


A Survey on 3D Point Cloud Compression Using Machine Learning Approaches

Reetu Hooda ^{*}†, W. David Pan ^{*} *Senior Member, IEEE* and Tamseel M. Syed ^{*} *Senior Member, IEEE*

^{*}Department of Electrical & Computer Engineering, University of Alabama in Huntsville, Huntsville, AL, USA 35899

[†]Email: rh0059@uah.edu

Abstract—Machine learning has been widely used for solving several data processing tasks and recently found applications in data compression domain as well, notably for point cloud (PC). Compression techniques based on deep learning (DL) methods such as convolutional neural network (CNN) have enabled exploiting higher dimensional correlations for improved performance. The most common DL based choice for point cloud compression (PCC) is an autoencoder, while there are few implementations that use recurrent neural network (RNN) and fully connected neural network. This paper surveys the on-going research on PCC using ML approaches. The benchmark datasets with performance metrics are also included. The survey shows the machine learning based methods offer performance comparable to conventional coding methods, while point out directions of promising improvements in the future.

Index Terms—Point cloud compression, Deep Learning, Autoencoders, Convolutional neural network.

I. INTRODUCTION

With the increasing use of virtual/augmented reality (VR/AR) in modern day applications like autonomous driving [1], gaming industry [2], etc., the need for efficient transmission of 3D data rises. A point cloud (PC) is a volumetric representation format for 3D data comprising of a set of unorganized points in the 3D space. Each point consists of spatial coordinates (X, Y, Z) that describe the geometry over a surface and attribute information such as color values (R, G, B) and/or normal vectors (n_x, n_y, n_z) that describe the visual appearance [3]. PC offers a highly realistic representation of an object which assists in creating an interactive/immersive visual experience to its users [4].

The advantage of having more than one viewpoint is helpful in practical applications where complicated decision making tasks are performed by relying on real-time multi-dimensional view of a scene or an object [5]. Some applications that could benefit from point cloud representation of data include robotic applications [6], preservation of cultural heritage, or historical architectures, video streaming for 5G systems [2], etc.

With the advancements in 3D point cloud acquisition technologies, it is now more affordable to generate a PC with a very high precision capturing several finer details. However, there exists an overhead in terms of increased bandwidth as well as information storage and processing when transmitting PC data over a communication channel [7]. Such transmission of volumetric data (represented using a PC) becomes even

more challenging for transmission in a wireless setting [8], [9].

As mentioned in [8], transmitting PC data using wireless channels is even more challenging due to the lack of efficient encoding, adaptive transmission methods, effective quality metrics and accurate viewing angle prediction. Furthermore, the prohibitively large resources required for transmitting PC data are generally limited, therefore, efficient point cloud compression (PCC) has become an increasingly important issue to be addressed for the practical use of a PC [10].

In the past decade, machine learning (ML) has become an emerging field that offers solutions to problems associated with image and video data [11]. These ML solutions also include data compression techniques with encouraging results that are comparable to traditional methods [12], [13]. Therefore, ML techniques were explored to offer potential solutions to point cloud compression.

Additionally, the advances in artificial intelligence (AI) or ML has provided an opportunity to consider the use of AI/ML in real-time PC video streaming [14]. As mentioned in [14], the transmission of PC data using AI-powered techniques provides more than 30 times compression ratio for their network conditions.

Although ML has the flexibility to exploit 3D correlations in the data, due to the non-uniformity and unstructured nature of point clouds, the application of ML techniques to PCC is not straightforward. Hence, most relevant ML methods first convert the 3D data into a corresponding 2D format as a preprocessing step before feeding it to an ML-based compression algorithm [15]. There are a few research surveys that provide an overview of recent PCC techniques, but to the best of our knowledge, this paper is a first attempt to provide a structured review of ML-based solutions to PCC in recent years. The paper includes the benchmark data sets with performance metrics used to evaluate previous contributions [16]. The metric comparison is summarized in Table I.

Given the fundamental representation of PC data, the research problems related to PC Compression (PCC) are categorized into two groups:

- 1) Geometry Compression: Focused on compressing the locations of the points.
- 2) Attribute Compression: Aim at exploiting the redundancy in attribute values given the locations of the points.

TABLE I
COMPARISON OF MACHINE LEARNING BASED METHODS FOR PCC.

Paper	Feature	Technique	Dataset	Benchmark	Metrics	Loss Function	Optimizer
[18]	Geometry	CNN-based AE	MPEG	PCL	RD curve, D1	SGD (BCE)	Adam
[19]	Geometry	CNN-based AE	MPEG	MPEG GPCC	RD curve, D1	RD Loss	-
[20]	Geometry	3D CNN AE	Train: ModelNet40 Test: MVUB	MPEG GPCC	D1, D2	RD Loss	Adam
[21]	Geometry	3D Stacked CNN	Train: ShapeNet Test :MPEG, JPEG	MPEG GPCC, PCL MPEG VPCC	D1, D2	RD Loss WBCE	-
[22]	Geometry	FCNN, MLPs	ShapeNet	MPEG GPCC	D1	RD Loss Chamfer Dist	Adam
[23]	Geometry	Folding-based NN	ShapeNet ModelNet10	Fully connected decoder	-	Chamfer Dist	Adam
[24]	Geometry	RNN	TierIV	Octree, JPEG	SNNRMSE	-	-

Since ML approaches for PCC specifically pertaining to geometry compression [17] are relatively in their initial stages of development, a few recent contributions in this field form the main focus of this survey paper. The taxonomy of schemes covered in this paper is shown in Fig 1.

