat, Y., Lindenbaum, M.: Dpdist: Comparing point clouds using deep point cloud distance. In: European Conference on Computer Vision. pp. 545–560. Springer (2020)
21. Wang, H., Jiang, Z., Yi, L., Mo, K., Su, H., Guibas, L.J.: Rethinking sampling in 3d point cloud generative adversarial networks. arXiv preprint arXiv:2006.07029 (2020)
22. Wang, X., Ang Jr, M.H., Lee, G.H.: Cascaded refinement network for point cloud completion. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 790–799 (2020)
23. Wang, Y., Sun, Y., Liu, Z., Sarma, S.E., Bronstein, M.M., Solomon, J.M.: Dynamic graph cnn for learning on point clouds. Acm Transactions On Graphics (tog) **38**(5), 1–12 (2019)
24. Wu, C.H., Hsu, C.F., Kuo, T.C., Griwodz, C., Riegler, M., Morin, G., Hsu, C.H.: Pcc arena: a benchmark platform for point cloud compression algorithms. In: Proceedings of the 12th ACM International Workshop on Immersive Mixed and Virtual Environment Systems. pp. 1–6 (2020)
25. Wu, T., Pan, L., Zhang, J., Wang, T., Liu, Z., Lin, D.: Density-aware chamfer distance as a comprehensive metric for point cloud completion. arXiv preprint arXiv:2111.12702 (2021)
26. Wu, Z., Song, S., Khosla, A., Yu, F., Zhang, L., Tang, X., Xiao, J.: 3d shapenets: A deep representation for volumetric shapes. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 1912–1920 (2015)
27. Yang, Y., Feng, C., Shen, Y., Tian, D.: Foldingnet: Point cloud auto-encoder via deep grid deformation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 206–215 (2018)
28. Yi, L., Kim, V.G., Ceylan, D., Shen, I.C., Yan, M., Su, H., Lu, C., Huang, Q., Sheffer, A., Guibas, L.: A scalable active framework for region annotation in 3d shape collections. ACM Transactions on Graphics (ToG) **35**(6), 1–12 (2016)
29. Yu, L., Li, X., Fu, C.W., Cohen-Or, D., Heng, P.A.: Pu-net: Point cloud upsampling network. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 2790–2799 (2018)
30. Yuan, W., Khot, T., Held, D., Mertz, C., Hebert, M.: Pcn: Point completion network. In: 2018 International Conference on 3D Vision (3DV). pp. 728–737. IEEE (2018)
31. Zhao, Y., Birdal, T., Deng, H., Tombari, F.: 3d point capsule networks. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 1009–1018 (2019)