3D Point Cloud Geometry Compression on Deep Learning

Tianxin Huang[1,2], Yong Liu[1,2*]
[1] The state key laboratory of industrial control technology, Zhejiang University [2] NetEase Fuxi AI Lab
21725129@zju.edu.cn yongliu@iipc.zju.edu.cn

ABSTRACT

3D point cloud presentation has been widely used in computer vision, automatic driving, augmented reality, smart cities and virtual reality. 3D point cloud compression method with higher compression ratio and tiny loss is the key to improve data transportation efficiency. In this paper, we propose a new 3D point cloud geometry compression method based on deep learning, also an auto-encoder performing better than other networks in detail reconstruction. It can reach much higher compression ratio than the state-of-art while keeping tolerable loss. It also supports parallel compressing multiple models by GPU, which can improve processing efficiency greatly. The compression process is composed of two parts. Firstly, Raw data is compressed into codeword by extracting feature of raw model with encoder. Then, the codeword is further compressed with sparse coding. Decompression process is implemented in reverse order. Codeword is recovered and fed into decoder to reconstruct point cloud. Detail reconstruction ability is improved by a hierarchical structure in our decoder. Latter outputs are grown from former fuzzier outputs. In this way, details are added to former output by latter layers step by step to make a more precise prediction. We compare our method with PCL compression and Draco compression on ShapeNet40 part dataset. Our method may be the first deep learning-based point cloud compression algorithm. The experiments demonstrate it is superior to former common compression algorithms with large compression ratio, which can also reserve original shapes with tiny loss.

CCS CONCEPTS

• **Computing methodologies ~ 3D imaging**; **Reconstruction**;
• Computing methodologies ~ Image compression;

KEYWORDS

3D point cloud; geometry compression; auto-encoder; hierarchical structure; detail reconstruction

* Corresponding author.

ACM Reference format:
Tianxin Huang and Yong Liu. 2019. 3D Point Cloud Geometry Compression on Deep Learning. In *Proceedings of the 27th ACM International Conference on Multimedia, Nice, France, Oct. 21-25, 2019 (MM '19)*, ACM, NY, USA. 8 pages. https://doi.org/10.1145/3343031.3351061

1 INTRODUCTION

3D point cloud data is playing a more and more important role in many real-life fields, such as automatic navigation and immersive communication. It puts higher and higher requirement on efficiency of data transmission and memory of data storage. Under the circumstance, better lossless point cloud data compression methods with higher compression ratio are necessary. Deep learning is a new idea for data compression of 3D point cloud, supposed to reach higher compression ratio than existing methods and keep tiny loss of data. With appropriate structure, the network can find out one reasonable solution for the problem of point cloud compression. However, existing deep learning methods for point cloud reconstruction do poorly in detail reconstruction. On this occasion, we propose an auto-encoder network having good performance on detail representation of 3D model, better than the state of art. By adding constraints to size of codeword extracted by the encoder, our network can reach higher compression ratio than almost any existing 3D point cloud compression method. Besides, compression by deep learning is based on matrix operations, which means our compression and decompression processes can use GPU to achieve parallel processing, dealing with multiple models at a time, which can greatly improve efficiency.

Our network architecture is shown in Figure 1. In this work, we use similar hierarchical structure proposed in PointNet++ as our encoder to capture local structure of the model. As for the decoder, we use a reverse hierarchical neutral network to generate models of different resolutions gradually. High-resolution model will grow up from former low-resolution model. The progressive generation method is good at detail reconstruction because network between two output layers only needs to add details to the basic framework. Data compression process for point cloud data based on deep learning can be divided into two parts. Firstly, a model is encoded by the network encoder to a codeword; Then the codeword is further encoded by sparse coding. Receiver will decode the codeword containing information of raw model in reverse order. It will be then fed to the decoder to reconstruct point cloud.