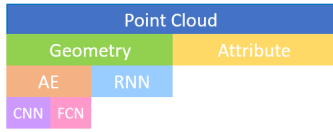


Fig. 1. Taxonomy of the proposed schemes in the literature.

The rest of the paper is organized as follows. Section II discusses the various CNN-based autoencoder implementations for PCC along with an evaluation of their merits and demerits. The use of fully connected convolutional neural network (FCNN) approach to PCC is presented in Section III, and the recurrent neural network (RNN) based PCC approaches are provided in Section IV. The use of AI in transmitting compressed PC data and potential advantages of using AI are discussed in Section V. Section VI summarizes and concludes the paper.

II. AUTOENCODERS

CNNs are known to be one of the most effective neural network (NN) in extracting useful features in uniform structure sharing a pattern/correlation. These characteristics are mostly to be found in images and video resulting in wide usage of CNNs. And hence, most recently proposed DL based PC geometry compression techniques adopt autoencoder (AE) neural network design involving convolutional layers. This section describes some of the recently proposed PC geometry coding schemes using CNN based autoencoders.

A. Point Cloud Geometry Autoencoder (PCG-AE)

PCG-AE proposes a basic implementation of DL approach for PCC to highlight the potential improvements of AE PC coding paradigm. The general 3D coordinate representation

of PC is changed to a set of binary 3D blocks format where occupied voxels are signalled by either a '1' or '0'. Since points in PC are generally arranged in disorderly fashion, coding them using NN can be very challenging. Thus this type of formatting introduces some correlation similar to images and video to be exploited by a CNN based AE in 3D domain. The underlying idea is to use an adaptive forward and inverse transform instead of using a fixed predefined transform such as DCT [18].

The details of the structure of transform is shown in Fig 2. First, the input PC geometry is divided into 3D binary voxelized blocks of size $32 \times 32 \times 32$ and the position of the 3D sub-blocks in original PC are sent as index in the final bitstream. Then AE is used to map PC geometry into the feature space and is addressed as latent representation as shown in Fig 2. So, the original 3D geometry is now represented by latents at the end of the encoding process with reduced dimensionality. These latents are then binarized using a certain threshold (for e.g. 0.5) to generate the bitstream.

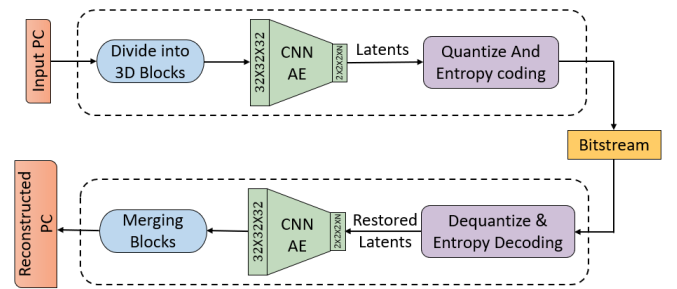


Fig. 2. PCG-AE coding solution.

At the decoder, the restored latent representation by decoding the incoming bit stream is then fed to the CNN AE with the aim of reconstructing the geometry as close as possible to the original geometry. The 3D blocks recovered the restored latents are then placed in their respective positions using the indices as side information to eventually reconstruct the entire PC.

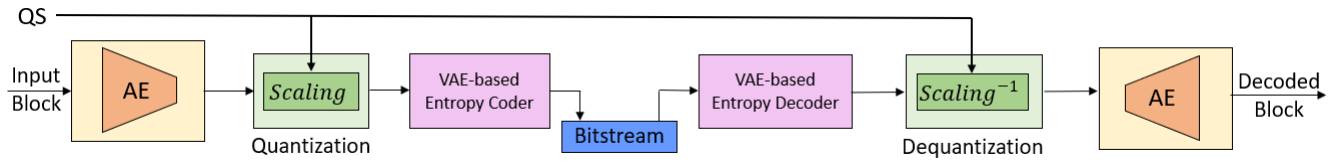


Fig. 3. DL-based PC coding architecture using implicit-explicit quantization.

Adaptive Transform: This implementation uses a very basic 3D CNN design which is an extension of the one typically used for 2D CNN image processing. It has 4 convolutional layers at the encoder and 4 at the decoder, each with width, depth and height of 3 and a stride of 2 to downsample and reduce the dimensionality of the latent representation. N represents the number of filters which is a hyperparameter and four models were trained for these N values: 32, 64, 96 and 128. N decides the reconstruction quality with the associated bitrate and hence affects the RD performance. N is tuned experimentally. All CNN layers use sigmoid as the activation function with stochastic gradient descent (SGD) as the convergence function and binary cross entropy (BCE) as the loss function to reconstruct the availability of voxels as accurately as possible. The decoder has a symmetric structure with downsampling replaced with upsampling by a factor of 2.

