sensors

MDPI

Article

An Optimized Convolutional Neural Network for the 3D Point-Cloud Compression

Guoliang Luo [1], Bingqin He [1], Yanbo Xiong [1], Luqi Wang [1], Hui Wang [1], Zhiliang Zhu [1,*] and Xiangren Shi [2]

[1] Virtual Reality and Interactive Techniques Institute, East China Jiaotong University, Nanchang 330013, China
[2] School of Informatics, Xiamen University, Xiamen 361005, China
* Correspondence: rj_zzl@ecjtu.edu.cn

Abstract: Due to the tremendous volume taken by the 3D point-cloud models, knowing how to achieve the balance between a high compression ratio, a low distortion rate, and computing cost in point-cloud compression is a significant issue in the field of virtual reality (VR). Convolutional neural networks have been used in numerous point-cloud compression research approaches during the past few years in an effort to progress the research state. In this work, we have evaluated the effects of different network parameters, including neural network depth, stride, and activation function on point-cloud compression, resulting in an optimized convolutional neural network for compression. We first have analyzed earlier research on point-cloud compression based on convolutional neural networks before designing our own convolutional neural network. Then, we have modified our model parameters using the experimental data to further enhance the effect of point-cloud compression. Based on the experimental results, we have found that the neural network with the 4 layers and 2 strides parameter configuration using the *Sigmoid* activation function outperforms the default configuration by 208% in terms of the compression-distortion rate. The experimental results show that our findings are effective and universal and make a great contribution to the research of point-cloud compression using convolutional neural networks.

Keywords: point-cloud compression; convolutional neural network; activation function

Citation: Luo, G.; He, B.; Xiong, Y.; Wang, L.; Wang, H.; Zhu Z.; Shi, X. An Optimized Convolutional Neural Network for the 3D Point-Cloud Compression. *Sensors* **2023**, *23*, 2250. https://doi.org/10.3390/s23042250

Academic Editors: Baochang Zhang and Ying Huang

Received: 16 January 2023
Revised: 9 February 2023
Accepted: 14 February 2023
Published: 16 February 2023

1. Introduction

A 3D point-cloud model is a collection of 3D data points in 3D space that represent a 3D shape or object. 3D point-cloud compression technology is also known as 6-dimensional data compression technology since point-cloud data are composed of geometric coordinates (X, Y, and Z) as well as RGB or other three color descriptors. Point-cloud data comes from a computer-aided design model, radar scanning, and depth camera acquisition. For example, Popiștер et al. studied the point-cloud acquisition process with the help of a contact/non-contact 3D scanner and storage of point cloud data [1]. At present, there rarely exists such large bandwidth to support the direct transmission of point clouds on the network layer without compression. Therefore, it is necessary to compress point clouds. In the spatial domain, the point-cloud model typically contains a few hundreds and thousands to tens of millions of points. Without compression, the transmission rate of 30 frames per second for a point-cloud model with one million points per frame consumes a total bandwidth of 3.6 GB per second, placing strain on the available storage space and network transmission bandwidth. Consequently, attaining a low bit rate and low distortion point-cloud compression with limited storage space capacity and network transmission bandwidth remains a critical practical role in achieving efficient visualization on VR applications.

For the 3D point-cloud compression, the traditional octree-based point-cloud compression method in which the deeper depth will lead to exponential growth of data volume and the shallow depth will lose a lot of information. With the rapid development of convolutional neural networks, researchers proposed the method of using convolutional

neural network to compress point clouds. For example, some researchers replaced the 2D-convolution in image compression with the 3D-convolution adapted to point clouds and converted the original point cloud into a voxel grid. This method led to a better compression result but had problems of low compression rate and high distortion rate. Therefore, we have conducted a number of experiments such as choosing the appropriate network parameters and activation functions to optimize the effect of compressing point clouds.

Convolutional neural networks require a large amount of computation and storage requirements. Thereby, knowing how to reduce computational costs and storage space is the focus of research on convolutional neural networks. Zhao et al. improved the performance and efficiency of 1-bit CNNs by combining Bayesian learning in CNN species 1-bit CNNs [2]. The computational redundancy of the complete model can be reduced within the allowed accuracy by pruning the neural network. Li et al. proposed the EagleEye pruning algorithm, which improved the efficiency and accuracy of pruning by the adaptive batch normalization of the network [3]. Zhang developed a new deep learning model: MCNs to reduce the storage cost of convolutional filters [4]. Similarly, Yeom et al. segmented a large number of experimental data using a SegNet-based CNN, greatly reducing the human time required to analyze the data when training the model [5]. Qayyum et al. evaluated each pre-trained convolutional neural network model for image classification processing, which provided a basis for reducing human subjectivity and time-consuming redundancy [6]. Due to improve the performance of related deep learning models, Wang et al. classified point-cloud data by focusing on the relationship between points and its neighbors, which achieved the purpose of capturing high-resolution or fine-grained features [7]. Liu et al. proposed a point-voxel CNN for 3D deep learning by pooling the advantages of voxels and points, which effectively improved the execution efficiency and effectiveness of convolution [8]. Xu et al. developed a deep learning model for point-cloud processing. By using the POEM method to build a 1-bit fully connected layer (Bi-FC) in point-cloud networks, the storage and computing costs of point-cloud data are effectively reduced [9]. Our work also benefits from these studies on increasing the performance of neural networks with the help of several activation functions. Specifically, Alkhouly et al. conducted comprehensive research on and the classification of common activation functions, which provided an in-depth analysis of the effects of several common activation functions on deep network architectures [10]. Liu et al. proposed an extended GLIT method for asymmetric activation functions, which expanded the range of activation functions and provided the possibility for further research on neural networks [11]. Kumar et al. explored a NewSigmoid function in neural networks, which is also as powerful as *tansig* and *logsig* [12]. Siegel et al. advanced the existing results on the approximation rates obtained for two layer neural networks with an increasing number of neurons [13]. These results were extended to polynomially decaying activation functions and to general bounded ones.

In this work, we have investigated the various effects of the neural network's parameters, including stride, depth, and activation function on the impact of point-cloud compression. Based on the experimental results, we have designed an optimized scheme to maximize the compression performance. We first design the experimental scheme based on the convolutional neural network and then conduct experiments and modify our model parameters based on the experimental data to further optimize the point-cloud compression effect. Finally, we present a convolutional neural network that shows improved performance on the compression of 3D point cloud data.

This work contains the following major contributions:

- First, we have evaluated the compression effect by altering the parameters of depth, stride, and activation function of the neural networks. The experimental results show that the *Sigmoid* function outperforms the other activation functions.
- Second, we have proposed an optimized point-cloud compression scheme to enhance the effectiveness of the point-cloud compression.

The structure of the paper is organized as follows. We first review the state of point-cloud compression based on the convolutional neural network in Section 2. Then, we describe our working methods in detail in Section 3, followed by the results and the discussions in Section 4. Finally, we conclude the work in Section 5.

2. Related Work

The goal of point-cloud compression is to achieve a balance between high compression ratio, low distortion rate, and computing cost. The MPEG-3DG working group divided point-cloud compression standards into video-based point-cloud compression (V-PCC) and geometry-based point-cloud compression (G-PCC) according to processing methods. V-PCC aims to provide low-complexity decoding capability for applications requiring real-time decoding, such as virtual/augmented reality and immersive communication [14]. G-PCC is believed to provide efficient lossless and lossy compression for the deployment of autonomous driving, 3D maps, and other applications using point clouds generated by radar [15].