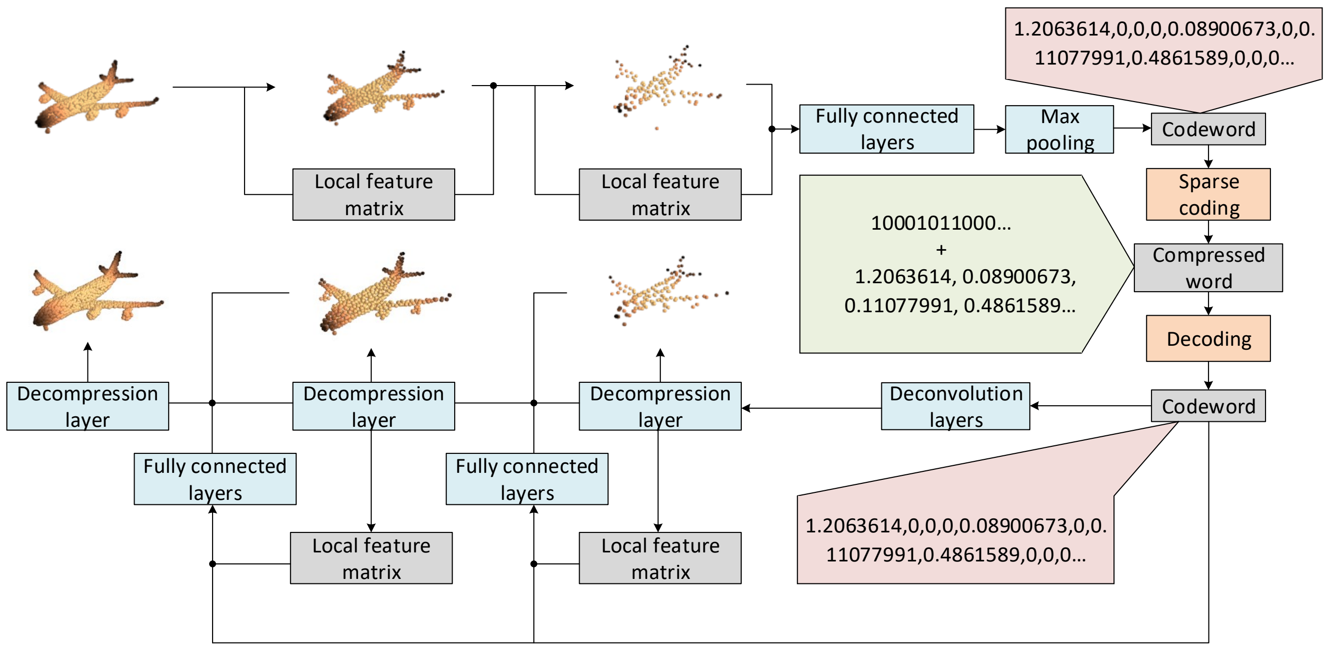


Figure 1 The architecture of our network. The whole process is consisting of network part and sparse coding part. The network part achieves data compression and decompression by a pair of encoder and decoder. Codeword from encoder is sparse due to sparse constraint in loss function. Sparse coding is used to improve compression ratio by further compressing sparse codeword from network encoder. Compressed codeword is transferred from sender to receiver. Encoder and decoder will be saved separately by sender and receiver to finish the whole process.

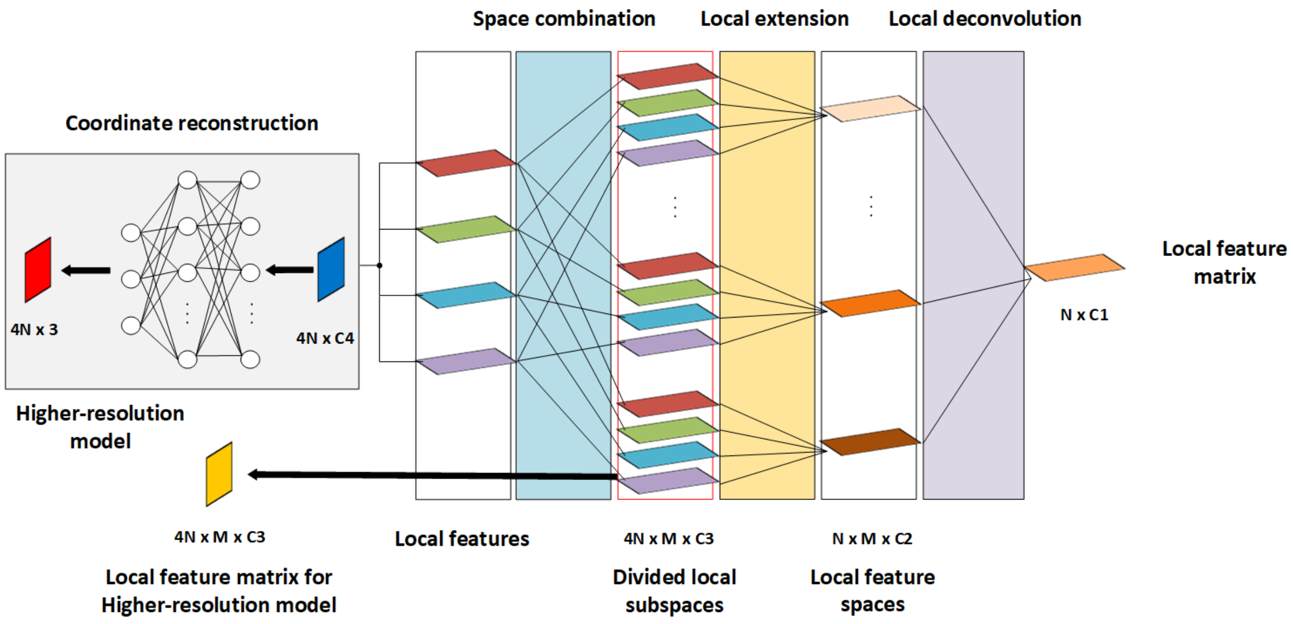


Figure 2 Structure of Decompression layer. Our decompression layer generates local feature spaces consist of M feature vectors by deconvolution. Then the local feature spaces are divided to 4 subspaces by deconvolution in another dimension. Subspaces are finally combined into single feature vectors again by convolution. In this case, raw feature vectors are expanded to 4 times more, then transformed into higher resolution model. Meanwhile, local feature vectors from subspaces are directly propagated to next decompression layer and merged with global feature to make up local feature matrix in next decompression layer.