The training of the encoder and decoder is performed simultaneously. Point clouds from MPEG datasets were used for training and testing. Point cloud library (PCL) codec was set as the benchmark to evaluate the performance of the proposed scheme. RD performance with ratio of reconstructed points and original points were used as the performance metric. It was concluded that using more filters (N) achieves higher quality as expected. Although for lower bitrates, PCL performs better than PCG-AE, but in case of medium and higher bitrates, PCG-AE outperforms PCL by a significant margin.

B. RD Control Through Implicit and Explicit Quantization

In traditional transform-based coding methods, RD loss function (more specifically Lagrangian multiplier λ) defined in (1) is used to control the RD trade-offs in DL-based coding solutions.

$$Loss = dist + \lambda \times rate \quad (1)$$

where $dist$ is the distortion between the original and reconstructed PC, and $rate$ is the entropy of the quantized latent representation. The extension of PCG-AE has been very recently proposed to provide flexible RD control functionality [19]. Typical RD control can be achieved using either of the following:

- 1) Implicit quantization which employs tuning between the entropy of latents shown in Fig. 2 and the reconstruction error (controlled by λ).
- 2) Explicit quantization which employs a parameter quantization step (QS) to obtain different RD points.

Although the implicit quantization approach requires training of multiple DL models for each desired RD point, it performs substantially better than explicit quantization. The scheme in [2] suggests the use of the combination of implicit and explicit quantization, which reduces the training complexity and memory requirement to reach multiple RD points, when compared to implicit-only quantization coding. Therefore, RD trade-off is achieved using small number of trained DL models.

The overview of the scheme is depicted in Fig. 3, where it uses the loss function defined in Eq. (1) for training. Models trained for different values of λ allow us to achieve various RD points without using explicit quantization. A new PC will use these trained models for compression.

Similar to the approach mentioned in previous section, latent representation is entropy coded after quantization. But to have more efficient entropy coding, variational autoencoder (VAE) is used to adapt the entropy model to current latent data.

When the model shown in Fig. 3 is trained for $QS = 1$, it corresponds to implicit quantization even though we are explicitly specifying the amount of quantization desired. RD control can be achieved using higher value of QS ($QS > 1$ i.e., explicit quantization) to generate multiple RD points using the same DL model. The *Scaling* block down-scales the latent representation by QS , which is further entropy coded. At the decoder, the *Scaling*⁻¹ block up-scales the latent values by the same QS . And the entropy model adapts to a certain value of QS .

Now, to reach multiple RD points, single trained model can be used with

- 1) $QS = 1$ (implicit quantization)
- 2) $QS > 1$ (explicit quantization)

In addition, implicit-explicit balance is to be made since the model was initially trained for $QS = 1$ and higher QS may incorporate some distortion. This happens because there is a certain limit to the number of RD points generated using single trained DL model.

Performance Evaluation: In this approach, unlike PCG-AE, PCs were divided into blocks of size $64 \times 64 \times 64$. The QS value was varied from 1 to 20. And for implicit-only scheme, λ was varied from 50 to 20000. It was observed that single model was unable to reach lower bitrate (as QS increases) when trained for higher bitrate and was unable to reach higher bitrate (as QS decreases) when trained for lower bitrates. Hence, implicit-explicit quantization scheme was not able to cover wide range of quality and rates unlike multiple DL models trained using implicit-only quan-

tization. Therefore, to address this issue three models were used instead of a single model to cover wide range of bitrates. Implicit-explicit quantization with fairly fewer models ($\lambda = 100, 1000, 10000$) could achieve similar performance (for some cases slightly better) as that of using 10 trained models ($\lambda = 100, 250, 500, 600, 750, 1000, 2000, 3000, 10000, 20000$) with implicit-only quantization.

C. Learning Convolutional Transforms (LCT)

LCT approach uses 3D convolutional autoencoder which directly operates on voxels. The decoding is interpreted as a classification problem to predict/infer the occupancy of the voxel by defining the geometry as a binary signal across the voxel grid. The architecture of the method is depicted in Fig. 4, where f_a and f_s are the analysis and synthesis transform respectively. N specifies the number of filters (which is 32 in the proposed scheme). k^3 and s^3 specify the filter size and strides in each direction respectively [20].

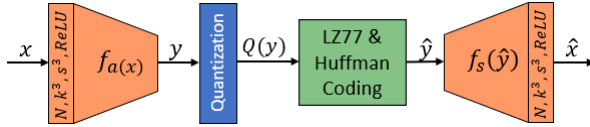


Fig. 4. LCT architecture (No. of filters(N), filter size(k), strides(s), activation function).

The latent values are represented as $y = f_a(x)$, which is further quantized using uniform quantization to obtain $\hat{y} = Q(y)$ and is decompressed to get $\hat{x} = f_s(\hat{y})$. f_a and f_s are learned during the training process. The encoder and the decoder use convolutions and transpose convolutions respectively with the same padding and stride of 2^3 . The k value is 9 for the first layer, and 5 for the second and last layer with no activation function.

The scheme does not use any entropy coder, but instead, uses a combination of LZ77 and Huffman coding to compress $Q(y)$. While decoding is considered as a classification problem, due to the sparsity of the point cloud, an imbalance is created between the filled and empty voxels. This issue is addressed using α -balanced focal loss. However, the final loss is the same as defined in Eq. (1).