Aiming at the traditional point-cloud compression method based on octree, Diogo et al. proposed geometric octree coding of a point cloud based on an intra frame context [16,17]. The octree structure provides better context for entropy coding and improves the average rate. An algorithm for motion estimation and compensation was presented by Thanou et al. [18]. It is utilized to eliminate the temporal redundancy from the predictive coding of 3D positions and the color properties of point-cloud sequences. On the basis of this, Quach et al. suggested a novel motion-compensated approach to encoding dynamic voxelized point clouds at low bit rates where both the geometry and the color are encoded with distortion, allowing for reduced bit-rates [19]. Mekur et al. implemented point-cloud coding and decoding by dividing the octree voxel space into multiple macroblocks that are progressively subdivided to achieve intra-frame coding, mainly applied for 3D immersive video for general purpose and real-time time variation to improve point cloud acquisition rate to a large extent [20]. However, the octree-based compression technique tends to create "blocky" outputs at low to medium bit rate rendering stages and causes an exponential drop in the number of point clouds when the depth of the tree is decreased.

In recent years, due to the development and breakthrough of deep learning in the field of image research, Wu et al. represented a geometric 3D shape as a probability distribution of binary variables on a 3D voxel grid, using a convolutional deep neural network [21]. Valenzise et al. found that compression approaches based on deep auto-encoders can achieve coding performance higher than JPEG 2000 [22]. Huang et al. proposed a 3D point-cloud geometric compression method based on deep learning, which was an auto-encoder that also supported the parallel compression of multiple models via the GPU, considerably increasing processing efficiency [23]. When compared to PCL compression [24] and Draco compression [25], this method retains the original shape with little loss. To learn how to represent disordered point clouds, Panos et al. constructed an auto-encoder using convolutional layers and fully linked layers [26]. They compared the generative capacities of several distinct autoencoder-based GANs that were trained as generative networks [27]. The approach of compressing a point cloud using convolution was improved by Ball et al., who also suggested utilizing additive uniform noise in place of quantization during training and doing actual quantization during evaluation [28]. Inspired by this, Maurice et al. proposed a convolution transformation learning method for lossy point cloud geometric compression [29]. They also directly learned filters from data, taking into account quantization and rate-distortion (RD) in training. In contrast to the octree-based method, this method's model has strong universality and does not exhibit an exponential decline in the number of points when the bit rate is decreased.

In addition, deep learning-based convolutional neural networks showed promising performance in medical detection, GPS, and risk assessment [30–32]. Recent research on binary neural networks (BNNs) has also been successively published. Liu et al. introduced a reshaped point-wise convolution (RPC) to replace the conventional one to build binary

neural networks (BNNs) [33]. On the other hand, Xu et al. first attempted to optimize BNNs from a bilinear perspective, which improved the learning process of BNNs by correlating intrinsic bilinear variables during backward propagation [34]. Similarly, Jin et al. introduced a deep-walk strategy into graph convolutional networks (GCNs) to efficiently explore the global graph information, which enabled the more efficient extraction of potential representations of graph structure data [35]. Moreover, Zeng et al. proposed a face neural volume rendering (FNeVR) network and lightweight pose editor to enhance the facial details for image rendering [36].

Combined with the above research, our optimized convolutional neural network framework greatly improves the quality of point-cloud compression, and the work is highly scalable, which is an important reference for related work in this area.

3. Methods

Figure 1 shows flow chart of CNN-based 3D point cloud compression scheme. The remainder of the section is organized as follows. We first specify the model-training process in Section 3.1. Then, we describe the experimental process of obtaining the outputs of our network in Section 3.2. Finally, We explain the loss-calculation process in Section 3.3.

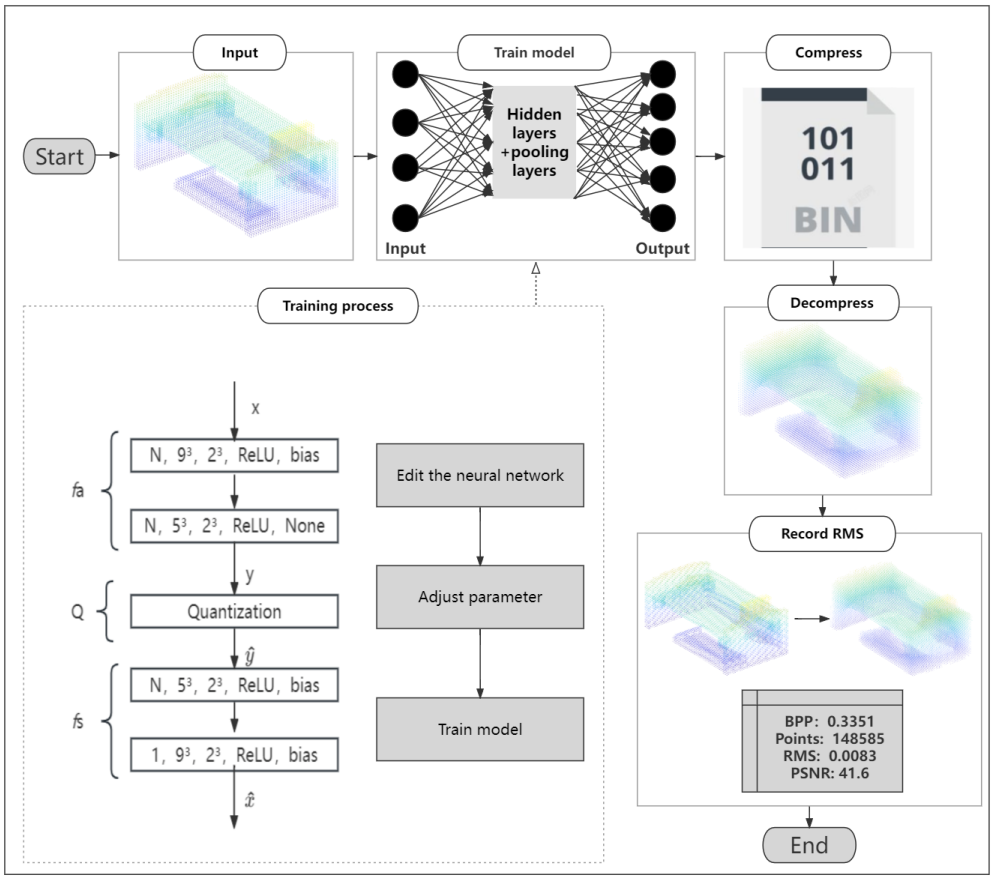

Figure 1. Flow chart of CNN-based 3D point-cloud compression scheme. First, we input the point-cloud model and then use the trained model to obtain the compressed binary file. Then, we decompress the compressed file to compare the decompressed point-cloud model with the one before decompression. Finally, we evaluate the performance by recording the RMS and other parameters between them. The steps of training model are as follows (dash line): First, edit the neural network mesh and adjust convolutional neural network depth, stride, and activation function. Second, adjust the compression accuracy and the number of training rounds. Third, observe model training index total loss and total quantities. If the training requirements are not fulfilled, the model will be retrained until it meets the requirements.

3.1. Model Training

3.1.1. Neural Network Structure

The first step in this work is to design the initial structure of the convolutional neural network. In our method, x represents the input original point cloud, $\hat{x}$ represents the output decompressed point cloud, and f_a represents the analysis transformation. Furthermore, $y = f_a(x)$, Q is the quantification function, $y = Q(y)$, f_s is the synthesis transformation, $\hat{x} = f_s(\hat{y})$. We use N to represent the filter, and s^3 and p^3 to represent padding and stride. For example, 9^3 and 2^3 represent $9 \times 9 \times 9$ padding, and the filter stride is $2 \times 2 \times 2$.

We use 2 layers, 3 layers, 4 layers with 2 strides and 2 layers, and 3 layers with 3 strides for the test. The architecture of the used neural network is shown in Figures 2 and 3. The minimal compression requirements cannot be achieved by 4 layers with 3 strides, and thus it is not taken into account in this case.

