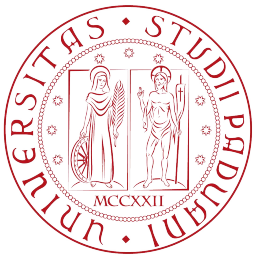


Università degli studi di Padova
Department of Information Engineering
Master Degree in Computer Engineering

Point Cloud Geometry Compression Using Neural Implicit Representations

Supervisor
Prof. Simone Milani
Co-supervisor
Daniele Mari

Graduating Student
AmirHossein Motameni
Matricola: 2055981

Academic Year 2022/2023

Graduation Date 24/10/2023

"The best way to predict the future is to create it."
Abraham Lincoln

Contents

List of Figures

List of Tables

Sommario

Negli ultimi anni, la crescente importanza delle nuvole di punti in varie applicazioni ha portato a una crescente necessità di metodi di archiviazione e trasmissione efficienti. La dimensione di questo tipo dato presenta sfide in termini di rendering, trasmissione e usabilità generale. Questa tesi introduce un nuovo approccio alla compressione della geometria delle nuvole di punti sfruttando le Implicit Neural Representations, in particolare attraverso l'uso di un modello di rete DiGS. Addestrando questo modello su una singola nuvola di punti, otteniamo una rappresentazione neurale compatta della sua geometria. In particolare, questa consente la ricostruzione del modello 3D con risoluzione arbitraria. Viene applicata a quasto punto inoltre la quantizzazione dinamica sul modello addestrato, riducendone significativamente le dimensioni senza compromettere molto la qualità della nuvola di punti ricostruita. I successivi processi di dequantizzazione vengono utilizzati per ricostruire una rappresentazione ad alta fedeltà della nuvola di punti originale. I nostri risultati sperimentali dimostrano l'efficacia di questo approccio in termini di rapporti di compressione e qualità della ricostruzione, valutati in termini di tradeoff tra distorsione e dimensione del bitstream. Questa ricerca fornisce una direzione promettente per l'archiviazione e la trasmissione efficienti della geometria della nuvola di punti, soddisfacendo alcune delle crescenti esigenze dell'era dei dati 3D.

Abstract

In recent years, the increasing prominence of 3D point clouds in various applications has led to an escalating need for efficient storage and transmission methods. The sheer size of these point cloud datasets presents challenges in rendering, transmission, and general usability. This thesis introduces a novel approach to point cloud geometry compression leveraging neural implicit representations, specifically through the use of a DiGS network model. By training this model on a single point cloud, we achieve a compact neural representation of its geometry. Notably, this representation allows for the reconstruction of the point cloud with an arbitrary resolution. After training a reconstructing network, dynamic quantization is applied on the trained weights, significantly reducing its overall bitrate without strongly compromising the quality of the reconstructed point cloud. A dequantization is then used to rebuild a high-fidelity representation of the original point cloud. Our experimental results demonstrate the efficacy of this approach in terms of compression ratios and reconstruction quality, assessed using PSNR relative to the bitrate. This research provides a promising direction for efficient point cloud geometry storage and transmission, addressing some of the growing demands of the 3D data era.

1

Introduction

In this chapter, a brief introduction of point clouds, their importance, main applications and challenges are going to be presented.

1.1 Background on 3D Point Clouds

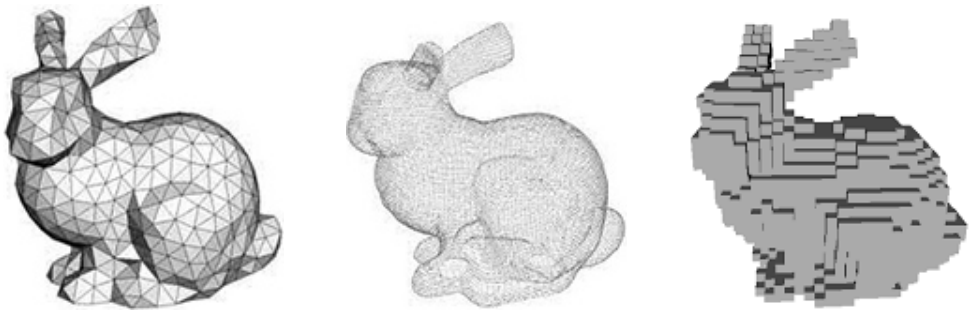
The evolution of digital technology has amplified the accessibility and application of three-dimensional data across a myriad of fields, from architectural design and virtual reality simulations to environmental modeling and medical imaging. The way we represent 3D structures and geometries deeply affects the relevance of such advancements. Three primary representations have emerged as dominant in capturing the intricacies of the 3D world: meshes (Figure 1.1a), point clouds (Figure 1.1b), and voxels (Figure 1.1c).

- **Point Clouds**: At its core, a point cloud is a set of data points in a three-dimensional coordinate system. These points, which represent the external surfaces of objects or landscapes, provide a direct and often unprocessed view of the 3D world. They are commonly acquired using methods such as LiDAR scanning, stereo imaging, or structured light techniques.

- **Voxels**: Voxels, or volumetric pixels, offer a different take on 3D representation. Here, the 3D space is discretized into a regular grid, with each cell (or voxel) typically carrying information about the presence or absence of the object in that region of space. This representation is invaluable in applications demanding a volumetric approximation of the inspected object, like medical imaging.

- **Meshes**: Meshes, on the other hand, are a structured approach to 3D representation. Comprising vertices (points), edges (lines connecting points), and faces (surfaces), meshes provide a continuous view of 3D objects and are particularly prominent in computer graphics and 3D modeling.

While each of these 3D representations carries its unique advantages tailored to specific applications, they all share a common challenge: the sheer volume of data. The resulting data, especially at high resolutions, becomes unwieldy, posing significant challenges in storage, transmission, and real-time rendering. Our research explicitly delves into the realm of point clouds, aiming to tackle the size problem through the lens of neural implicit representations and dynamic compression techniques.