Last layer of the encoder is named as bottleneck layer. Dimension of bottleneck layer and bits occupied by each floating number in the codeword are two key factors of compression result. There are three kinds of data type for floating number: semi-precision floating number taking up 16 bits, single-precision floating number taking up 32 bits, and double-precision floating number taking up 64 bits. In our work, we test the influence of dimension of bottleneck layer on the recovery effect in 16 bits and 32 bits data type precision of codewords. We also give a best compression ratio for 3D point clouds by short codeword and low data type precision of codewords in sec 4.

Our contribution can be summarized as:

We propose a new deep auto-encoder processing unorder point clouds data with lower reconstruction loss and stronger detail reconstruction ability than former unsupervised neutral networks;

We design a new deep learning-based method for sparse point clouds geometry compression. It can reach higher compression ratio than any existing compression methods with acceptable loss. It also provides three outputs in different resolutions, suitable for different occasions.

2 RELATED WORKS

2.1 Representation Learning for Point Cloud

Most neutral networks dealing with point cloud are based on the voxel model such as [1],[2] and [3], partitioning raw model into 3d regular voxels like pixel in 2d. However, point cloud data is often sparse in Euclidean space. It leads to great waste of time and memory to convert raw data detected in coordinates format to voxels. Besides, resolution of models is greatly restricted by memory, since models with higher resolution ask for much smaller voxels. OctNet proposed by Gernot et al. [4] tries to reduce computational and memory requirements by converting voxel models to unbalanced octrees with different size of leaf nodes. In this way, high-resolution model can be represented as the combination of voxels of different sizes, reducing memory waste and improve computation efficiency. But it still wastes a lot to convert raw data in xyz format to voxels. Kd-Network [5] deals with point cloud based on Kd-trees to reduce space complexity. Some works also describes 3d point cloud models as multi-view images, such as [6] and [7]. However, conversion to collections of images may destroy spatial feature of 3d model.

PointNet [8] and PointNet++ [9] have started the fashion to process point clouds in xyz format directly. They concentrate on point clouds classification and scene segmentation. FoldingNet by Yao et al. [10] may be first one learning representation of model from unorder raw point clouds directly. They modified a new operation named folding to complete point clouds reconstruction from codewords extracted by similar 1-D convolution with PointNet. The folding operation means to combine feature with 2-D grid samples and fold the 2-D grid to 3D models. This operation can save a lot of memory. Their folding operation has been used as an efficient method in point clouds reconstruction to expand dimension such as in [11]. However, some subtle shape may be omitted in generation because they are too difficult to be transformed from 2-D grid, such as the porous structure. Panos et al. [12] build an auto-encoder on convolution layers and fully connected layers to learn representation of unorder point clouds. They train a few different GANs [13] based on the auto-encoder as generative networks and compare their generation ability. Their network gives a good generation result on unseen shapes, while still weak in details reconstruction. Jiaxin Li et al. [14] propose a new feature extraction method by sampling and grouping model with self-organized maps (SOM), instead of directly sampling by farthest point sampling in PointNet ++. It improves classification accuracy greatly.

2.2 Point Cloud Geometry Compression

Point cloud compression has always been an important research orientation in computer graphics since the increasing capability of 3D data scanning devices. Point cloud compression includes three main types [15]: geometry compression, attribute compression and dynamic motion-compensated compression. We concentrate on geometry compression in this paper. Some compression algorithms compress point clouds by saving data in a memory efficient data structure such as [16], [17] and [18]. Some encode point cloud by building a mathematic model to describe point cloud structure like [19]. Tim.et al [20] propose a compression method by transforming 3d point cloud to 2D maps, compressing them with image compression algorithms such as JPEG. Two common point cloud compression methods are PCL [21] and Draco [22]. PCL compresses point cloud by converting raw data to voxels and encoding the voxels with an octree. Its compression ratio can be adjusted by changing the voxel precision. Draco provides 10 alternative compression levels. It also supports further lossy compression by reducing quantization bits.

3 HIERARCHICAL AUTO-ENCODER

3.1 Multiple Scale Feature Extraction

PointNet and PointNet++ are the pioneers of directly using raw point cloud data only composed of coordinates. PointNet uses 1-D convolutional layers with kernel size 1 to extract point feature, and max pooling layer to produce a joint representation, invariant to permutation of point sets. However, PointNet doesn't offer much attention to preserve local structure of raw data. On this occasion, PointNet++ applies PointNet recursively in each local region to acquire feature of local structures. We use modified multiscale hierarchical encoder in our work. Local region centers are obtained by farthest point sampling (FPS), which means to choose point farthest from all former sampled points in rest point set iteratively. And local regions are acquired by calculating k nearest neighbors in r radius bounding sphere. If the number of points in this bounding sphere is less than k, then the region center will be replicated to keep k neighbors in the bounding sphere. Features of local regions are also extracted by 1-D convolutional layers and max pooling layers. Centers of local regions are used to confirm new centers for next local feature extraction.

Coordinates of these centers are combined with local features to make up new features fed to next local feature extraction layer. Finally, all local features will be combined by a global max pooling layer to produce a synthesize feature. However, hierarchical encoder in PointNet ++ ignores to reserve basic shape feature. The lowest resolution model sampled describes contour of raw model. Its concatenation with feature matrix may affect the reservation of shape feature. So, we use an additional structure to extract feature of the lowest-resolution model. Feature extracted from the lowest-resolution model is concatenated together with synthesize feature from former hierarchical extraction. In this way, feature of basic contour is strengthened in the final codeword, meaningful to improve reconstruction quality.