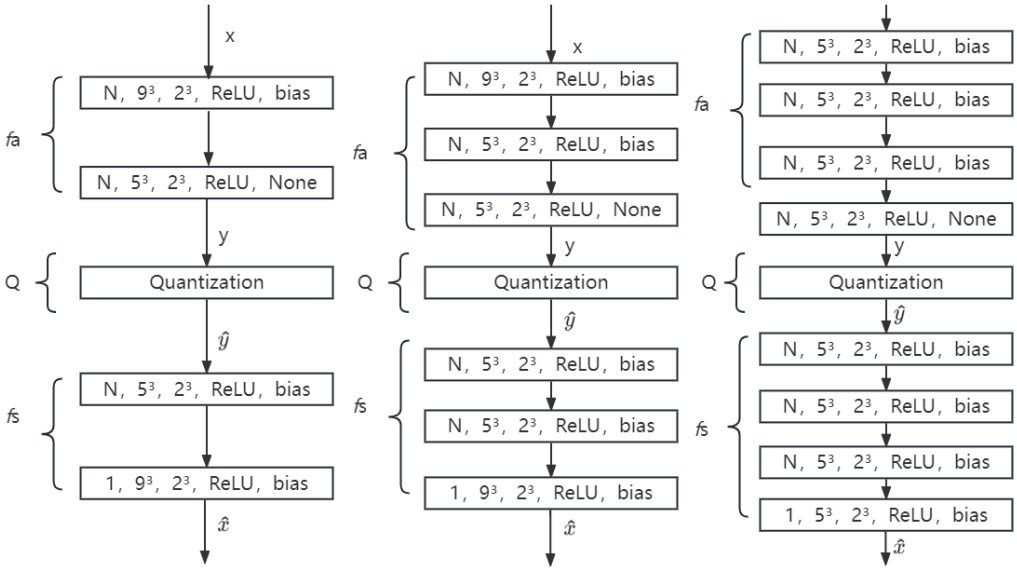

Figure 2. The architecture of the used neural network with different layers of 2 strides. Layers are specified using the following format: the number of feature maps, filter size, strides, activation, and bias. *Relu* in the figure is the reference activation function. Left to right shows 2 layers, 3 layers, and 4 layers with 2 strides, respectively.

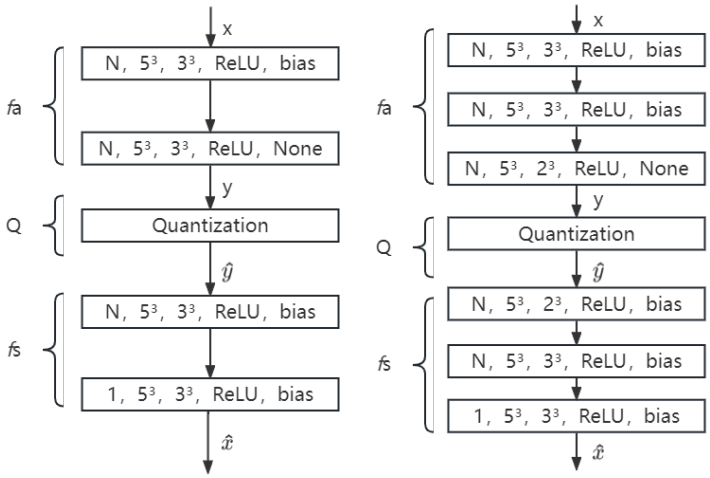

Figure 3. The architecture of the used neural network with different layers of 3 strides. *Relu* in the figure is the reference activation function. Left to right shows 2 layers and 3 layers with 3 strides, respectively.

3.1.2. Activation Functions

The choices of activation functions have a significant impact on the convolutional neural network. An activation function is a function transformation between the input and output of the neural network layer, which allows one to add nonlinear factors and enhance the expression ability of the model. However, in practical applications, it remains challenging to determine which activation function is the best or the most effective. In this study, we evaluate nine activation functions.

(1) *Sigmoid*.
The *Sigmoid* function is also known as the logistic function, which is used for the output of hidden layer neurons and the binary classification. The value range is (0, 1), which can map a real number to the interval of (0, 1). The effect is more preferable when the feature difference is complex or not large.

$$f(x) = \frac{1}{1 + e^{-x}} \tag{1}$$

(2) *Tanh*.
The *Tanh* activation function is also named as the hyperbolic tangent activation function. Similar to the *Sigmoid* function, the *Tanh* function also uses the truth values, and the *Tanh* function converts it to the range of -1 to 1. Therefore, the output of the *Tanh* function is zero-centered, which solves the problem of slow convergence of *Sigmoid* function and improves the convergence speed compared with *Sigmoid*.

$$\begin{array}{l} Tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x'}} \\ Tanh(x) = 2Sigmoid(2x) - 1 \end{array} \tag{2}$$

(3) *Relu*.
The *Relu* function converts the output of some neurons to be 0, which causes the sparsity of the network. Thus, the *Relu* function has the advantages of reducing the parameter's interdependence and alleviating the occurrence of over-fitting problems.

$$Relu(x) = \max(0, x) \tag{3}$$

(4) *Leaky Relu*.
Compared to the *Relu* function, the output of this function is no longer 0 under the condition that x is negative. Here, a is usually taken as 0.01. Thus, it solves the problem of gradient disappearance in the *Relu* activation function, which also shows the advantages of efficient computation and faster convergence rates than the *Sigmoid*/*Tanh* function.

$$Leaky\ Relu(x) = \begin{cases} x(x \geq 0) \\ \text{ax}(x < 0) \end{cases} \tag{4}$$

(5) *Elu*
The exponential linear unit (*Elu*) activation function is proved to have high noise robustness and can convert the average activation value of neurons that approach zero. Due to the necessity to calculate the index, the calculation cost is high.

$$Elu(x) = \begin{cases} x, & \text{if } x > 0 \\ \alpha(e^x - 1), & \text{otherwise} \end{cases} \tag{5}$$

(6) *Gelu*
The Gaussian error linear units (*Gelu*) function introduces the idea of stochastic regularity in activation, a probabilistic description of neuronal input that intuitively complies with our natural understanding.

$$Gelu(x) = \frac{x}{1 + e^{-1.702x}} \tag{6}$$

(7) *Selu*
The scaled empirical linear units (*Selu*) function introduces the attribute of self-normalization, which avoids the problem of sudden disappearance or the explosive growth of the gradient. Therefore, this function enables the learning process to be more stable than other functions mentioned above.

$$Selu(x) = \lambda \begin{cases} x & \text{if } x > 0 \\ \alpha e^x - \alpha & \text{if } x \leq 0 \end{cases} \tag{7}$$

(8) *R-Relu*
The Random Relu (*R-Relu*) function is also a variant of *Leaky Relu*, where the slope of negative value is random in training.

$$\begin{gathered} y_{ji} = \begin{cases} x_{ji} & \text{if } x_{ji} \geq 0 \\ a_{ji} x_{ji} & \text{if } x_{ji} < 0 \end{cases} \\ \text{where} \\ a_{ji} \sim U(l, u), l < u \text{ and } l, u \in [0, 1) \end{gathered} \tag{8}$$

(9) *HardSwitch*
The *HardSwitch* activation function is used to approximate the *Switch* activation function. This function introduces subsection calculation, which greatly reduces the amount of calculation under the condition. This allows one to retain the feature, and thus the *Switch* function can enable the neural network layer to have more expressive ability.

$$HardSwish(x) = x \cdot \begin{cases} 1, & x \geq 3 \\ \frac{x}{6} + \frac{1}{2}, & -3 < x < 3 \\ 0, & x \leq -3 \end{cases} \tag{9}$$

3.1.3. Training

In this work, the model date we use is from the Princeton 3D Object Dataset ModelNet40, which provides a comprehensive clean collection of 3D CAD models for objects [21,37]. ModelNet40 is collection of 3D CAD models for object classification and segmentation, including 40 categories, 9843 (80%)point-cloud data in the training set, and 2468 (20%) point-cloud data in the testing set. The experimental environment is configured with Python = 3.6 and Tensorflow = 0.12. The configuration of the convolution neural network is filters N = 32, batch size = 64, and Adam's learning rate $lr = 10^{-4}$.