(a) Mesh **(b)** Point Cloud **(c)** Voxels

Figure 1.1: Mesh, point cloud, and voxels representations of a bunny.

1.2 Importance and Applications of 3D Data Representations

The rise of digital transformation across various industries underscores the increasing relevance of 3D data. The choice of representation, whether point clouds, voxels, or meshes, often hinges on the specific application in question. Below we delve deeper into the value these representations bring and their wide-ranging applications:

- **Urban Planning and Infrastructure Development**. City planners and architects harness scanned 3D data (most of the time point clouds) for site inspections, building modeling, and infrastructure development. The ability to capture a real-world environment in detail aids in designing more efficient transportation systems, optimizing city layouts, and even preserving historical sites. [5]

- **Medical Imaging**. The medical field has seen revolutionary advancements through 3D imaging, especially with voxel-based representations. From MRI scans to CT images, a volumetric understanding of human parts aids in diagnostics, treatment planning, and even surgical simulations. [6]

- **Gaming and Entertainment**. The entertainment industry, including video games and movie production, leans heavily on mesh representations. These provide the means for a structured and detailed visualization necessary for character design, environment modeling, and special effects, ensuring a lifelike and immersive user experience. [7]

- **Virtual and Augmented Reality (VR/AR)**. VR and AR platforms thrive on detailed 3D representations. Whether it's for educational simulations or interactive gaming, these platforms require high-fidelity models to provide a seamless and realistic experience.

- **Autonomous Navigation**. The autonomous vehicle industry, including drones and self-driving cars, critically depends on point clouds for obstacle detection and navigation. Generated in real-time through LiDAR or other sensors, these point clouds provide a 360-degree view of the vehicle's environment, enabling safe and efficient navigation. [8]

- **Environmental and Geological Surveys**. For understanding terrain morphology, forest density, or even riverbed topographies, 3D representations are invaluable. They provide researchers with tools to monitor environmental changes, predict natural disasters, or even assist in mineral and oil exploration.

While these applications underscore the potential and versatility of 3D data, they also underline a critical challenge. The sheer density and size of this data pose hurdles in efficient storage, quick rendering, and seamless transmission. Recognizing this, our work seeks to address the dual challenge of data fidelity and size efficiency, focusing mainly on the point cloud geometry.

1.3 Challenges of 3D Data Size and the Need for Compression

3D datasets, given their rich detail and comprehensive coverage, are inherently very large in size. As the digital world continues its evolution, 3D representations are becoming denser and more detailed, worsening the problem further and thus introducing several notable challenges.

High-fidelity 3D datasets, particularly point clouds, can be vast. A single point cloud can comprise millions, or even billions, of points, with each point recording its spatial coordinates (typically x, y, and z) [9].

1.3.1 Impediments Posed by Large 3D Datasets

Handling extensive 3D datasets introduces several impediments:

- **Storage Challenges**. Larger datasets naturally demand more storage space, increasing the infrastructure and maintenance costs. A single high-resolution 3D model can occupy gigabytes of storage [10].
- **Transmission Overheads**. Transporting vast 3D models across networks is both time-consuming and bandwidth-consuming. This is especially problematic in real-time applications or environments with limited connectivity [11].
- **Rendering Latency**. Real-time rendering, essential in industries like gaming and virtual reality, can experience significant delays when processing large datasets. The latency can hamper the user experience, particularly in immersive environments [12].
- **Memory Limitations**. Processing extensive 3D models often exceeds the memory capabilities of standard computational devices, requiring specialized or distributed systems [13].

1.3.2 The Imperative for Compression

Given these challenges, there's a palpable need for efficient compression methods. The goal is twofold: substantially reduce data size while retaining the quality and integrity

of the original model. This ensures faster transmission, efficient storage, and seamless rendering, all without significant quality degradation [14]. Our research seeks to address this pressing concern, emphasizing neural implicit representations' potential for point cloud geometry compression.

1.4 Baseline

The usage of 3D point clouds in various domains underscores the necessity for efficient storage and transmission methods. However, the vast sizes of these datasets pose challenges in both these areas. This research aims to address this crucial concern, delineating both our goal and the methodology adopted to attain it.

Our primary objective is to devise a method that enables the compression of 3D point cloud geometry without significantly compromising its integrity or fidelity. In essence, the aim is to reduce the data size for easier storage, transmission, and rendering, while preserving the quality and the salient features of the original point cloud.

1.4.1 Adopted Approach

Achieving this objective entails a two-fold approach:

1. **Neural Implicit Representation**: We leverage the DiGS [15] network model to train and establish a neural implicit representation of the 3D point cloud. This representation allows for the reconstruction of the point cloud with arbitrary resolution, providing flexibility in its use.

2. **Dynamic Quantization**: Post-training, dynamic quantization is applied to the model, effectively compressing it. This step is instrumental in reducing the size of the neural model. Subsequent dequantization, based on stored quantization parameters, ensures the reconstruction of a model closely resembling the original. The efficacy of this compression-decompression cycle is evaluated based on the difference between the original and reconstructed point cloud, called distortion, in relation to the size of the compressed bitstream.

This baseline approach forms the foundation of our research, guiding our experiments and evaluations in subsequent sections of this thesis.

1.5 Chapter Descriptions

Chapter 2 provides a comprehensive review of current point cloud compression techniques, followed by an exploration of neural implicit representations. We also shed light on the role of quantization techniques in deep learning models and how they have been applied in related domains.

Chapter 3 provides a detailed account of our research approach. It describes the dataset employed, introduces the DiGS network model, and elucidates the process of reconstructing point clouds with arbitrary resolutions. The chapter also offers insight

into the dynamic quantization process for the model, its subsequent dequantization, and the metrics adopted for evaluating the effectiveness of the compression.