3.2 Hierarchical Reconstruction

As for the reconstruction of point cloud from the compressed data, we use a hierarchical structure to improve the ability to reconstruct details. The structure consists of three output layers in different resolutions. The output of first output layer gives a basic frame of the whole point cloud, and latter layers add more details to the frame gradually. The output of latter layer relies on former output. In this way, we can get a multi-resolution representation of raw point cloud data.

The core of more accurate generation is the decompression layer, shown in Figure 2. Local feature decompressed in the last part, together with codeword containing global feature, and the output coordinates are concatenated together and fed to the decompression layer to acquire new local feature matrix. The decompression layer is achieved by local deconvolution and local extension architectures shown in Figure 2.

The decompression process is similar to an upsampling process in [23]. They all generate more precise model from a fuzzier one. However, size of first two models in out structure are too small to contain enough structure information, almost impossible to reconstruct more precise model by directly upsampling former model. So, our decompression layer combines local feature matrix and fuzzier model together to make accurate reconstruction.

3.3 Loss Function

Loss function is important for the training process. The key is how to evaluate the difference between two similar point cloud models. There are a few kinds of methods to evaluate the difference. The comparison of training results of them is in sec.4.

3.1.1 Chamfer Distance.

The Chamfer Distance (CD) is a commonly used metric to calculate the loss for point sets. It measures the mean distance of one point to its nearest neighbor between two point sets.

$$loss_{CD}\left(S_1,S_2\right)=max\left\{\begin{matrix}\frac{1}{\left|S_1\right|}\sum_{x\in S_1}\min_{y\in S_2}\left\|x-y\right\|_2, \\ \frac{1}{\left|S_2\right|}\sum_{x\in S_2}\min_{x\in S_1}\left\|x-y\right\|_2\end{matrix}\right\}$$

3.1.2 Earth Mover's Distance.

Similar to the Chamfer Distance, the Earth Mover's Distance (EMD) [24] is also a commonly used loss function. It is supposed to perform better than CD, also far more complicated. EMD tries to find a bijection between two point sets and calculate the mean distance between corresponding points:

$$loss_{EMD}\left(S_1,S_2\right)=\min_{\varnothing:S_1\rightarrow S_2}\sum_{x\in S_1}\left\|x-\varnothing\left(x\right)\right\|_2$$

$\varnothing$ is the bijection from point set S_1 to S_2 . The calculation of EMD needs to solve an optimization problem to acquire the bijection, which is a great waste of time and memory.

3.1.3 Root Mean Square Error.

Root Mean Square Error (RMS) is not often used as loss function. However, another metric related to RMS, Peak Signal to Noise Ratio (PSNR), is widely used in point cloud compression to evaluate the quality of decompression.

The RMS is defined as square root of mean squared distance of one point to its nearest neighbor between two point sets:

$$RMS\left(S_1,S_2\right)=\sqrt{max\left\{\begin{matrix}\frac{1}{\left|S_1\right|}\sum_{x\in S_1}\min_{y\in S_2}\left\|x-y\right\|_2^2, \\ \frac{1}{\left|S_2\right|}\sum_{y\in S_2}\min_{x\in S_1}\left\|x-y\right\|_2^2\end{matrix}\right\}}$$

3.1.4 Multiscale Loss.

Former metrics to evaluate the model difference is lack of constraint of local details. Taking consideration of local difference can accelerate convergence of network and improve detail reconstruction ability. We combine local Loss evaluation from multiple scales of the sampled points' neighbors with the basic Loss evaluation method to make up the Multiscale loss function. It is defined as:

$$Loss_{multiscale}\left(S_1,S_2\right)=\xi\cdot Loss\left(S_1,S_2\right) \\ +\sum_r\xi_r\cdot\frac{1}{N}\sum_{i=1}^{N}Loss\left(S_1^{i,r},S_2^{i,r}\right)$$

Table 1 Illustration of training process

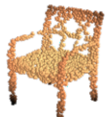
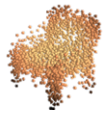
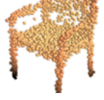
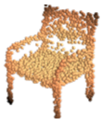
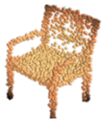
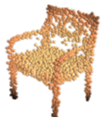
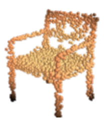
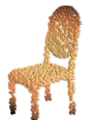
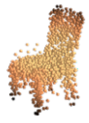
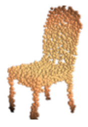
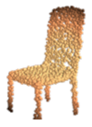
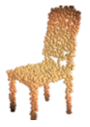
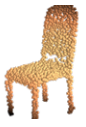
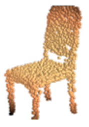
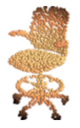
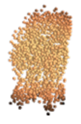
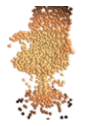
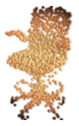
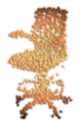
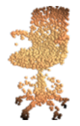
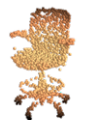
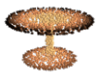
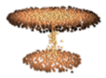
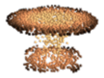
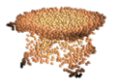
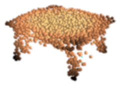
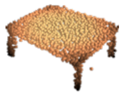
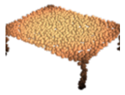
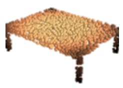
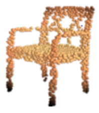
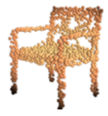
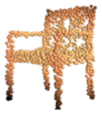
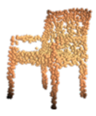
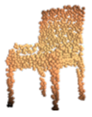
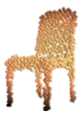
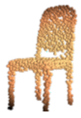
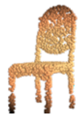
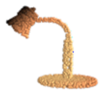
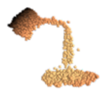
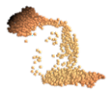
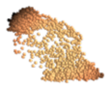
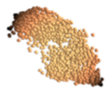
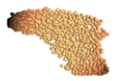
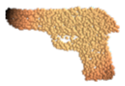
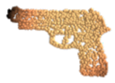
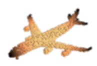
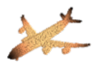
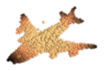
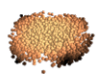
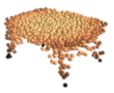
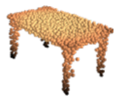
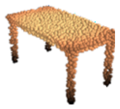
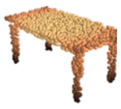
input	1 epoch	20 epochs	40 epochs	60 epochs	80 epochs	100 epochs
						