The results showed that the model was able to generalize well even though it was trained using ModelNet40 mesh dataset (to cover a wide range of variety and quantity) and tested on completely different dataset (MVUB-Microsoft Voxelized Upper Bodies dataset [4]). The proposed scheme

outperformed MPEG anchor (described in [2], MPEG GPCC for static PCs and MPEG VPCC for video based PCs) for all the sequences and at all bitrates. For the similar bitrates, it achieves lower distortion and reconstructs more points compared to MPEG anchor, resulting in better quality. Different λ values ($10^{-4}, 5 \times 10^{-5}, 10^{-5}, 5 \times 10^{-6}, 10^{-6}$) were used to generate RD points and different octree depths for the MPEG anchor. It improved the Bjontegaard-delta bitrate (BDBR) by 51.5% on average for the testing data.

D. Learned Point Cloud Geometry Compression (LPCGC)

The LCT method was followed by other new approaches in the literature. One of them was Learned PCGC that uses DNN-based variational autoencoder. The overview of the scheme is illustrated in Fig. 5.

Preprocessing: This module involves three steps: voxelization, scaling and partition. *Voxelization* is an optional step and is skipped if PC is already in 3D volumetric presentation. Cartesian coordinate system is used to perform voxelization, where a voxel is set to 1 if it is occupied and 0 otherwise. Inter-voxel correlation which can be exploited by 3D convolutions to generate very compact latent representation of a 3D block [21].

Similar to image down-sampling used in image and video compression, down-sampling was also used in PCC in order to maintain the quality at lower bit rates. *Scaling* is incorporated to reduce the sparsity and hence increase the density of the PC. Doing so was found to improve the efficiency by a considerable amount, especially for sparser PC such as dynamically acquired point clouds (category 3) of MPEG dataset. The scaling is performed as follows:

$$\hat{X}_n = \text{round}(X_n \times s), \quad (2)$$

where $(X_n \times s) = (i_n \times s, j_n \times s, k_n \times s)$, and X_n is the original PC geometry for $n = 1, \dots, N$. X_n is scaled by s where $s < 1$. Then the PC is partitioned into 3D non-overlapping cubes of size $W \times W \times W$. The respective position of the valid cube (at least one occupied voxel) with the number of occupied voxels in each cube is signaled explicitly, thereby adding a very small amount of overhead (metadata).

The 3D cubes are fed to the analysis transform one-by-one, which consists of 3D stacked CNNs, to generate a compact latent representation of size (*channel, length, width, height*) defined as $y = f_e(x; \theta_e)$, where θ_e is the set of convolutional weights. For decoding, the synthesis transform is so designed

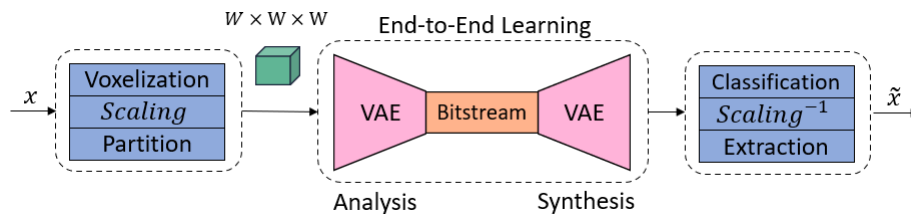


Fig. 5. Overview of learned PCGC.

that quantized latent values \hat{y} are used to reconstruct the cubes, formulated as $\tilde{x} = f_d(\hat{y}; \phi_d)$, where ϕ_d is the set of parameters.

The basic 3D convolutional unit in the transforms used is Voxelation-Resnet (VRN) because of its efficiency obtained due to the residual and inception network [25].

The scheme uses fairly smaller kernels ($3 \times 3 \times 3$), offering lower complexity. Hyperprior is used for entropy modeling similar to the LCT approach.

Post-processing: It comprises of classification, inverse scaling and extraction. Even though the median value of threshold is 0.5, it is shown to be not-optimal and hence, the *Classification* step uses adaptive thresholds to infer whether the voxels in the reconstructed cube \tilde{x} are present or not. The classification of the voxels in the cube is also influenced by the number of occupied points in the original cube. This information is sent as the metadata during encoding. *Inverse scaling* is implemented as the next step, which factors the reconstructed geometry by $1/s$ for rendering and display. *Extraction* is an optional step that changes the volumetric representation to raw file format such as polygon file format (.ply) or ASCII.

For model generalization, the scheme used ShapeNet as the training dataset, and MPEG and JPEG pleno standardization datasets were used for testing. The models were trained from end-to-end for each bitrate by changing λ (Lagrangian) and scaling factor s . Weighted binary cross entropy loss (WBCE) was used while training whereas adaptive thresholding was used for inference. Simulation results show that learned PCGC outperforms GPCC (octree) by 77% & 69%, GPCC (trisoop) by 67% & 62%, and it offers 88% & 82% gains against PCL for D1 (point to point) and D2 (point to plane), respectively.

III. FULLY CONNECTED NEURAL NETWORK (FCNN)

CNNs not only retain the spatial relations in images and videos but also have the advantage of weight sharing, which significantly reduces the complexity of ML solutions. Therefore, CNNs have become the most common choice for solving vision-based tasks for their practical application, especially for advanced visual data representations such as PC and meshes [4]. Nonetheless, there are few approaches that directly work on the geometry coordinate of PC to achieve compression. Two of those schemes are discussed in this section and compared against the previously summarized 3D CNN based AE solutions.