In the 4th Chapter, we document our experimental procedures and present the results. The training process of the DiGS model is detailed, followed by a deep dive into the outcomes of the quantization and dequantization processes. The chapter places significant emphasis on assessing the quality of reconstructed point clouds, both through visual inspections and quantitative metrics such as PSNR.

Concluding the thesis, in the 5th Chapter, we summarize our primary findings, their potential impact on the industry, and how they could shape future research. This chapter also provides directions for further studies, innovations, and enhancements in the domain of 3D point cloud compression.

2

Literature Review

In this chapter, we will delve into the foundational literature encompassing neural implicit representations (NIR) compression and the existing techniques for point cloud coding.

2.1 Neural Implicit Representations (NIR)

Neural Implicit Representations (NIRs) are a way to represent 3D geometric data using a neural network as the underlying function. Unlike traditional methods that represent 3D shapes through explicit data structures like meshes, point clouds, or voxels, NIRs utilize the neural network to implicitly encode the shape information. The neural network is trained to map 3D coordinates to certain attributes like occupancy or signed distance, thereby defining the shape in an implicit manner.

In NIRs, a neural network is trained to learn a mapping function $f : \mathbb{R}^3 \rightarrow \mathbb{R}$ where $\mathbb{R}^3$ represents the 3D coordinate space and $\mathbb{R}$ is the attribute value, which can be either an occupancy flag or a signed distance from the surface. For a given point in 3D space $\mathbf{x}$, the function $f(\mathbf{x})$ would output a value that indicates whether the point is inside or outside the shape, or how far it is from the surface.

In Figure 2.1 you can see a simple example of NIR on a 2D Image. As you can see, the input of the model is the coordinates of the image and the output is the attribute value, which is the color of the pixel in this example.

In the context of 3D data, this function f effectively acts as an implicit field over the 3D space. You can query any point $\mathbf{x}$ in this space to determine its attribute with respect to the represented shape. This allows NIRs to represent complex 3D shapes with high fidelity, simply by querying the trained neural network.

The concept of NIRs gained traction with the introduction of methods like DeepSDF [16] and Occupancy Networks [17]. DeepSDF, for instance, uses a Signed Distance Function (SDF) to represent a shape. In particular, it trains a neural network to predict the SDF value for any given point in 3D space. Occupancy Networks employ a similar approach but use occupancy grids as the representation.

One of the key advantages of NIRs is their ability to compactly represent complex geometries. This makes them particularly useful for tasks such as shape completion, interpolation, and reconstruction from partial or noisy data. Furthermore, NIRs facilitate differentiable rendering, opening up new solutions in inverse graphics and simulation.

NIRs have a significant role in point cloud coding, especially when considering the compression and transmission of 3D data. By converting explicit point clouds into an implicit format, NIRs offer new scalability and flexibility opportunities in an efficient point cloud coding scheme.

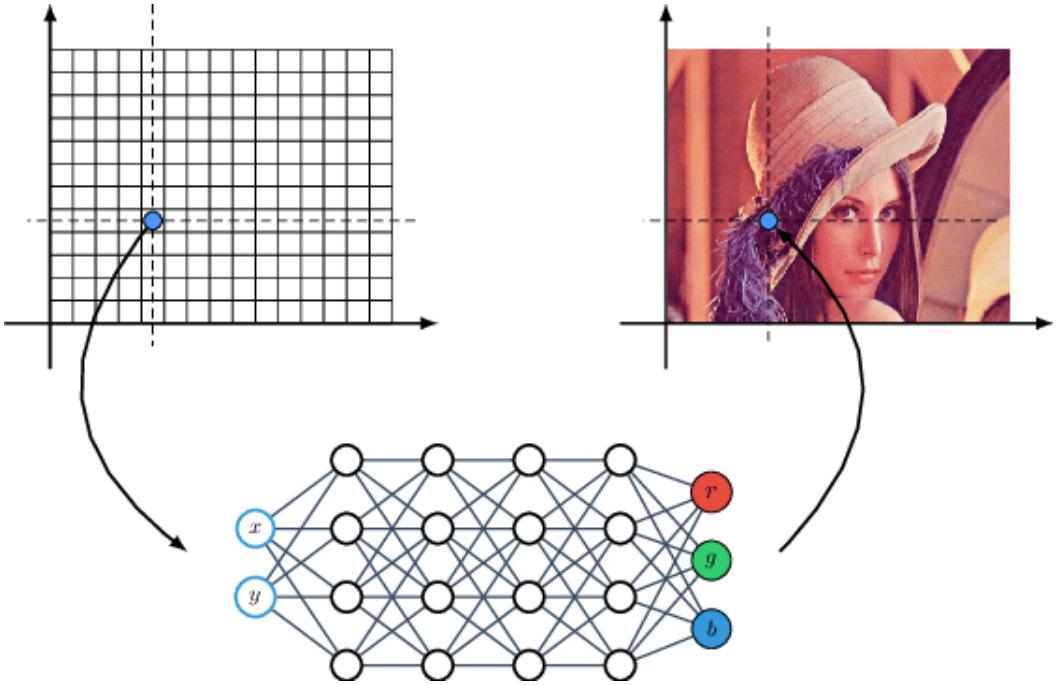


Figure 2.1: Simple example of NIR on a 2D Image.

Despite their potential, NIRs face challenges in scalability and real-time rendering. However, ongoing research focuses on optimizing these representations for better efficiency and speed, such as the introduction of hierarchical and adaptive methods [18].

2.2 Neural Implicit Representations (NIR) Compression

Neural Implicit Representations (NIR) introduce a paradigm shift in data compression by leveraging deep learning techniques to represent complex data structures implicitly. Instead of directly storing the raw data, NIR methods train neural networks to approximate it, encapsulating its essence in the network's weights and architecture. Once trained, only the weights (and occasionally the network architecture) need to be stored/transmitted, generally leading to significant reductions in the allocated memory space. In the context of 3D point cloud data, as explored in this thesis, NIR compression becomes particularly compelling. Point clouds, inherently voluminous and intricate, can be efficiently and compactly represented using neural networks, allowing for high compression rates without compromising the fidelity of the reconstructed data. This approach not only addresses storage challenges but also simplifies data transmission, making it highly-valuable for applications that require efficient handling and transportation of large 3D datasets (see Figure 2.2).