						
						

Table 2 Interpolation between point clouds in latent space

Source	Interpolation between different point clouds in same class						Target
							
							
	Interpolation between point clouds in different classes						
							
							

ξ is the coefficient of frame loss, and ξ_r means the coefficient of loss of local region with radius of r. $S_1^{i,r}$ and $S_2^{i,r}$ mean local point sets located by the ith sampled central point with radius of r from raw point sets S_1 and S_2 .

3.1.5 Sparse Constraint.

To reach higher compression ratio, we need to restrain the size of compressed codeword. Raw codeword is further compressed by sparse coding. Sparse coding means to record positions of nonzero in codeword by encoding these positions into binary format.

In this way, a codeword of size 256 only need 8 32 dimensional floats to record its nonzero distribution, followed by n nonzero values to contain all information.

So, decreasing the number of nonzero values in codeword can improve compression ratio. With that in mind, L1 regularization is used to reduce the number of nonzero values in codeword. It is defined as:

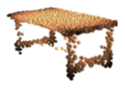
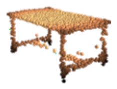
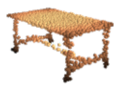
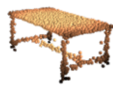
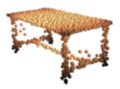
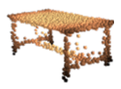
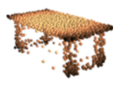
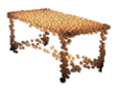
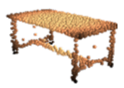
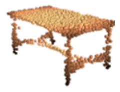
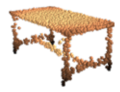
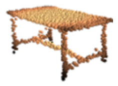
$$CodeLimit(V) = \varepsilon \cdot \sum_i |V(i)|$$

V is the codeword, and ε is the coefficient of the regularization.

3.1.6 Sparse Multiscale Loss.

Finally, to improve compression ratio as high as possible while keeping good reconstruction quality, we combine Multiscale loss and sparse constraint to compose the Sparse Multiscale loss. It is defined as:

Table 3 Compression experimental result

32 bits							Source
Bottleneck size	8	16	32	64	128	256	
bpp	0.1170	0.1408	0.1734	0.2258	0.2989	0.3936	
loss	0.0663	0.0643	0.0630	0.0634	0.0628	0.0628	
shape							
16 bits							
Bottleneck size	8	16	32	64	128	256	
bpp	0.0380	0.0530	0.1026	0.1261	0.1897	0.2551	
loss	0.0732	0.0670	0.0629	0.0629	0.0634	0.0620	
shape							

$$Loss_{sparse\,multiscale}(S_1,S_2)=\xi\cdot Loss(S_1,S_2)+\sum_r \xi_r\cdot\frac{1}{N}\sum_{i=1}^{N}Loss\left(S_1^{i,r},S_2^{i,r}\right)+\varepsilon\cdot\sum_j\left|V(j)\right|$$

The meanings of coefficients are same with former parts.

4 EXPERIMENT

We train our network on ShapeNet part dataset[25] containing 12288 models in train split, 2874 models in test split, 16 categories from ShapeNet dataset. We train it using ADAM [26] optimizer with an initial learning rate of 0.001, batch size of 16, for 100 epochs. Our experimental platform is a TITAN Xp GPU with 3.5Ghz Xeon CPU. Point sets in coordinates format are acquired by sampling random points on the triangles from corresponding mesh models.

Because ShapeNet dataset is a sparse point clouds dataset. It is not often used in point cloud compression task. However, sparse point clouds can be used to test point clouds compression algorithms because they are even more difficult to be compressed than dense ones. Sparse point clouds contain less redundant information than dense ones because they represent shapes or surfaces by less points.