A. Deep AE-based PCGC

This scheme focuses on design of a codec that adapts to the features of PC for improved compression efficiency. The

architecture of the proposed network is shown in Fig. 6. It comprises of four main modules: a pointnet based encoder that acts as an analysis transform [26], a uniform quantizer, an entropy estimation block in the middle, and the decoder that acts as a non-linear synthesis transform. The geometry of a PC is represented as 3D points consisting of n points. These n points are downsampled first, to reduce the sparsity of the original PC. Sampling layer will result in duplicate points which are merged together resulting in fewer points (m), which are the subset of original sampled n points [22].

Next, the m points are fed to the encoder, which comprises of five 1D convolutional layers with kernel size of 1, followed by a *ReLU* and batch normalization layer. The filters in each layer is 64, 128, 128, 256 and k , respectively. Here k is decided upon the number of input points. Then there is a max-pooling layer used to produce k -dimensional latent code z . The latent code is quantized and encoded by the entropy coder.

The decoder uses a FCN, which decodes the latent code z using 3 fully-connected layers with hidden layers of 256, 256 and $n \times 3$ neurons to generate $n \times 3$ reconstructed PC. This whole process can be summarized using the following equation:

$$z = D(Q(E(S(x)))), \quad (3)$$

where S , E , Q and D are sampling layer, encoder, quantization and decoder, respectively. Analysis and synthesis transform are formulated as $z = f_e(x; \theta_e)$ and $\hat{x} = f_d(y; \phi_d)$, respectively, where x is the original PC, z is the compressive representation, and \hat{x} is the reconstructed PC. θ_e & ϕ_d are the trainable parameters.

Quantization using rounding function makes the derivative 0 or undefined, rendering the loss non-differentiable. Therefore, the proposed scheme replaces quantization by an additive uniform noise.

$$[f(x)] \approx f(x) + u \quad (4)$$

where u is the random noise. The distortion is computed using Chamfer Distance defined as follows:

$$d_{CH}(S_1, S_2) = \sum_{x \in S_1} \min_{y \in S_2} \|x - y\|_2^2 + \sum_{y \in S_2} \min_{x \in S_1} \|x - y\|_2^2 \quad (5)$$

where S_1 is the original PC with n points and S_2 is the reconstructed PC with m points. The proposed scheme uses ShapeNet as the training and testing set from a single class with 90% and 10% split. The results are compared with MPEG

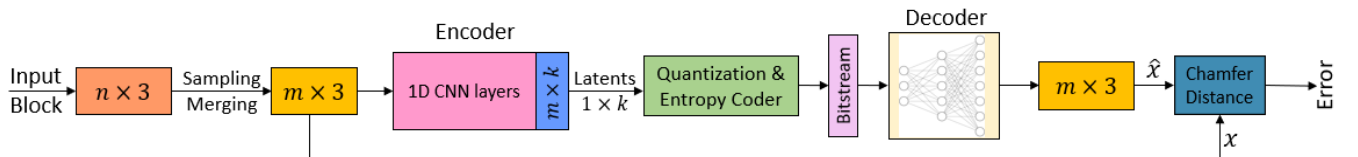


Fig. 6. Deep AE-based architecture.

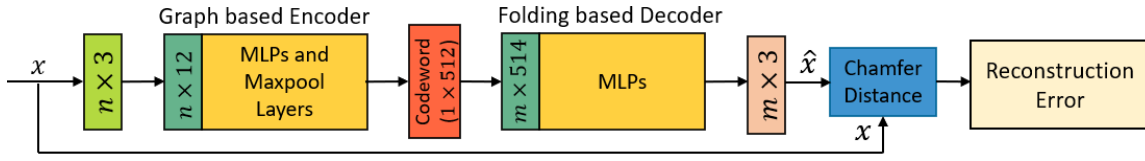


Fig. 7. FoldingNet architecture overview.

anchor (TMC13). The model uses data from four categories: chair (train: 6101, test: 678), airplane (train: 3640, test: 405), table (train: 7658, test: 851) and car (train: 6747, test: 750). Deep AE-based PCGC achieves BD rate gain of 73.15% on average over MPEG GPCC.

B. FoldingNet: PC AE via Deep Grid Deformation

A CNN requires its neighbors at a certain distance to share fixed spatial relation. Since PC does not exhibit such characteristics in raw format, voxelization is performed to make convolution operation on PC more meaningful. But voxelization sacrifices the details of the original representation of the PC. There are very few approaches that work on the original PC without voxelization. One of them is FoldingNet recently proposed. It is a MLP based AE, which uses a 2D grid structure to construct a PC through Folding operation [23].