Neural Implicit Representations have emerged as a promising way for data compression, especially for complex high-dimensional data. Several research endeavors have explored this domain, presenting innovative methodologies that leverage the power of neural networks to achieve impressive compression rates without significant loss in data fidelity.

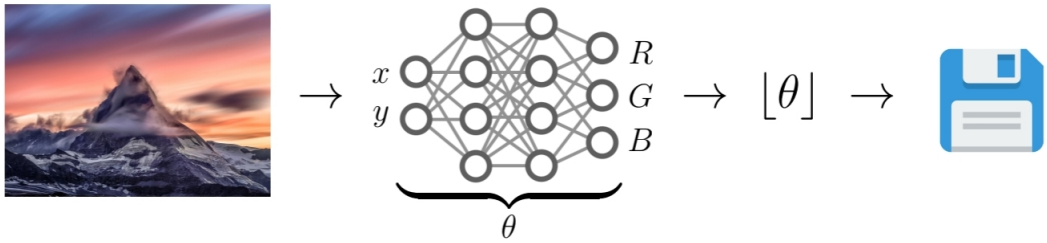


Figure 2.2: Compressed implicit neural representations. An image is overfitted with a neural network mapping pixel locations (x, y) to RGB values (often referred to as an implicit neural representation). The weights are then quantized to a lower bit-width and transmit them. [1]

2.2.1 COIN: COmpression with Implicit Neural representations

The COIN methodology, introduced in [1], stands at the forefront of data compression techniques that adopt implicit neural representations. It showcases a paradigm shift from traditional compression methods by leveraging the SIREN [19] network architecture to represent data implicitly.

The core principle behind COIN focuses on training a neural network, specifically a SIREN network, to replicate the input data that is compressed at a second stage.

What makes COIN particularly notable is its applicability to a diverse set of data types, showing its versatility by effectively compressing both 2D images and 3D shapes. Such flexibility indicates the robustness of the underlying SIREN network architecture and its adaptability to different data intricacies.

However, the most groundbreaking contribution of the paper is the idea of using neural network's weights, and specifically the SIREN architecture. This approach introduces a novel perspective on data representation, storage, and transmission. This not only offers a promising direction for future research in data compression but also paves the way for applications where efficient data handling is paramount.

2.2.2 COIN++: Neural Compression Across Modalities

COIN++, an advanced extension of the original COIN model, marks a significant milestone in the field of neural network-based data compression. The original COIN model was groundbreaking in its own right (indeed, offering a neural network-based approach to data compression across different data modalities). COIN++ pushes the boundaries even further. By adopting the Model-Agnostic Meta-Learning (MAML), a learning algorithm that significantly improves both training speed and model performance.

The essence of COIN++ lies in its innovative application of MAML, which allows for quick adaptation to new tasks with minimal data. Traditional machine learning models often require extensive training to perform well on new tasks, but the introduction of MAML in COIN++ eliminates this bottleneck. By pre-training a model in a way that it can be fine-tuned with a small amount of data for each new task, MAML makes COIN++ far more efficient and adaptable (see Figure 2.3).

This adaptability is of paramount importance in the field of data compression. Tradi-

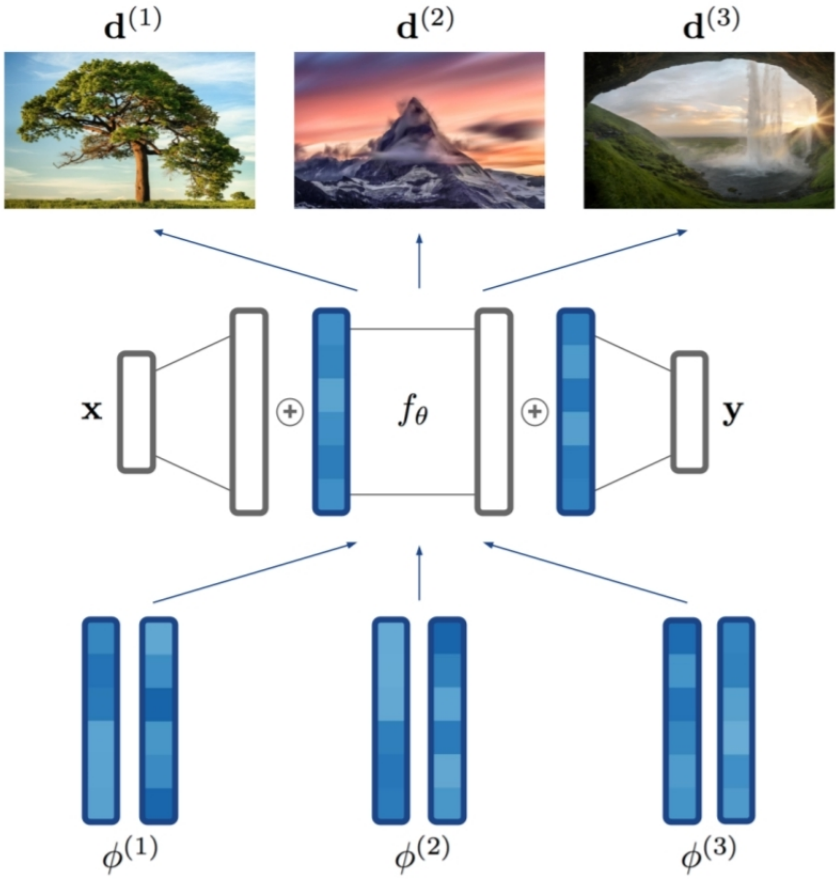


Figure 2.3: By applying modulations $\phi^{(1)}, \phi^{(2)}, \phi^{(3)}$ to a base network f_θ, it is possible to obtain different functions that can be decoded into data points $d^{(1)}, d^{(2)}, d^{(3)}$ by evaluating the functions at various coordinates. While images are showing the same principle can be applied to a range of data modalities. [2]

tional compression algorithms are often designed to work well for specific types of data but can perform poorly when applied to different data types. COIN++ addresses this limitation by being a more universal, adaptable solution, capable of handling a myriad of data modalities without the need for extensive re-training. Additionally, since each new image can be seen as a new modality this allows for considerably reducing the encoding time, i.e. the time required to overfit the given sample.