We compare the difference between the compression of sparse point clouds and dense ones on simplified Stanford bunny point cloud down sampled to 2048 points and 35947 points. PCL and Draco are applied to compress the two models. The result is in Table 4. The bpp means bits per point of compressed data and loss is calculated by RMS to save computation. It proved Compression on sparse point clouds is often harder to reach high compression ratio and low reconstruction loss than that on dense ones. Besides, we focus on the superiority of deep-learning based compression methods in this paper. So, it is enough and sound to use same sparse dataset to evaluate our compression quality here.

4.1 Visualization of Training Process

To clearly show training process of the network, we choose a few models and display their reconstructed point clouds after different number of iterations, the result is in Table 1. From the result, we can see that details appear gradually during the training process. To demonstrate our encoder extracts efficient features from point clouds, we implement interpolation between two codewords in the latent space and reconstruct model on the interpolation results. The model generalized is shown in Table 2.

Table 4 Sparse and dense point clouds compression

Points	2048		35947	
Shape				
	bpp	Loss	bpp	Loss
Draco	1.02	0.0014	0.27	0.0013
PCL	2.50	0.0011	1.34	0.0009

4.2 3D Point Cloud Geometry Compression

Two key factors of compression ability are dimension of bottleneck layer in encoder and bits for a single value in codeword. To improve compression ratio, we add sparse constraint for codeword to the loss function in this part. The result when the size of bottleneck layer changes from 8 to 256 with 16 bits and 32 bits for a single value is shown in Table 3. We choose a sample from test split of dataset to demonstrate the influence of higher compression ratio on reconstruction. Loss between raw model and decompressed model is calculated by mean Multiscale CD on the whole test split of dataset.

From result in Table 3, we can see that changing bits for single value or increasing bottleneck size makes little difference when bottleneck size is large. The reason is sparse constraints remove redundant information from codeword. Bits for single value or bottleneck size only have considerable influence when the codeword is too short to contain enough information. Then we can conclude from Table 3 that codeword with bottleneck size of 32 and 16 bits for single value is the best compression condition with exact information. Loss increases greatly when bottleneck size is smaller. So, our method can reach a best compression ratio of 0.1026 bpp with reconstruction loss of 0.0629 on average.

We compare our deep learning-based compression method with PCL and Draco in ultimate condition when bpp is small. The result is displayed in Figure 3. Ours-16 and Ours-32 mean our method with 16 bits and 32 bits for a single value. Numerical analysis result is shown in Table 5. The compression ratio, reconstruction loss and computation load consist of time cost and minimum memory requirement are discussed then. We test our work with batch size of 1 and 16. The result shows that our work can reach 10 times higher compression ratio than Draco and even 110 times higher than PCL while keeping a smaller reconstruction loss.

However, PCL and Draco are faster and more memory saving. It is because both PCL and Draco are based on C Program, which can be much more efficient than Python. Besides, all intermediate matrices may be saved in the memory during the forward propagation process in TensorFlow framework, which will greatly increase memory cost in our work. So, simplifying network and replacing more operation in TensorFlow with C program may improve the efficiency and reduce memory load.

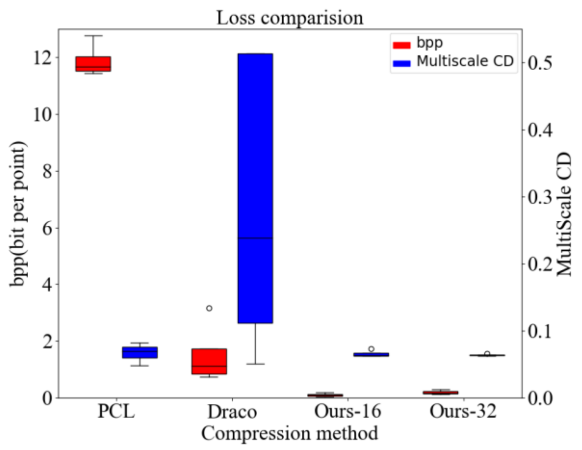


Figure 3 Comparison with other compression algorithms for point clouds in high compression ratio. Compression ratio is evaluated with bpp (bits per point) and the loss is calculated by Multiscale CD. Smaller bpp means higher compression ratio. Compression ratio is improved by increasing voxel size in PCL and decreasing the quantization bits in Draco.

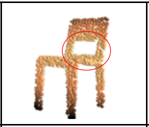
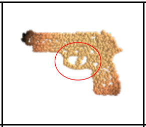
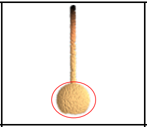
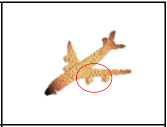
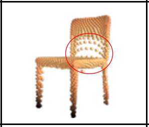
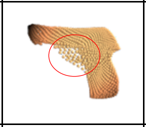
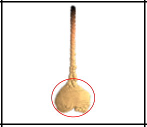
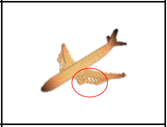
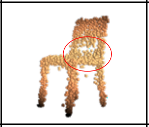
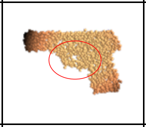
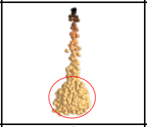
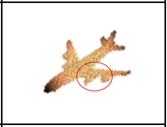
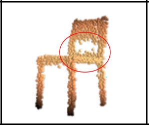
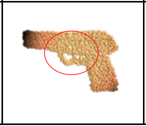
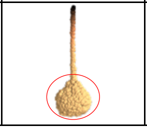
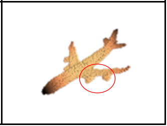
Table 5 Numerical comparison of compression methods