In our work, we use two measurement indicators in the training process, *total_loss* and *total_quantiles_loss*. If *total_loss* < 2 and *total_quantiles_loss* < 50, we can determine that we have obtained an effective model. Before the training starts, we need to set the training rounds. According to the experience gained in the experiments, the general setting is 350–400 rounds. It worth to mention that more training rounds is not always better because too many training rounds cannot increase the time cost and may even reduce the quality of the model of compressed 3D point cloud. If the experimental results do not meet the requirements, we need to restart the training process until we obtain the experimental model that meets the requirements.

3.2. Network Outputs

First, we input the point-cloud model and then use the trained model to obtain the compressed binary file. We compute the size of the compressed file b as follows:

$$BPP = b * 8/n, \tag{10}$$

where BPP represents bits per pixel, n represents the number of points in the point cloud file, and a lower BPP means a higher compression rate. Then, we decompress the compressed file to obtain a reconstructed 3D point-cloud model and compare it with original model. Furthermore, we use the point cloud library (PCL) to measure the reconstruction error between the original input point cloud and the reconstructed data.

3.3. Loss Function

Our decoding procedure is viewed as a binary classification problem in which the existence or absence of the voxel grid's point z is determined. We resolve the decompressed point cloud $\hat{x} = \hat{v}_S$ into its individual voxel z, which is related to the value of p_z. Most voxels are not occupied as a result of the point cloud's sparse distribution, so the majority of its $v_S(z)$ values are zero. To balance the empty voxels and occupied voxels, we use

$$FL\left(p_z^t\right) = -\alpha_z\left(1 - p_z^t\right)^\gamma \log\left(p_z^t\right) \tag{11}$$

if $v_S(z) = 1$, p_z^t is equal to p_z or $1 - p_z$. The compressed point cloud's focal loss is indicated as:

$$FL(\tilde{x}) = \sum_{z \in S} FL\left(p_z^t\right) \tag{12}$$

Our final loss can be expressed as $L = \lambda D + R$, with D denoting the distortion derived from focal loss and R denoting the quantity of bits per input occupied voxel.

4. Result and Discussion

We use the evaluation criteria released by the moving picture experts group (MPEG) [38] and choose the symmetric rms distance between the point clouds (RMS) to evaluate the compression performance of point cloud. The point cloud is made up of a collection of points denoted by (x,y,z) and a number of attributes, with color components (y,u,v) playing a crucial role. We can define the point v, which has a specific position in a 3D space (x,y,z) and an optional colour attribute c, where the components of the attribute c are r, g, b or y, u, v and optional alternative parameters may represent normal or texture maps.

$$\text{point } v = (((x, y, z), [c], [a_0 \ldots a_A]) : x, y, z \in \boldsymbol{R}, [c \in (r, g, b) \mid r, g, b \in N], [a_i \in [0, 1]]) \tag{13}$$

At present, the point cloud consists of only a collection of K loosely ordered points:

$$\text{Original Point Cloud } V_{\text{or}} = \{(v_i) : i = 0 \ldots . K - 1\} \tag{14}$$

The first cloud serve as the standard for judging the quality of the second inferior cloud V_{deg} that consists of N points.

$$\text{Degraded Point Cloud } V_{\text{deg}} = \{(v_i) : i = 0 \ldots .. N - 1\} \tag{15}$$

RMS can be calculated as follows:

$$d_{rms}\left(V_{or}, V_{deg}\right) = \sqrt{\frac{1}{K} \sum_{vo \in Vor} [vo - vd\_\text{nearest-neighbour}]^2} \tag{16}$$

$$d_{\text{symmetric--rms}}\left(V_{or}, V_{\text{deg}}\right) = \max\left(d_{rms}\left(V_{or}, V_{\text{deg}}\right), d_{rms}\left(V_{\text{deg}}, V_{or}\right)\right) \tag{17}$$

The smaller the RMS value will result in less error between the original point cloud and the decompressed point cloud, so the point cloud will be more effectively compressed.

4.1. Evaluation of the Compression by Different Depth of The Network

Different depths determine the shape of the output graph. By deepening the network, we can decompose the problems to be studied in different levels. At present, in the study of depth, it is generally believed that if the problem is more complex and higher order, it needs a deeper network. By deepening the depth of the convolution neural network, the better the sensitivity and convergence of the model can be improved. In practice, it is not the deeper the better because the deeper the structure, the more likely the gradient disappearance phenomenon happens. Meanwhile, the output of each layer will lose part of the edge information. In our experiment, the depth less than 4 in the network can obtain better experimental results.

To ensure the effectiveness of the experiment, we first verify the framework in three layers and two strides using activation functions of *Relu*, *Selu*, and *Elu*.

Figure 4 demonstrate that our approach is successful and efficient, fulfills the anticipated demands, and outperforms octree compression by a wide margin. Then, we expand the experiment and change the depth of the convolutional neural network to two and four layers to observe the impact on the point-cloud compression effect. The results are shown in Figure 5.

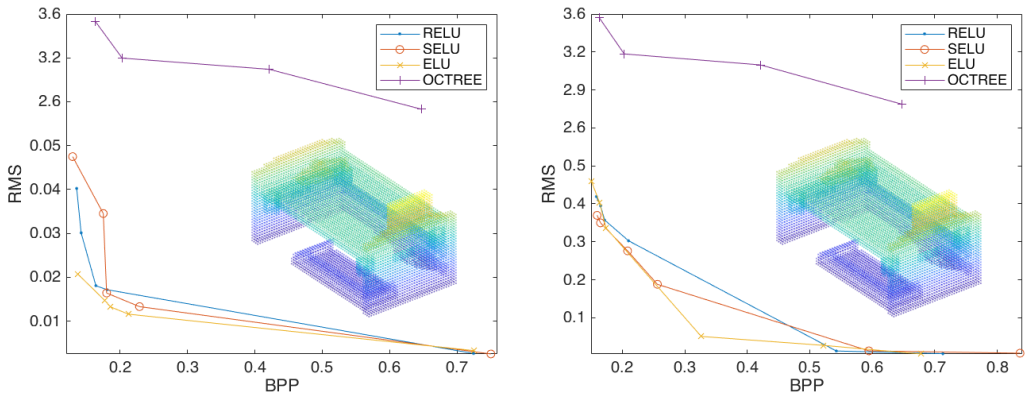

Figure 4. The compression performance of the scheme configured with three layers of convolution and two strides (**left**) and two layers convolution and two strides (**right**), using the activation functions *Relu*, *Selu*, and *Elu*.

By comparing the best performing curves in each plot in Figure 5, we find that the effect of convolutional neural network compression point cloud is better than that of octree compression, the compression effect of three layers of compression convolution is better than that of two layers compression convolution, and the compression effect of four layers of compression convolution is better than that of three layers compression convolution, while using the activation function *Elu*, which is also the best compression effect. Specifically, the four layer compression convolution is better than that of the three layer compression convolution, with an average compression rate that has been improved by 32.29%.