However, it's essential to note that COIN++ does not generally outperform specialized compression algorithms or even its autoencoder counterparts in all aspects. The rate-distortion curves indicate that while COIN++ offers the advantage of adaptability and training efficiency, it may not always deliver the highest compression rates or the best reconstruction quality.

Moreover, COIN++ extends the vision for the future of neural network-based compression methods. Its use of MAML points towards a new paradigm where the focus shifts from designing specialized algorithms for each type of data to developing more universal methods that can be easily adapted to new tasks. This paradigm shift could have far-reaching implications, potentially revolutionizing how we think about and approach the problem of data compression across varied data types.

2.2.3 LVAC: Learned Volumetric Attribute Compression

The paper "Learned Volumetric Attribute Compression for Point Clouds using Coordinate-Based Networks" [20] delves into the novel use of coordinate-based networks for the compression of 3D point cloud attributes. Unlike traditional methods that rely on domain-specific algorithms, LVAC employs neural networks trained to map 3D coordinates to attributes such as colors and normals. This significantly reduces the need for separate compression algorithms tailored to each attribute type.

The core idea behind LVAC is to utilize a neural network, specifically a coordinate-based network, to implicitly represent the attributes of a 3D point cloud. During the training phase, the network learns to associate 3D coordinates with corresponding attributes, thereby forming a compact representation. The trained model is then used to reconstruct the attributes, allowing for effective compression and decompression.

One of the key innovations in LVAC is the use of Meta-Learning for task-agnostic initialization. This enables the model to be fine-tuned on specific datasets, further enhancing compression efficiency. The model's parameters serve as a compressed representation of the attributes, which can be easily decoded to regenerate the original data with high fidelity.

LVAC also incorporates a rate-distortion optimization framework that balances between compression rate and reconstruction quality. This makes it versatile and applicable for a variety of use cases, from real-time applications where low bit-rate is crucial, to archival storage where quality preservation is more important.

In terms of performance, LVAC was benchmarked against traditional attribute compression methods and showed competitive results. While it may not outperform specialized algorithms in every aspect, the adaptability and simplicity of LVAC offer a compelling advantage.

In the end, LVAC introduces a new paradigm in attribute compression for point clouds by leveraging the power of coordinate-based neural networks. Its ability to adapt and fine-tune makes it a flexible solution for a wide range of applications opening up new investigation tracks.

2.2.4 Signal Compression via Neural Implicit Representations

The paper "Signal Compression via Neural Implicit Representations" [21] introduces a novel framework for signal compression utilizing neural implicit representations. Traditional signal compression methods largely rely on autoencoder-like structures. In contrast, this paper leverages neural networks trained to implicitly represent signals, where the compact representation of the signal is not a latent space but the network's weights themselves (see Figure 2.4).

The authors propose a new compression paradigm termed Neural Implicit Compression (NIC). Unlike autoencoders, which use a universal encoding function to create a compact latent space, NIC trains a neural network to represent a specific signal directly. In this paradigm, the network's weights, biases, and architecture become the compact representation of the signal.

The input to the neural network is a single coordinate from the signal domain (e.g.,

pixel location), and the output is the signal value at that coordinate (e.g., RGB value). The authors particularly focus on SIREN (Sinusoidal Representation Networks) [19] architectures for their implicit representations.

The paper also establishes a formal connection between NIC and traditional transform coding, such as Discrete Cosine Transform (DCT). It shows that a two-layer SIREN approximating a continuous function can be equivalent to an N-point 1D-DCT.

The authors introduce the use of meta-learning to provide an initial set of weights that are close to an optimal solution for a specific class of signals. This pre-training step facilitates efficient fine-tuning for individual signals, enabling further rate savings.

Experiments were conducted on the task of compressing point cloud attributes. The authors reported that NIC showed very competitive performance, reaching close to the latest MPEG G-PCC standard. The methodology was also shown to be highly adaptable, allowing for different numbers of layers, features, and quantization step sizes.

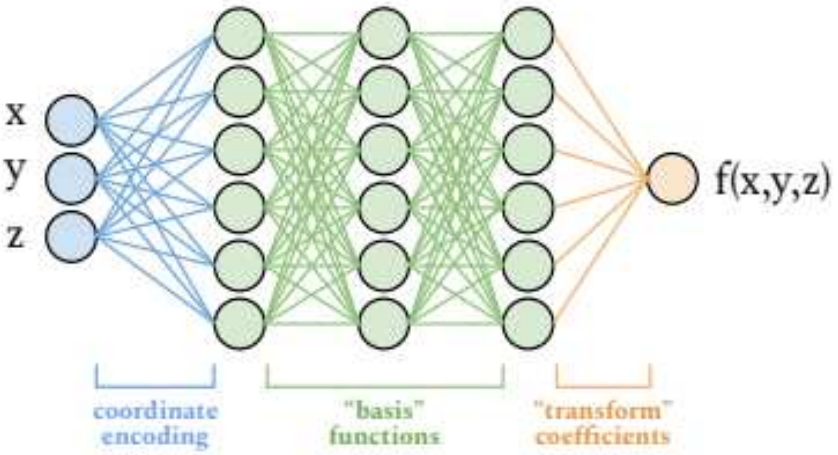


Figure 2.4: Neural implicit representation network mapping input coordinates to signal values, analogous to transform coding.