Fig. 7 shows the FoldingNet architecture. The encoder comprises of MLP and graph based max-pooling layers. The geometry of the PC $n \times 3$ (where n is the number of points) is the input to the encoder. Then the local covariance matrix of size 3×3 for each point v is computed using the coordinates, which are one hop neighbor of v and vectorized to 1×9 . The covariances ($n \times 9$) and geometry ($n \times 3$) are concatenated to a matrix of size $n \times 12$ and fed to the 3 layer perceptron. The output of the perceptron is fed to two consecutive graph layers to eventually generate a codeword (1×512), which maps a high-dimensional embedding of an input PC. The codeword is replicated m times and concatenated with fixed 2D grid points of size $m \times 2$ to generate a matrix of $m \times 514$. This matrix is the input to the 3-layer perceptron and generates the output of the first folding operation ($m \times 3$). Then again, it is concatenated with replicated codewords and becomes the input to the second folding operation, which results in the reconstructed PC. Two folding operation can be perceived as a 2D to 3D transformation/mapping that produces elaborating surfaces. The first one folds the 2D grid to 3D space and the second one folds inside the 3D space. The number of output points m is not necessarily the same as n . The distortion error between the input and the reconstructed PC is calculated using the Chamfer distance defined in previous section.

During training, the model starts with a random folding but eventually turns into a meaningful PC. The model was trained using 16 categories of ShapeNet dataset. One major advantage of Folding based decoder shown in Fig. 7 is that it only uses 7% parameters of a decoder compared to as that of a FCNN decoder and it is more focused towards solving the classification problem rather than compressing the PC. However, the idea could be adapted for PCC to validate the

performance in comparison to international standardization. As far as classification problem is considered, the codewords generated at the end of the encoder are used to classify using SVM.

The folding-based decoder was compared against the FC decoder. During the training process, the reconstruction loss (Chamfer Distance) kept decreasing, which implies the reconstructed PC is becoming more and more similar to the original PC. And the classification accuracy of the linear SVM which was trained on the codewords improved, implying that the codeword becomes more linearly separable while training. For validation, the ShapeNet trained model was used for transfer learning and tested on ModelNet dataset that achieved an 88.4% classification accuracy.

IV. RNN

There are very limited ML approaches that use RNN for point cloud compression. One of them is discussed in this section that focuses on data from 3D LiDAR sensors. The conventional methods that are part of the standardization are not very successful in reducing the point cloud to a very low volume, especially 3D LiDAR sequences. Therefore, some approaches operate on the raw packet data from a 3D LiDAR, which is shown to be highly cost effective approach for compression. Although it is possible to transform raw packet data into 2D matrix losslessly, they are very irregular in nature and does not share any spatial correlation, if taken without calibrations. The advantage of doing so is reduced sparsity of the PC which is helpful for the further processing [24].

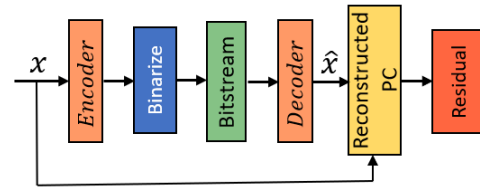


Fig. 8. Residual calculation.

The proposed scheme targets static PC and introduces a RNN based compression method with residual blocks. First, 2D matrices are constructed by transforming the frames of raw packet data. 2D-matrix normalization is then performed as a pre-processing step before the data is fed to the network comprising of an encoder, a binarizer and a decoder. As 2D format can have a much larger range of possible values due to the accuracy and long range LiDAR detection scanners, the scheme uses a few steps for normalization: The packet data

are randomly chosen from 100 frames to calculate the mean μ and the histogram of distance values are created and the data matrix R is normalized as follows: $(R - \mu)/\theta$ where threshold θ is set such that 95% of the distances fall under this value.

The encoder, binarizer and decoder in Fig. 8 each comprise of a combination of convolutional layers and convolutional LSTM layers. The decoder also uses residual blocks because getting an accurate decompression can be challenging while using such a network. Specifically, the decoder iteratively uses incoming binary file to reconstruct the original input and the residual is computed by comparing the reconstructed data with the original PC. This residual becomes the input for next iteration. This process is repeated until the decompressed data is reasonably accurate. The process of generating this residue is also shown in Fig. 8. The decompression loss is computed using SNNRMSE (symmetric neighbor root mean squared error) [24]. The scheme is tested on driving data from 11 areas of Japan and is shown to outperform the octree compression and JPEG compression benchmark methods.

V. AI-BASED PC TRANSMISSION

Transmission of PC video data over resource constrained channels is a challenge. Moreover, real-time transmission of PC video data using the existing bandwidth of available networks is difficult. Hence the availability of a compression scheme for PC data is very useful to accomplish such PC video streaming applications.

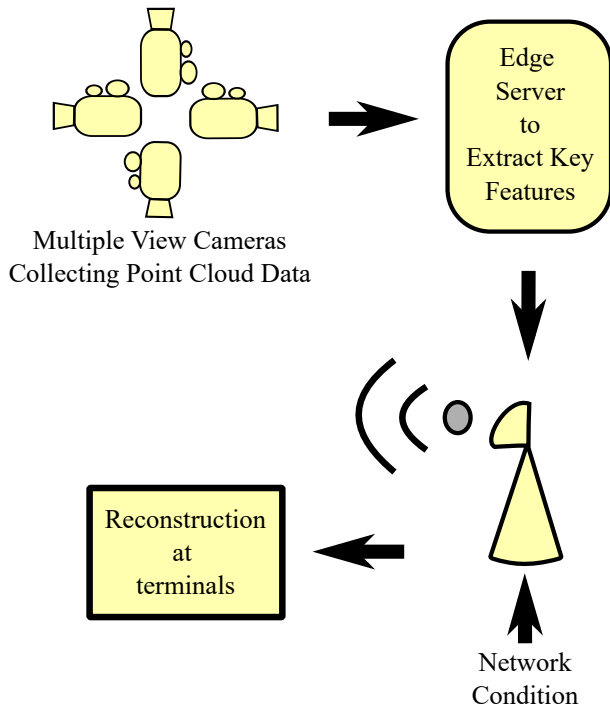


Fig. 9. Components of AITransfer system.