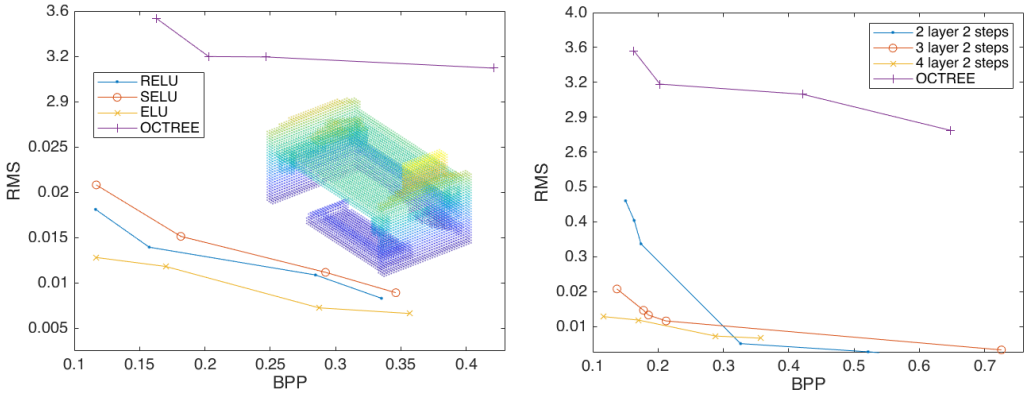

Figure 5. The compression performance of the scheme configured with four layers of convolution, and two strides, using the activation functions *Relu*, *Selu*, and *Elu* (**left**). Comparison of convolution effects of each layer (**right**).

4.2. Evaluation of the Compression by Different Number of Strides

The purpose of setting the stride is to reduce the number of input parameters and reduce the amount of calculation. The value of the side parameter is the multiple of reduction. In general, the smaller the stride size is, the easier it is to obtain local optimization. Additionally, the larger the stride size is, the better it will obtain global optimization. However, too large of a step size will lose a lot of graphic details. When preparing for our experiment, we found that if the stride is set to 1, the program would be down due to excessive computation. If the step size is more than 3, the compression effect is significantly reduced. For this reason, we chose strides 2 and 3 as the experimental parameters and used the half-filling method to increase the size of the output graphics.

In order to further explore the impact of stride on the point-cloud compression effect, we changed the experimental stride to 3 and repeated the above experiment. The results showed that the compression with four layers and three strides did not meet the experimental requirements, which is excluded here. Figure A1 shows the compression performance of the scheme configured with different depths and strides. Figure 6 shows the overall compression performance with different specifications to the networks.

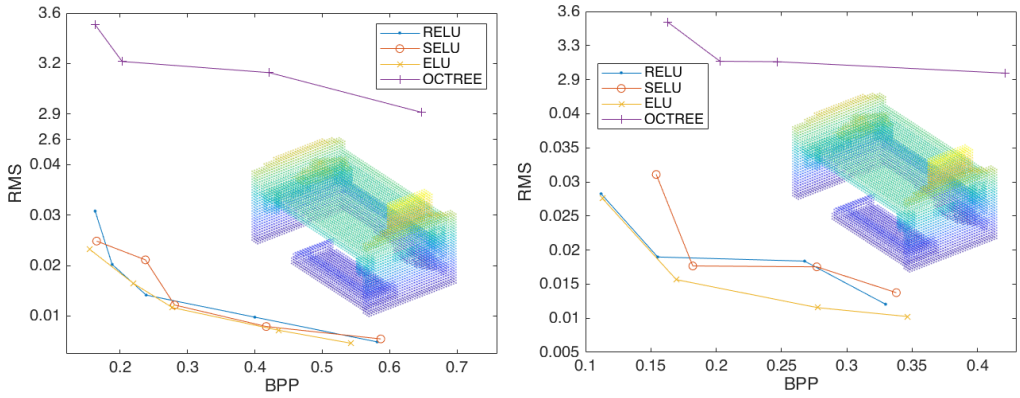

Figure 6. The compression performance of the scheme configured with two layers of convolution and three strides (**left**) and three layers of convolution and three strides (**right**), using the activation functions *Relu*, *Selu*, and *Elu*.

Figure 7 demonstrates that when the depth of compression convolutions are the same, with three layers of compression convolution, the compression effect of three strides is significantly better than that of two strides. This shows that the compression impact of two

layers of compression convolution is less effective under two strides condition. Under three layers of compression convolution using the activation function *Elu*, the compression effect of two strides is better than that of stride three, with an average compression rate improved by 28.35%. This demonstrated that by changing the stride one can optimize the compression effect of the point cloud to a certain extent, but it is unstable.

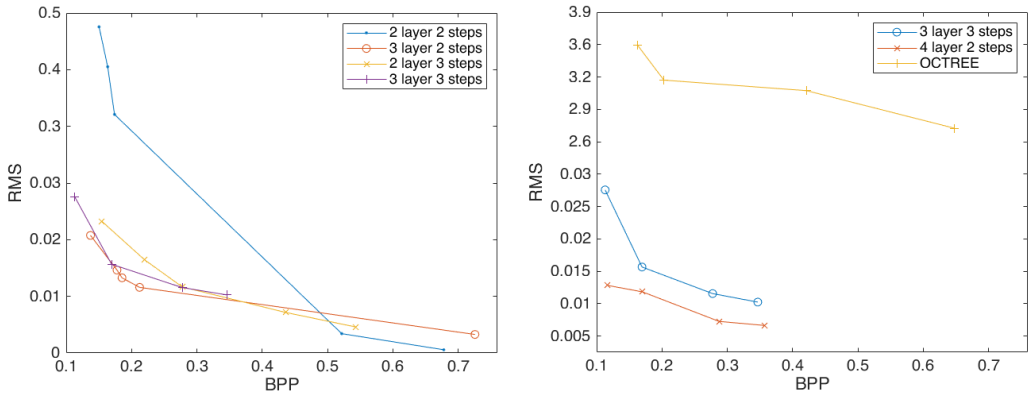

Figure 7. Comparison of stride effects of each layer (**left**); curve comparison for best results (**right**).

Furthermore, we have discovered that the compression effect produced by four layers and two strides is the best. Meanwhile, the compression impact of three strides is discovered to not always be worse than that of two strides. Throughout all of the experiments, the activation function *Elu* exhibits the best compression result, demonstrating the stability of the activation function's support for point-cloud compression.

4.3. Evaluation of the Compression of Different Activation Functions

The activation functions are significantly important in neural networks. They determine whether a neuron is activated, whether the information received by the neuron is useful, and whether it should be left or abandoned. The neural network without an activation function is a linear regression model. In the research field of graphics processing, the nonlinear transformation of the activation function can make the neural network process very complex. The application scenarios of different activation functions are also different. Although their introductions have a certain reference significance for us to select activation functions, for different studies, the activation functions that are most suitable for the model can only be determined based on experimental comparison.

Therefore, we continue to investigate the activation function and provide the following six activation functions for experiments: *Sigmoid*, *Tanh*, *Leaky Relu*, *R-Relu*, *Gelu and HardSwitch*. Figure A2 shows the compression effects of 3D point clouds with different activation functions.

Figure 8 shows that when the depth is 2, the best activation function is *Sigmoid*, which is not far from the other functions. Figure 9 shows that when the depth is 3, the *Sigmoid* function is significantly improved compared to the other functions. Figure 10 and Table 1 shows that the *Sigmoid* function can still achieve better a compression effect when the depth is 4. In Figure 11, the highest performance across all experiments is demonstrated by the point-cloud compression utilizing the *Sigmoid* function of four layers and two strides. Therefore, the *Sigmoid* function performs best at improving the compression effect of point clouds.