The realm of Neural Implicit Representations (NIR) compression has witnessed significant contributions, with four seminal papers standing out in recent times. The first, COIN [1], pioneered the concept of employing SIREN networks for compression, focusing on 2D images and 3D shapes. Its successor, COIN++ [2], built upon this foundation, introducing multi-modal compression capabilities, thus broadening the scope to encompass diverse data modalities such as images, videos, and 3D models. Both these papers emphasize the revolutionary idea of storing only the weights of trained networks, highlighting the potential for impressive compression rates. However, while COIN laid the groundwork, COIN++ showcased enhanced flexibility by catering to multiple data types, indicating its potential as a more universal compression tool. Diverging from image and video data, LVAC [20] delved into the challenge of point cloud attribute compression, innovatively employing coordinate-based networks. The paper underscored the intricacies of 3D data and provided a solution that achieved commendable compression rates specifically for point cloud attributes. Lastly, the work on signal compression via neural implicit representations [21] expanded the NIR paradigm to include various signal types, emphasizing the technique's adaptability and robustness. In essence, while all

four papers converge on the core idea of using neural networks for compression, they diverge in their specific methodologies, data types targeted, and underlying architectures. Their collective contributions highlight the vast potential of NIR compression but also underscore the need for specialized approaches depending on the data's complexity and nature.

2.3 Point Cloud Coding

Point cloud data, representing three-dimensional spatial coordinates, has become increasingly vital in various domains such as virtual reality, autonomous driving, medical imaging, and geographical mapping. A point cloud is a collection of data points in a three-dimensional coordinate system, defining the external surfaces of objects or environments. With the burgeoning use of 3D data, the efficient coding and compression of point clouds have become paramount challenges.

Point cloud coding refers to the process of efficiently representing point cloud data in a compressed form. The main goal of point cloud coding is to reduce the data size for efficient storage and transmission while preserving the quality and integrity of the original data. This is a complex task due to the inherent high dimensionality and irregular structure of point clouds, leading to the necessity for sophisticated coding techniques.

Traditional methods for point cloud coding often involve fixed transformation, quantization, and entropy coding. Deep learning-based methods, on the other hand, learn the transformation directly from the data through the network architecture. While traditional methods are effective in certain scenarios, they may struggle to handle the complexity and diversity of point cloud data. Consequently, recent years have witnessed the emergence of novel approaches that leverage machine learning, deep learning, and hybrid techniques to enhance point cloud coding.

The importance of point cloud coding extends beyond mere data compression. It plays a crucial role in enabling real-time processing, visualization, and analysis of 3D data, particularly in applications where bandwidth and storage constraints are significant. Moreover, scalable coding allows for accessing the data at various resolutions, suiting different application needs.

This section will explore various advancements in point cloud coding, focusing on the innovative methods and techniques that have shaped the field in recent years. The discussion will be categorized into different thematic areas, including deep learning-based approaches, loss functions and quantization techniques, standards and improved methods, and synergistic and hybrid approaches.

2.3.1 Deep Learning-based Approaches

Autoencoder-based Approaches

Autoencoders, a specific class of neural networks, have emerged as a powerful tool for point cloud coding. An autoencoder learns to encode the input data into a compressed representation and then decode it back to reconstruct the original data. This ability

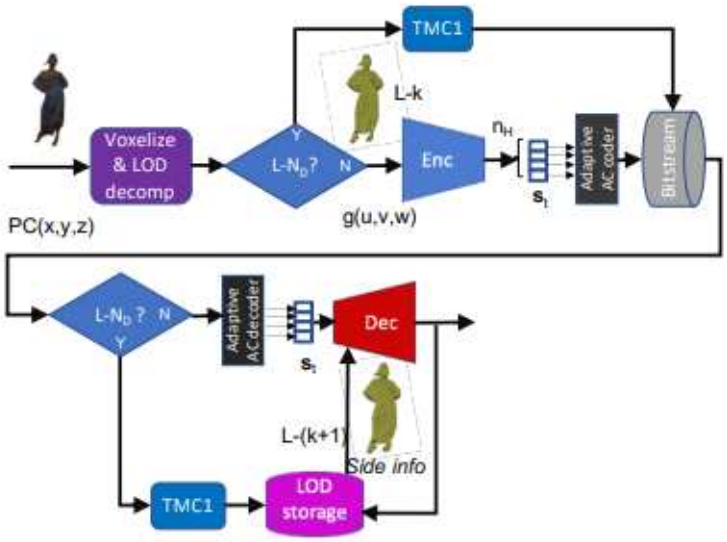


Figure 2.5: Block diagram for the ADAE encoding/decoding scheme. [3]

to learn efficient encodings has been exploited for point cloud compression, offering promising advancements in the field.

Syndrome-Based Autoencoder for Point Cloud Geometry Compression

A notable example is the introduction of a syndrome-based autoencoder for point cloud geometry compression [22]. This novel approach leverages error-correcting codes, known as syndromes, within an autoencoder architecture. By utilizing syndromes, the method efficiently represents point cloud geometry, achieving significant compression rates without losing quality in the reconstructed data. The combination of syndromes with neural network architectures represents a unique advancement, setting a new precedent in data compression.

Adversarial Distributed Source Autoencoder for Point Cloud Compression (ADAE)

Following the aforementioned approach ADAE (Figure 2.5), an Adversarial Distributed Source Autoencoder for point cloud compression [3]. ADAE synergizes autoencoding with adversarial training, creating a robust model capable of handling complex point cloud data. The adversarial component fine-tunes the compression, resulting in remarkable results in both compression rates and reconstruction quality. ADAE's ability to synergistically combine autoencoding with adversarial training makes it a promising tool for various applications where point cloud data is vital.