As mentioned previously, the use of AI/ML to assist in compression of PC data yields promising results. Likewise, the use of AI/ML in assisting with the transmission of PC

video data is discussed in this section. In [14], an AI enabled technique called AITransfer is proposed. This method uses an AI-powered adaptive transmission technique, which is bandwidth-aware. The reduction in bandwidth consumption and the alleviation of computations is achieved by only extracting and transferring the key features from the PC. Thereby, as mentioned in [14], one of the benefits of using the AITransfer technique is the incorporation of dynamic network bandwidth in the design of an end-to-end architecture using fundamental concepts like feature extraction and reconstruction. Another contribution of AITransfer is the use of online adapter to sense the network bandwidth for a matching optimal inference model.

Fig. 9 shows the architecture of AITransfer highlighting its components and system flow obtained from [14]. As can be seen in Fig. 9, this technique extracts key features from the PC data collected by multiple view cameras using an AI-based neural network. Extracting just these features compresses the PC data and makes it more compatible to the networks bandwidth handling capacity. The feature extracted data are then transmitted using the existing network and reconstructed at the intended terminals. The feature extraction steps are presented in Fig. 10.

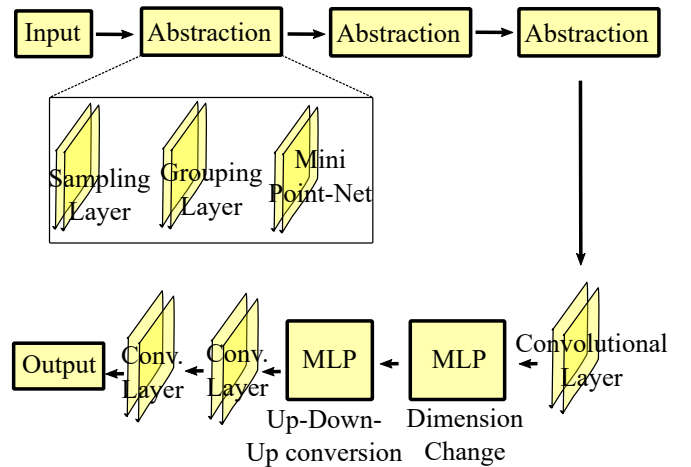


Fig. 10. Design of the network architecture

From Fig. 10, it can be seen that the various layers used in the AI-based network for extracting key features have a few abstraction steps, followed by convolutional layers, dimensionality changes and more convolutional layers, which yields the compressed PC output. These data are then transmitted according to the network conditions and subsequently reconstructed at the terminals like cell-phones, VR headsets, etc. [14] presents a case study to evaluate the performance of AITransfer and shows that a compression ratio of almost over 30 to 1 can be attained.

VI. CONCLUSION

This paper presents a survey on an emerging research area of point cloud compression (PCC) using deep learning/machine learning techniques. Due to early stage of development in this

area, a summary of only 8 research articles was presented in this survey. In addition to the summary of the most recent developments in this area, we present a comparative performance study of the surveyed techniques.

The majority of PCC techniques using DL/ML proposed in the literature are related to the geometry compression, with very few techniques focused on attribute compression of PC and video compression. It was also seen that most of the approaches use convolutional based auto-encoders, multi-layer perceptrons and fully connected neural network decoders to achieve PCC. Most recently, the use of RNN based techniques to perform PC compression was also studied and one such method is also included in this survey.

Transmission of PC or compressed PC data is also challenging, hence making it an emerging topic area. A most recent simultaneous AI based compression and transmission technique called AITransfer is also briefly discussed in this survey.