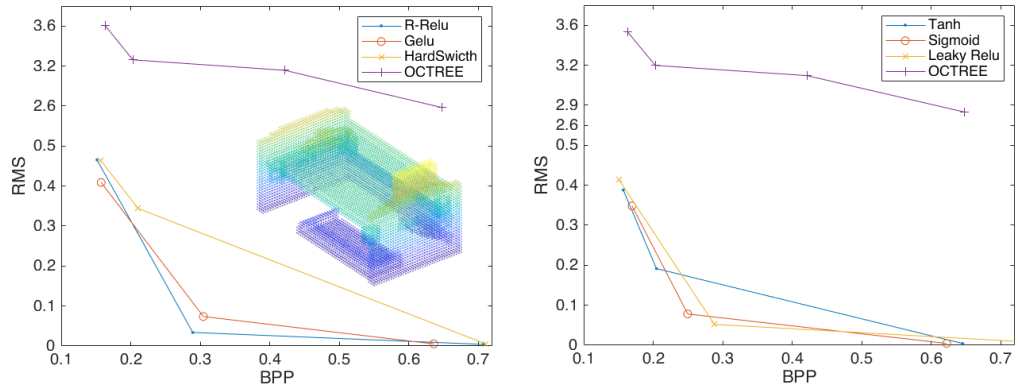

Figure 8. The compression performance of the scheme configured with two layers of convolution and two strides, using the activation functions *R-Relu*, *Gelu*, and *HardSwitch* (**left**) and *Tanh*, *Sigmoid*, and *Leaky Relu* (**right**).

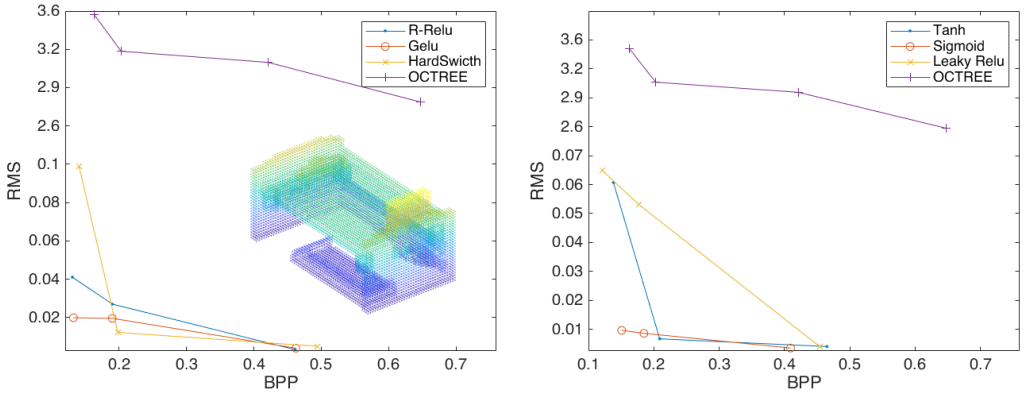

Figure 9. The compression performance of the scheme configured with three layers of convolution and two strides using the activation functions *R-Relu*, *Gelu*, and *HardSwitch* (**left**) and *Tanh*, Sigmoid, and Leaky Relu (**right**).

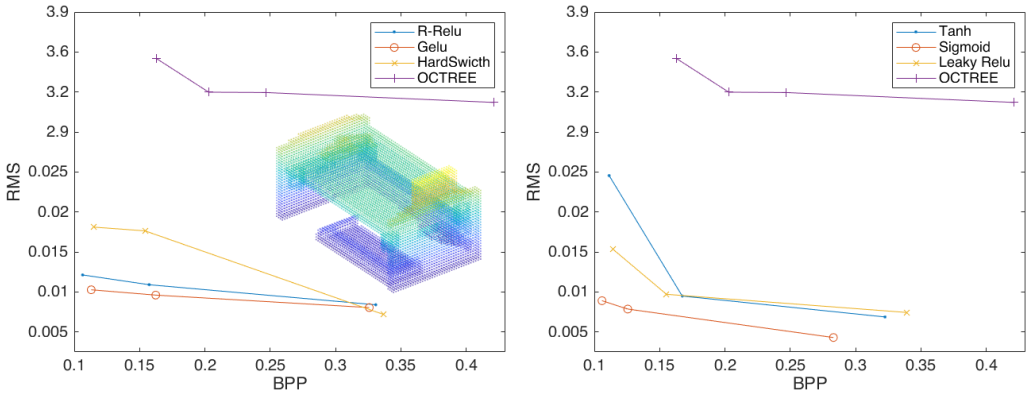

Figure 10. The compression performance of the scheme configured with four layers of convolution and two strides using the activation functions *R-Relu*, *Gelu*, and *HardSwitch* (**left**) and *Tanh*, *Sigmoid*, and *Leaky Relu* (**right**).

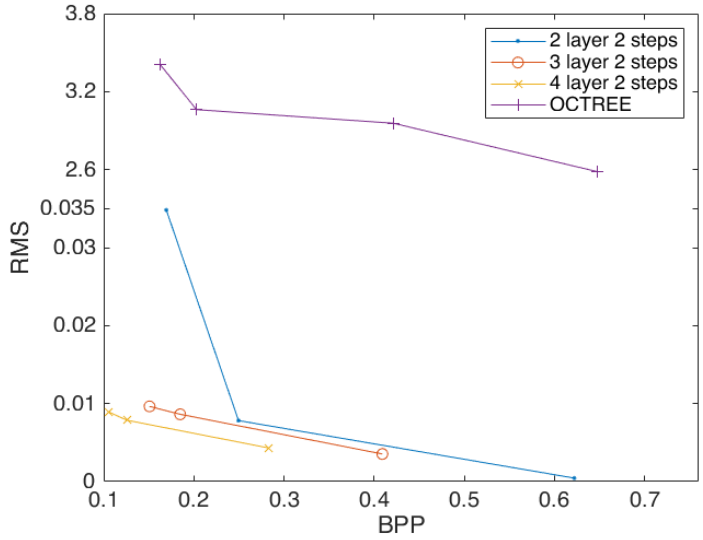

Figure 11. Three curves that perform best at each depth using the *Sigmoid* function.

Table 1. RMS of 9 activation functions with four layers of convolution and two strides. Italic indicates the activation function, and bolded indicates the minimum RMS data.

Accuracy / Function	5×10^{-5}	1×10^{-5}	5×10^{-6}
Relu	0.00831488	0.0139485	0.0181215
Selu	0.00893118	0.0151461	0.0208207
Elu	0.00664042	0.0118255	0.0128144
Tanh	0.00688373	0.0094999	0.0245558
Leaky Relu	0.00744641	0.0097154	0.0153705
R-Relu	0.00840702	0.0109112	0.0121359
Gelu	0.00805524	0.0096145	0.0102793
HardSwitch	0.00719350	0.0176457	0.0181156
sigmoid	**0.00431234**	**0.0078694**	**0.0089128**

5. Conclusions

In this work, we have used the optimized convolutional neural network to compress the point cloud and have investigated the compression effect of the point cloud at the same time. By varying the depth, strides, and activation functions of various convolutional neural networks, we have evaluated the impact on the point-cloud compression effect and further the research that has been done so far on compressing point clouds using convolutional neural networks. We have found that the deeper the depth of the neural network is, the better the compression effect of the point cloud is. More specifically, it can significantly improve the compression effect of the point cloud when the depth of the convolutional neural network is 4. Meanwhile, the compression effect of the point cloud is also affected by the stride; the shorter the stride size achieves the better experimental effect. Since an excessively small stride can lead to a significant increase in computational costs, the most suitable step size for our experiments is 2. We have discovered that the improvement of the point-cloud compression effect by changing the activation function is stable. Furthermore, We have concluded that the *Sigmoid* function outperformed the other activation functions mentioned above by comparing the ultimate compression effects. Therefore, we obtain the optimal parameter configuration of four layers and two strides using the *Sigmoid* activation function, which is 208.56% higher than the original framework proposal's default configuration of three layers of convolution and two strides using the *Relu* function. We have applied our experimental scheme to other 3D models, which demonstrated the validity and generality of our experimental results in Figure 12.