Scalable Coding Approaches

Scalable coding of point cloud geometry refers to the ability to encode and decode the data at different resolutions, quality levels, or bitrates. This adaptability is essential in various scenarios where the same point cloud data might be accessed and processed at varying levels of detail. Scalable coding approaches have been developed to address this

need, providing flexibility and efficiency in handling point cloud geometries of various complexities.

Deep Learning-based Point Cloud Geometry Coding with Resolution Scalability

A significant advancement in scalable coding is the introduction of a deep learning-based method that emphasizes resolution scalability [23]. This approach offers flexible and efficient coding across various resolutions by employing a hierarchical structure and a combination of convolutional neural networks (CNNs) and other deep learning components. The method ensures that point cloud data can be efficiently coded and decoded at different resolutions without significant loss of quality. This innovation not only enhances compression efficiency but also opens up new possibilities for applications that require access to point cloud data at multiple levels of detail.

Point Cloud Geometry Scalable Coding with a Single End-to-End Deep Learning Model

Another noteworthy approach is the development of a scalable coding method for point cloud geometry using a single end-to-end deep learning model [24]. This integrated model handles the entire scalable coding process, from encoding to decoding, in a streamlined manner. The end-to-end approach allows for flexibility and efficiency, adapting to different point cloud geometries and resolutions. This key innovation in scalable coding offers new opportunities for handling 3D data in various applications, from virtual reality to remote sensing.

Scalable coding approaches provide a versatile solution to the challenges of point cloud geometry compression. By enabling access to the data at various resolutions and quality levels, these methods cater to diverse application needs and scenarios. The integration of deep learning techniques further enhances the adaptability and performance of scalable coding, positioning it as an essential aspect of modern point cloud processing.

Adaptive and End-to-End Learning Approaches

Adaptive and end-to-end learning approaches in point cloud coding represent cutting-edge techniques that focus on flexibility, efficiency, and integrated processing. These methods are designed to tailor their processing to different point cloud geometries dynamically and to handle the complete compression process in an integrated fashion.

Adaptive Deep Learning-Based Point Cloud Geometry Coding

An exemplary method of adaptive coding is the deep learning-based framework presented in [25]. This method dynamically adapts to the complexity and characteristics of different point cloud geometries. By employing a combination of convolutional and recurrent neural networks, it achieves significant improvements in compression efficiency while maintaining high quality in the reconstructed data. The adaptive nature of this method represents a significant advancement, allowing for more efficient and effective coding that responds to the unique challenges of different point cloud geometries.

Lossy Point Cloud Geometry Compression via End-to-End Learning

In [26] the authors present an approach for lossy point cloud geometry compression where the network is optimized end to end. This method leverages deep learning to perform the entire compression process, from encoding to decoding, in a seamless and integrated manner. The end-to-end framework allows the method to adapt to various point cloud geometries, achieving high compression rates while controlling quality loss. This integrated approach simplifies the compression process and offers new possibilities for optimizing the trade-off between efficiency and quality.

Point Cloud Geometry Scalable Coding with a Single End-to-End Deep Learning Model

Building on top of the previous work, [24] introduces a scalable coding method for point cloud geometry using a single deep learning model. This approach integrates the entire scalable coding process into one unified framework, allowing for adaptability across different resolutions and quality levels. This end-to-end method represents a significant step in scalable coding, providing a streamlined solution for various applications.

Adaptive and end-to-end learning approaches offer innovative solutions to the challenges of point cloud coding. By creating models that can adapt to the specific characteristics of point cloud data and handle the entire coding process in an integrated manner, these methods pave the way for further advancements in the field.

2.3.2 Quantization Techniques and Loss Functions

The performance of point cloud compression algorithms is highly dependent on the quantization techniques employed and the loss functions used. These factors control how well the compression algorithm adapts to the specific characteristics of point cloud data. In this section, we delve into the details of both.

Quantization in Point Cloud Coding

Quantization is a crucial step in any compression process, serving to map continuous values to discrete representations. However, incorporating quantization into neural network-based point cloud coding presents challenges due to its non-differentiable nature. This is typically overcome by employing techniques like straight-through estimator (STE), or by adding uniform noise $\mathcal{U}(-0.5, 0.5)$, to enable backpropagation during training.

Another innovative approach is described in [27], where they integrate quantization with loss functions tailored for point cloud data. This ensures a balance between compression efficiency and the quality of the reconstructed point cloud.

Loss Functions

After defining the quantization process, the choice of loss function becomes critical. The loss functions aim to minimize the difference between the original and reconstructed point cloud, subject to the constraints imposed by the quantization process.

Binary Cross-Entropy Loss (BCE):

Often used in point cloud compression, it is defined as:

$$\text{BCE} = -\frac{1}{N}\sum_{i=1}^{N}\left[y_i \log(p(y_i)) + (1-y_i)\log(1-p(y_i))\right]$$

Focal Loss:

It is an extension of BCE and is particularly useful when there is a class imbalance. This is particularly useful when considering voxelized representations since the number of empty voxels is much bigger than the number of filled ones. The focal loss is defined as:

$$\text{Focal Loss} = -\alpha\,(1-p_t)^{\gamma}\log(p_t)$$

Neighborhood Adaptive Loss Function (ω_u):

Introduced in [27], this loss function considers the spatial relationships within the point cloud. It adapts the traditional loss functions by incorporating the geometric and topological features of the point cloud, thus allowing for a more effective and nuanced compression process.

$$\omega_u = \max\left(\frac{1-\text{resemblance}_u}{\max(\text{resemblance})}, 10^{-3}\right).$$

In summary, the choice of quantization techniques and loss functions is pivotal in point cloud coding. By employing novel methods that take into account the spatial and geometric properties of point clouds, significant advancements have been made in the field, paving the way for more efficient and effective point cloud compression.