	PCL	Draco	Our batch 1	Our batch 16
Memory (on CPU)	26MB	**3MB**	-	-
Memory (on GPU)	-	-	**611MB**	2399MB
Encode time	0.0105s	**0.142ms**	0.19s	0.67s
Decode time	0.0154s	**0.125ms**	0.01s	0.11s
bpp	11.924	1.51	**0.1026**	
loss	0.0671	0.4708	**0.0629**	

Table 6 Numerical comparison of reconstruction quality

	FC	FN	Our work
loss	0.0656	0.0668	**0.0618**

Table 7 Comparison of reconstruction quality

Raw				
FN				
FC				
Ours				

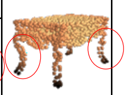
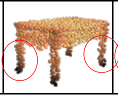
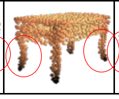
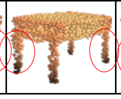
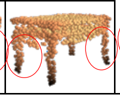
We compare reconstruction quality of our network with the state-of-art. The result is displayed in Table 7 with fragile details highlighted. It shows that our network performs well in reconstructing details such as trigger on the pistol or hole on the back of chair. Although FoldingNet (FN) [10] creates a smoother surface, it cannot reconstruct discontinuous shape or non-manifold surface well due to its folding operation transforming 2D plane to 3D surface. The fully connected auto-encoder (FC) [12] improves that point. But it may build rougher surfaces. Our network can reconstruct discontinuous shape smoothly, better than both FN and FC.

Numerical comparison result is given in Table 6. The loss is calculated by Multiscale CD proposed in section 3.1, which is proved proper to evaluate reconstruction quality in section 4.4. The result shows our work can reach a lower reconstruction loss than both FN and FC.

4.3 Influence of Local Feature Size

Our encoder is based on the local feature extraction of raw point clouds. Changing neighbor size for local feature extraction will greatly affect the training result. We take a research on the relationship between local region size and reconstruction quality. The result is in Table 8.

Table 8 Reconstruction comparison in different neighbor size for feature extraction. Loss is Multiscale CD with neighbor size of 0.1, 0.2 and 0.4, not 0.05, 0.1, 0.2 and 0.4 in other parts. Because it is not fair to evaluate loss with minimum radius of 0.05 when changing minimum feature extraction radius from 0.02 to 0.1

Neighbor size	0.02	0.05	0.08	0.1	Source
Loss	0.0519	**0.0518**	0.0523	0.0520	
Sample					

From the comparison we can see neighbor size of 0.05 has the best reconstruction quality for details like the bending table legs in Table 8. The result shows that neighbor size for local feature extraction has a great impact on detail reconstruction. Too small neighbor size cannot contain enough points to describe a local feature, while too big size ignores details in the local region, both leading to imprecise reconstruction.

4.4 Influence of Loss Function

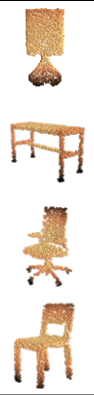
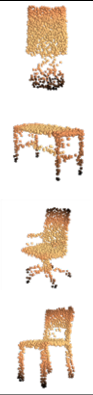
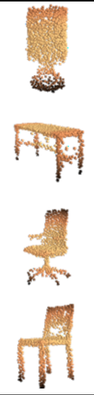
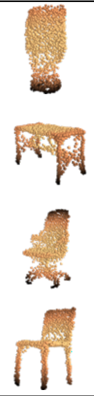
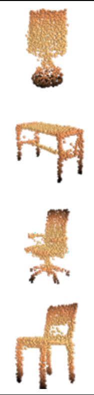
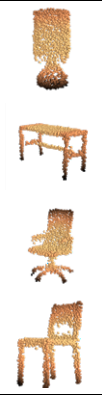
We train our network with different loss functions. The result is evaluated by different loss metrics on test split of dataset and reported in Table 9. To visually compare influence of different loss functions, we display reconstruction result of a few models on networks trained with different loss functions in Table 10. Multis-RMS means Multiscale RMS and Multis-CD means Multiscale CD.

We can see that CD and RMS can reserve basic contour of models, while leading to ununiform reconstruction. Some regions contain sparser points while some contain denser. Though EMD can reach more uniform result, it gives rougher shape. Multiscale RMS and Multiscale CD can both achieve accurate and uniform reconstruction. According to Table 9, network trained with CD reach both smaller CD loss and RMS loss, demonstrating CD is stronger constraint than RMS. Multiscale CD and Multiscale RMS follow the same rule. So, Multiscale CD is also stronger than Multiscale RMS. Multiscale CD is the best loss function, powerful to train network making precise and uniform reconstruction.