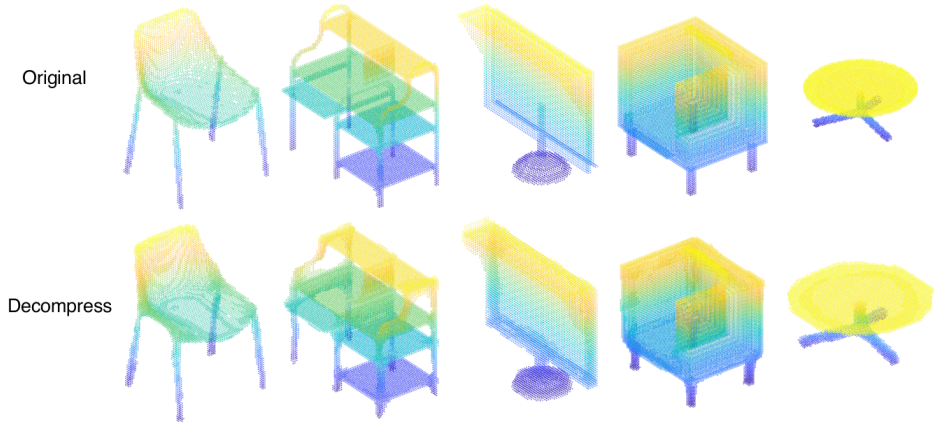

Figure 12. Results from different point clouds.

Based on this study, as a part of future works, it is possible to further improve the compression performance by changing the data set used for training, alternating other activation functions not involved in other articles, changing the depth of the convolution network used, etc. In addition, the time cost is also one of the factors used to evaluate the compression approaches. In this work, we have not taken the time cost into account because the model training takes a relatively short period of time.

Author Contributions: Conceptual design: G.L. and Z.Z.; methodology: Y.X.; experimental verification: B.H. and L.W.; formal analysis: H.W., X.S. and Y.X.; writing—draft: B.H.; writing—review and editing: B.H. and G.L. The supervisor is G.L. All authors have read and agreed to the published version of the manuscript.

Funding: This work has been jointly supported by the National Natural Science Foundation of China under Grant 61962021, 62202165; the Natural Science Foundation of Jiangxi Province under Grant 20223BBE51039, 20224BAB202016, and 20224BAB212014; and the Ministry of Science and Technology of China under Grant G2022022001L.

Institutional Review Board Statement: This article does not involve ethical research and does not require ethical approval.

Informed Consent Statement: Informed consent was obtained from all subjects involved in the study.

Data Availability Statement: The common dataset ModelNet40 used in this study can be obtained from link https://modelnet.cs.princeton.edu/.

Conflicts of Interest: The authors declare no conflict of interest.

Appendix A

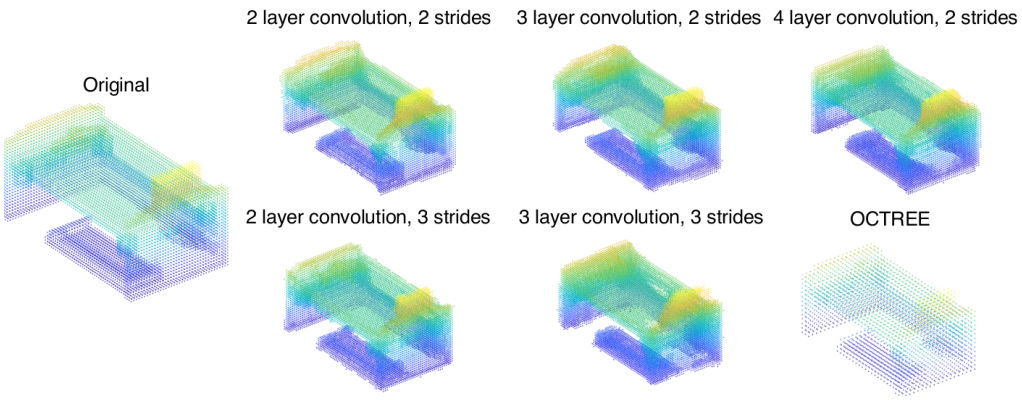

Figure A1. The compression performance of the scheme configured with different depths and strides.

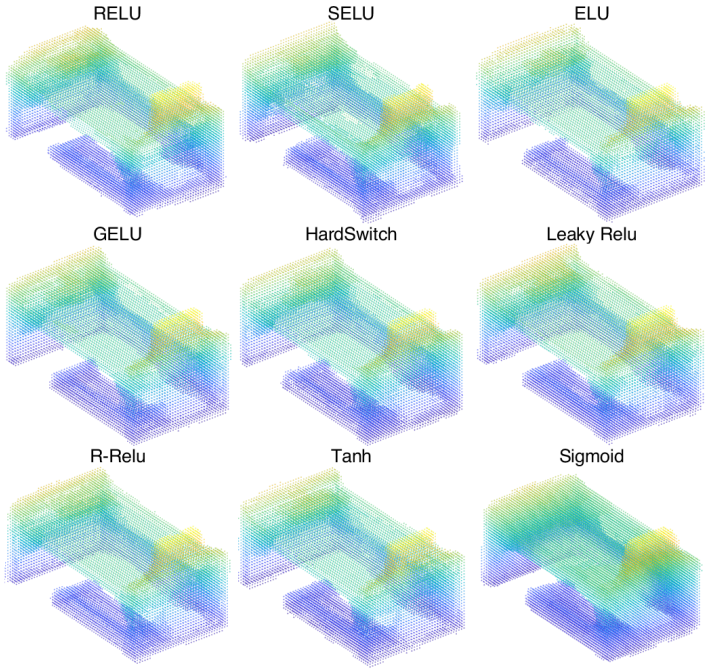

Figure A2. The compression effects of 3D point clouds with different activation functions.

References

1. Popişter, F.; Popescu, D.; Păcurar, A.; Păcurar, R. Mathematical Approach in Complex Surfaces Toolpaths. *Mathematics* **2021**, *9*, 1360. [CrossRef]
2. Zhao, J.; Xu, S.; Zhang, B.; Gu, J.; Doermann, D.; Guo, G. Towards compact 1-bit CNNs via Bayesian learning. *Int. J. Comput. Vis.* **2022**, *130*, 201–225. [CrossRef]
3. Li, B.; Wu, B.; Su, J.; Wang, G. Eagleeye: Fast sub-net evaluation for efficient neural network pruning. In Proceedings of the Computer Vision—ECCV 2020: 16th European Conference, Glasgow, UK, 23–28 August 2020; Proceedings, Part II 16; Springer: Berlin/Heildeberg, Germany, 2020; pp. 639–654.
4. Zhang, B.; Wang, R.; Wang, X.; Han, J.; Ji, R. Modulated convolutional networks. *IEEE Trans. Neural Netw. Learn. Syst.* **2021**, 1–14. [CrossRef] [PubMed]
5. Yeom, J.; Stan, T.; Hong, S.; Voorhees, P.W. Segmentation of experimental datasets via convolutional neural networks trained on phase field simulations. *Acta Mater.* **2021**, *214*, 116990. [CrossRef]
6. Qayyum, W.; Ehtisham, R.; Bahrami, A.; Camp, C.; Mir, J.; Ahmad, A. Assessment of Convolutional Neural Network Pre-Trained Models for Detection and Orientation of Cracks. *Materials* **2023**, *16*, 826. [CrossRef]
7. Wang, Y.; Sun, Y.; Liu, Z.; Sarma, S.E.; Bronstein, M.M.; Solomon, J.M. Dynamic graph cnn for learning on point clouds. *Acm Trans. Graph.* **2019**, *38*, 1–12. [CrossRef]
8. Liu, Z.; Tang, H.; Lin, Y.; Han, S. Point-voxel cnn for efficient 3d deep learning. *Adv. Neural Inf. Process. Syst.* **2019**, *32*. [CrossRef]
9. Xu, S.; Li, Y.; Zhao, J.; Zhang, B.; Guo, G. Poem: 1-bit point-wise operations based on expectation-maximization for efficient point-cloud processing. *arXiv* **2021**, arXiv:2111.13386.
10. Alkhouly, A.A.; Mohammed, A.; Hefny, H.A. Improving the performance of deep neural networks using two proposed activation functions. *IEEE Access* **2021**, *9*, 82249–82271. [CrossRef]
11. Liu, J.; Liu, Y.; Zhang, Q. A weight initialization method based on neural network with asymmetric activation function. *Neurocomputing* **2022**, *483*, 171–182. [CrossRef]
12. Kumar, A.; Sodhi, S.S. Neural network with NewSigmoid activation function. *J. Intell. Fuzzy Syst.* **2022**, *43*, 545–559. [CrossRef]
13. Siegel, J.W.; Xu, J. Approximation rates for neural networks with general activation functions. *Neural Netw.* **2020**, *128*, 313–321. [CrossRef] [PubMed]
14. Guede, C.; Andrivon, P.; Marvie, J.E.; Ricard, J.; Redmann, B.; Chevet, J.C. V-PCC: Performance evaluation of the first MPEG Point Cloud Codec. In Proceedings of the SMPTE 2020 Annual Technical Conference and Exhibition, SMPTE, Virtual, 10–12 November 2020; pp. 1–27.
15. Dumic, E.; da Silva Cruz, L.A. point-cloud coding solutions, subjective assessment and objective measures: A case study. *Symmetry* **2020**, *12*, 1955. [CrossRef]
16. Garcia, D.C.; de Queiroz, R.L. Intra-frame context-based octree coding for point-cloud geometry. In Proceedings of the 2018 25th IEEE International Conference on Image Processing (ICIP), Athens, Greece, 7–10 October 2018; IEEE: Piscataway, NJ, USA, 2018; pp. 1807–1811.