REFERENCES

- [1] C. Cao, M. Preda, and T. Zaharia, "3d point cloud compression: A survey," 07 2019, pp. 1–9.
- [2] D. Graziosi, O. Nakagami, S. Kuma, A. Zaghetto, T. Suzuki, and A. Tabatabai, "An overview of ongoing point cloud compression standardization activities: video-based (v-pcc) and geometry-based (g-pcc)," *APSIPA Transactions on Signal and Information Processing*, vol. 9, p. e13, 2020.
- [3] D. T. Nguyen, M. Quach, G. Valenzise, and P. Duhamel, "Learning-based lossless compression of 3d point cloud geometry," in *IEEE International Conference on Acoustics, Speech, and Signal Processing*, Toronto, Canada, Jun. 2021. [Online]. Available: <https://hal.archives-ouvertes.fr/hal-03141577>
- [4] C. Loop, Q. Cai, S. O. Escolano, and P. A. Chou, "Microsoft Voxelized Upper Bodies – A Voxelized Point Cloud Dataset," *ISO/IEC JTC1/SC29 Joint WG11/WG1 (MPEG/JPEG) input document m38673/M72012*, May 2016. [Online]. Available: <http://plenodb.jpeg.org/pc/microsoft/>
- [5] R. Hooda and W. D. Pan, "Early termination of dyadic region-adaptive hierarchical transform for efficient attribute compression of 3d point clouds," *IEEE Signal Processing Letters*, 2021.
- [6] L. Wiesmann, A. Milioto, X. Chen, C. Stachniss, and J. Behley, "Deep compression for dense point cloud maps," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 2060–2067, 2021.
- [7] R. Hooda and W. D. Pan, "Tree based search algorithm for binary image compression," in *2019 SoutheastCon*, 2019, pp. 1–6.
- [8] Z. Liu, Q. Li, X. Chen, C. Wu, S. Ishihara, J. Li, and Y. Ji, "Point cloud video streaming: Challenges and solutions," *IEEE Network*, vol. 35, no. 5, pp. 202–209, 2021.
- [9] R. Hooda and W. D. Pan, "Lossless compression of bilevel roi maps of hyperspectral images using optimization algorithms," *IEEE Geoscience and Remote Sensing Letters*, vol. 19, pp. 1–5, 2022.
- [10] M. Quach, G. Valenzise, and F. Dufaux, "Improved deep point cloud geometry compression," 2020.
- [11] X. Wen, X. Wang, J. Hou, L. Ma, Y. Zhou, and J. Jiang, "Lossy geometry compression of 3d point cloud data via an adaptive octree-guided network," in *2020 IEEE International Conference on Multimedia and Expo (ICME)*, 2020, pp. 1–6.
- [12] P. A. Chou, M. Koroteev, and M. Krivokuća, "A volumetric approach to point cloud compression—part i: Attribute compression," *IEEE Transactions on Image Processing*, vol. 29, pp. 2203–2216, 2020.
- [13] T. Chen, H. Liu, Q. Shen, T. Yue, X. Cao, and Z. Ma, "Deepcoder: A deep neural network based video compression," in *2017 IEEE Visual Communications and Image Processing (VCIP)*, 2017, pp. 1–4.
- [14] Y. Huang, Y. Zhu, X. Qiao, Z. Tan, and B. Bai, "Aittransfer: Progressive ai-powered transmission for real-time point cloud video streaming," in *Proc. of the 29th ACM Int. Conf. on Multimedia*. New York, NY, USA: Association for Computing Machinery, 2021, p. 3989–3997. [Online]. Available: <https://doi.org/10.1145/3474085.3475624>
- [15] W. Jia, L. Li, A. Akhtar, Z. Li, and S. Liu, "Convolutional neural network-based occupancy map accuracy improvement for video-based point cloud compression," *IEEE Transactions on Multimedia*, pp. 1–1, 2021.
- [16] E. Alexiou, K. Tung, and T. Ebrahimi, "Towards neural network approaches for point cloud compression," in *Applications of Digital Image Processing XLIII*, A. G. Tescher and T. Ebrahimi, Eds., vol. 11510, International Society for Optics and Photonics. SPIE, 2020, pp. 18 – 37. [Online]. Available: <https://doi.org/10.1117/12.2569115>
- [17] A. F. R. Guarda, N. M. M. Rodrigues, and F. Pereira, "Adaptive deep learning-based point cloud geometry coding," *IEEE Journal of Selected Topics in Signal Processing*, vol. 15, no. 2, pp. 415–430, 2021.
- [18] A. F. R. Guarda, N. M. M. Rodrigues, and F. Pereira, "Point cloud coding: Adopting a deep learning-based approach," in *2019 Picture Coding Symposium (PCS)*, 2019, pp. 1–5.
- [19] A. F. R. Guarda, N. M. M. Rodrigues, and F. Pereira, "Deep learning-based point cloud geometry coding: Rd control through implicit and explicit quantization," in *2020 IEEE International Conference on Multimedia Expo Workshops (ICMEW)*, 2020, pp. 1–6.
- [20] M. Quach, G. Valenzise, and F. Dufaux, "Learning convolutional transforms for lossy point cloud geometry compression," *CoRR*, vol. abs/1903.08548, 2019. [Online]. Available: <http://arxiv.org/abs/1903.08548>
- [21] J. Wang, H. Zhu, Z. Ma, T. Chen, H. Liu, and Q. Shen, "Learned point cloud geometry compression," 2019. [Online]. Available: <https://arxiv.org/abs/1909.12037>
- [22] W. Yan, Y. Shao, S. Liu, T. H. Li, Z. Li, and G. Li, "Deep autoencoder-based lossy geometry compression for point clouds," *CoRR*, vol. abs/1905.03691, 2019. [Online]. Available: <http://arxiv.org/abs/1905.03691>
- [23] Y. Yang, C. Feng, Y. Shen, and D. Tian, "Foldingnet: Interpretable unsupervised learning on 3d point clouds," *CoRR*, vol. abs/1712.07262, 2017. [Online]. Available: <http://arxiv.org/abs/1712.07262>
- [24] C. Tu, E. Takeuchi, A. Carballo, and K. Takeda, "Point cloud compression for 3d lidar sensor using recurrent neural network with residual blocks," in *2019 International Conference on Robotics and Automation (ICRA)*, 2019, pp. 3274–3280.
- [25] Y. Zhou and O. Tuzel, "Voxelnet: End-to-end learning for point cloud based 3d object detection," *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4490–4499, 2018.
- [26] R. Q. Charles, H. Su, M. Kaichun, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 77–85.