Table 9 Losses on the test split

Training Loss	RMS	CD	EMD	Multis-RMS	Multis-CD
Mean CD	0.0320	**0.0305**	0.0330	0.0325	0.0316
Mean RMS	0.0408	**0.0386**	0.0425	0.0419	0.0408
EMD	0.1065	0.1163	**0.0568**	0.0692	0.0683
Multis-CD	0.0664	0.0646	0.0621	0.0627	**0.0618**
Multis-RMS	0.0837	0.0817	0.0759	**0.0772**	**0.0772**

Table 10 Reconstruction quality of networks trained with different loss function

Source	RMS	CD	EMD	Multis-RMS	Multis-CD
					

5 CONCLUSION

In this work, we present a new method for 3D point cloud compression by deep learning, also the first deep-learning based point cloud geometry compression algorithm. It shows great potential due to its parallel processing ability and higher compression ratio than any other methods while keeping acceptable loss. The experiment has shown that it outperforms PCL and Draco in high compression ratio condition. The key of our compression method is an auto-encoder with hierarchical structure performing better than the state-of-art in reconstruction quality, especially on local details. More work will be devoted to further improving the reconstruction quality. There is no doubt that deep learning is the future development direction of data compression.

ACKNOWLEDGMENTS

This work is supported by the National Natural Science Foundation of China under Grant U1509210， 61836015.

REFERENCES

[1] A. Brock, T. Lim, J. M. Ritchie, and N. Weston. Generative and discriminative voxel modeling with convolutional neural networks. Advances in Neural Information Processing Systems, Workshop on 3D learning, 2017. 3

[2] A. Dai, A. X. Chang, M. Savva, M. Halber, T. Funkhouser, and M. Nießner. Scannet: Richly-annotated 3D reconstructions of indoor scenes. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017. 3

[3] D. Maturana and S. Scherer. Voxnet: A 3D convolutional neural network for real-time object recognition. In IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pages 922–928. IEEE, 2015. 3

[4] Riegler G, Osman Ulusoy A, Geiger A. Octnet: Learning deep 3d representations at high resolutions[C]//Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2017: 3577-3586.

[5] Klokov R, Lempitsky V. Escape from cells: Deep kd-networks for the recognition of 3d point cloud models[C]//Proceedings of the IEEE International Conference on Computer Vision. 2017: 863-872.

[6] Zhu Z, Wang X, Bai S, et al. Deep learning representation using autoencoder for 3D shape retrieval[J]. Neurocomputing, 2016, 204: 41-50.

[7] Chen D Y, Tian X P, Shen Y T, et al. On visual similarity based 3D model retrieval[C]//Computer graphics forum. Oxford, UK: Blackwell Publishing, Inc, 2003, 22(3): 223-232.

[8] Qi C R, Su H, Mo K, et al. Pointnet: Deep learning on point sets for 3d classification and segmentation[C]//Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2017: 652-660.

[9] Qi C R, Yi L, Su H, et al. Pointnet++: Deep hierarchical feature learning on point sets in a metric space[C]//Advances in Neural Information Processing Systems. 2017: 5099-5108.

[10] Yang Y, Feng C, Shen Y, et al. Foldingnet: Point cloud auto-encoder via deep grid deformation[C]//Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2018: 206-215.

[11] Mandikal P, Babu R V. Dense 3D Point Cloud Reconstruction Using a Deep Pyramid Network[J].

[12] Achlioptas P, Diamanti O, Mitliagkas I, et al. Learning Representations and Generative Models for 3D Point Clouds[C]//International Conference on Machine Learning. 2018: 40-49.

[13] Goodfellow I, Pouget-Abadie J, Mirza M, et al. Generative adversarial nets[C]//Advances in neural information processing systems. 2014: 2672-2680.

[14] Li J, Chen B M, Hee Lee G. So-net: Self-organizing network for point cloud analysis[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2018: 9397-9406.

[15] Shao Y, Zhang Q, Li G, et al. Hybrid Point Cloud Attribute Compression Using Slice-based Layered Structure and Block-based Intra Prediction[C]//2018 ACM Multimedia Conference on Multimedia Conference. ACM, 2018: 1199-1207.

[16] Schnabel R, Klein R. Octree-based Point-Cloud Compression[J]. Spbg, 2006, 6: 111-120.

[17] Gumhold S, Kami Z, Isenburg M, et al. Predictive point-cloud compression[C]//Siggraph Sketches. 2005: 137.

[18] Morell V, Orts S, Cazorla M, et al. Geometric 3D point cloud compression[J]. Pattern Recognition Letters, 2014, 50: 55-62.

[19] de Queiroz R L, Chou P A. Transform coding for point clouds using a gaussian process model[J]. IEEE Transactions on Image Processing, 2017, 26(7): 3507-3517.

[20] Kammerl J, Blodow N, Rusu R B, et al. Real-time compression of point cloud streams[C]//2012 IEEE International Conference on Robotics and Automation. IEEE, 2012: 778-785.

[21] Rusu R B, Cousins S. Point cloud library (pcl)[C]//2011 IEEE International Conference on Robotics and Automation. 2011: 1-4.

[22] "Draco 3D graphics compression," https://google.github.io/draco/, Accessed: 2018-01-10.

[23] Yu L, Li X, Fu C W, et al. Pu-net: Point cloud upsampling network[C]//Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2018: 2790-2799.

[24] Rubner, Y., Tomasi, C., and Guibas, L. J. The earth mover's distance as a metric for image retrieval. IJCV, 2000.

[25] Yi L, Kim V G, Ceylan D, et al. A scalable active framework for region annotation in 3d shape collections[J]. ACM Transactions on Graphics (TOG), 2016, 35(6): 210.

[26] D. Kingma and J. Ba. Adam: A method for stochastic optimization. In Int. Conf.on Learning Representations (ICLR), 2015.