17. de Queiroz, R.L.; Garcia, D.C.; Chou, P.A.; Florencio, D.A. Distance-based probability model for octree coding. *IEEE Signal Process. Lett.* **2018**, *25*, 739–742. [CrossRef]
18. Thanou, D.; Chou, P.A.; Frossard, P. Graph-based compression of dynamic 3D point-cloud sequences. *IEEE Trans. Image Process.* **2016**, *25*, 1765–1778. [CrossRef]
19. de Queiroz, R.L.; Chou, P.A. Motion-compensated compression of point cloud video. In Proceedings of the 2017 IEEE International Conference on Image Processing (ICIP), Beijing, China, 17–20 September 2017; IEEE: Piscatawaj, NJ, USA, 2017; pp. 1417–1421.
20. Mekuria, R.; Blom, K.; Cesar, P. Design, implementation, and evaluation of a point cloud codec for tele-immersive video. *IEEE Trans. Circuits Syst. Video Technol.* **2016**, *27*, 828–842. [CrossRef]
21. Wu, Z.; Song, S.; Khosla, A.; Yu, F.; Zhang, L.; Tang, X.; Xiao, J. 3d shapenets: A deep representation for volumetric shapes. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Boston, MA, USA, 7–15 June 2015; pp. 1912–1920.
22. Valenzise, G.; Purica, A.; Hulusic, V.; Cagnazzo, M. Quality assessment of deep-learning-based image compression. In Proceedings of the 2018 IEEE 20th International Workshop on Multimedia Signal Processing (MMSP), Vancouver, BC, Canada, 29–31 August 2018; IEEE: Piscataway, NJ, USA, 2018; pp. 1–6.
23. Huang, T.; Liu, Y. 3d point cloud geometry compression on deep learning. In Proceedings of the 27th ACM International Conference on Multimedia, Nice, France, 21–25 October 2019; pp. 890–898.
24. Rusu, R.B.; Cousins, S. 3d is here: Point cloud library (pcl). In Proceedings of the 2011 IEEE International Conference on Robotics and Automation, Shanghai, China, 09–13 May 2011; IEEE: Piscatawaj, NJ, USA, 2011; pp. 1–4.
25. de Hoog, J.; Ahmed, A.N.; Anwar, A.; Latré, S.; Hellinckx, P. Quality-aware compression of point clouds with google draco. In Proceedings of the Advances on P2P, Parallel, Grid, Cloud and Internet Computing: Proceedings of the 16th International Conference on P2P, Parallel, Grid, Cloud and Internet Computing (3PGCIC-2021), Fukuoka, Japan, 28–30 October 2021; Springer: Cham, Switzerland, 2022; pp. 227–236.
26. Achlioptas, P.; Diamanti, O.; Mitliagkas, I.; Guibas, L. Learning representations and generative models for 3d point clouds. In Proceedings of the International Conference on Machine Learning, PMLR, Stockholm, Sweden, 10–15 July 2018; pp. 40–49.
27. Goodfellow, I.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A.; Bengio, Y. Generative adversarial nets Advances in neural information processing systems. *arXiv* **2014** arXiv:1406.2661.
28. Ballé, J.; Laparra, V.; Simoncelli, E.P. End-to-end optimized image compression. *arXiv* **2016** arXiv:1611.01704.
29. Quach, M.; Valenzise, G.; Dufaux, F. Learning convolutional transforms for lossy point cloud geometry compression. In Proceedings of the 2019 IEEE International Conference on Image Processing (ICIP), Taipei, Taiwan, 22–25 September 2019; IEEE: Piscataway, NJ, USA, 2019; pp. 4320–4324.
30. Karim, A.M.; Kaya, H.; Alcan, V.; Sen, B.; Hadimlioglu, I.A. New optimized deep learning application for COVID-19 detection in chest X-ray images. *Symmetry* **2022**, *14*, 1003. [CrossRef]
31. Sung, Y.H.; Park, S.J.; Kim, D.Y.; Kim, S. GPS Spoofing Detection Method for Small UAVs Using 1D Convolution Neural Network. *Sensors* **2022**, *22*, 9412. [CrossRef]
32. Arenas, R. Design of a Forest Fire Early Alert System through a Deep 3D-CNN Structure and a WRF-CNN Bias Correction. *Sensors* **2022**, *22*, 8790. [CrossRef]
33. Liu, C.; Ding, W.; Chen, P.; Zhuang, B.; Wang, Y.; Zhao, Y.; Zhang, B.; Han, Y. RB-Net: Training highly accurate and efficient binary neural networks with reshaped point-wise convolution and balanced activation. *IEEE Trans. Circuits Syst. Video Technol.* **2022**, *32*, 6414–6424. [CrossRef]
34. Xu, S.; Li, Y.; Wang, T.; Ma, T.; Zhang, B.; Gao, P.; Qiao, Y.; Lü, J.; Guo, G. Recurrent bilinear optimization for binary neural networks. In Proceedings of the Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, 23–27 October 2022; Proceedings, Part XXIV; Springer: Cham, Switzerland, 2022; pp. 19–35.
35. Jin, T.; Dai, H.; Cao, L.; Zhang, B.; Huang, F.; Gao, Y.; Ji, R. Deepwalk-aware graph convolutional networks. *Sci. China Inf. Sci.* **2022**, *65*, 152104. [CrossRef]
36. Zeng, B.; Liu, B.; Li, H.; Liu, X.; Liu, J.; Chen, D.; Peng, W.; Zhang, B. FNeVR: Neural Volume Rendering for Face Animation. *arXiv* **2022** arXiv:2209.10340.
37. Sedaghat, N.; Zolfaghari, M.; Amiri, E.; Brox, T. Orientation-boosted voxel nets for 3D object recognition. *arXiv* **2016** arXiv:1604.03351.
38. Mekuria, R.; Li, Z.; Tulvan, C.; Chou, P. *Evaluation Criteria for PCC (Point-Cloud Compression)*; DANS: The Hague, The Netherlands, 2016.