2.3.3 Standards and Improved Methods

Emerging MPEG Standards

The Moving Picture Experts Group (MPEG) has developed two distinct standards for point cloud compression, one for static point clouds and another for dynamic point clouds. These standards are not based on deep learning but employ traditional methods to achieve robust, scalable, and efficient compression for point cloud data [28]. For static point clouds, the standard focuses on compressing the geometry and attributes efficiently, usually via octree-based methods. For dynamic point clouds, motion compensation techniques are often used to efficiently represent the temporal changes in the data.

These standards aim to offer diverse compression solutions that can adapt to different resolutions and quality levels. They serve as crucial resources for the future of point cloud data handling, set to revolutionize the way point cloud data is stored, transmitted, and utilized in various fields such as entertainment, manufacturing, and healthcare.

Improved Deep Learning-based Methods

Improvements to existing deep learning-based methods for point cloud geometry compression have led to significant advancements in the field [29]. Building upon existing techniques, researchers have focused on optimizing various aspects of the neural network

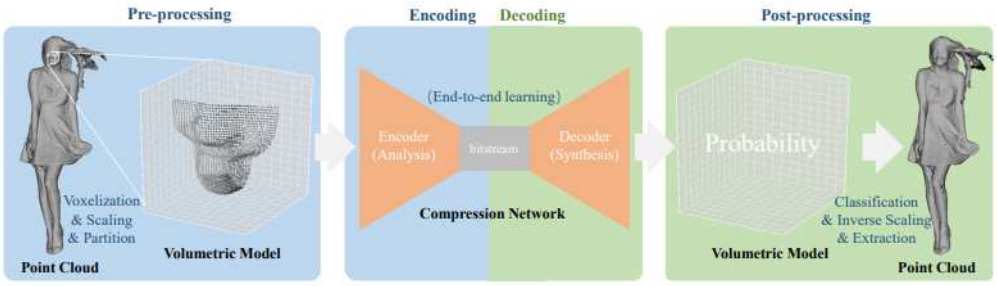


Figure 2.6: An Illustrated overview of the method in Learned-PCGC. [4]

architecture, training procedures, and coding strategies. These systematic enhancements have resulted in optimized compression processes and significant performance gains in terms of compression efficiency and quality preservation. The continued effort to refine and advance deep learning-based point cloud compression methods is paving the way for more efficient and intelligent 3D data processing.

2.3.4 Synergistic and Hybrid Approaches

Synergistic Combination of Autoencoding and Adversarial Training

One noteworthy advancement in point cloud compression is the synergistic combination of autoencoding with adversarial training, as introduced in ADAE [30]. This novel architecture leverages the benefits of both autoencoding and adversarial training to create a robust compression model capable of handling complex point cloud data. By utilizing adversarial training to fine-tune the compression, ADAE achieves impressive results in terms of compression rates and reconstruction quality. This synergistic combination sets a new standard in the field, demonstrating the potential of integrating different deep learning paradigms for point cloud compression.

Hybrid Compression Frameworks

Hybrid compression frameworks that combine machine learning with traditional coding strategies represent another significant innovation in the field [4]. Such approaches leverage neural networks alongside established coding techniques to achieve high compression rates without significant loss of quality. By integrating machine learning into the compression process, these hybrid methods enable models to adapt to different types of point cloud geometries, leading to more flexible and efficient compression. The development of hybrid frameworks opens new possibilities for more intelligent and adaptive 3D data processing.

Learning Convolutional Transforms

The application of convolutional transforms specifically for lossy point cloud geometry compression (Figure 2.6) has also emerged as a promising technique [31]. By employing learning-based techniques, the method enables a more nuanced and adaptable compression process that tailors to the specific characteristics of point cloud data. This

innovation provides a fresh perspective on lossy compression, offering the potential for further advancements in the handling of 3D data.

Synergistic and hybrid approaches in point cloud compression are pushing the boundaries of traditional methods by integrating diverse concepts and techniques. By creating synergies between different deep learning paradigms and combining machine learning with conventional coding, these approaches are paving the way for more innovative and effective solutions. These combinations underscore the versatility of point cloud compression methods and hint at an exciting future for the field, where integration and collaboration between techniques become key drivers of innovation.

2.3.5 Conclusions

The exploration of various methods and approaches in point cloud coding reflects the dynamic and rapidly evolving nature of this field. From the introduction of novel autoencoder-based and scalable coding techniques to the development of adaptive and end-to-end learning approaches, the landscape of point cloud compression is witnessing significant advancements.

- **Innovation in Architectures**: The emergence of synergistic combinations, such as the integration of autoencoding with adversarial training and the development of hybrid frameworks, demonstrates the creativity and adaptability within the field. These innovations in architectures offer improved efficiency, adaptability, and quality in point cloud compression.
- **Emphasis on Scalability and Adaptability**: The focus on scalable coding and adaptive learning reflects the growing need for flexibility in handling various resolutions, quality levels, and complexities in point cloud data. Methods that offer resolution scalability or dynamically adapt to different geometries are paving the way for more intelligent and responsive compression solutions.
- **Advancements in Loss Functions and Quantization**: The formulation of specialized loss functions and the integration of implicit and explicit quantization techniques underline the nuanced understanding of the geometric properties of point clouds. These advancements contribute to optimized compression that preserves quality while achieving high efficiency.
- **Emergence of Standards and Improved Methods**: The development of MPEG standards and continuous improvements to existing methods signify the maturation and standardization in the field. These efforts are essential in shaping the future of point cloud data handling across various industries and applications.
- **Exploration of Hybrid Approaches**: The blend of traditional coding strategies with machine learning and deep learning techniques showcases the potential for collaboration between different paradigms.

In conclusion, the field of point cloud coding is marked by a rich diversity of methods, a relentless pursuit of innovation, and a clear focus on addressing the unique challenges of 3D data handling. The insights gained from the reviewed papers underline the potential

and vitality of this field, setting the stage for continued growth and exploration. As technology advances and the demand for efficient 3D data processing grows, point cloud coding will undoubtedly remain a vibrant and essential area of research and development.

2.4 Quantization Techniques for model compression

Quantization techniques play a critical role in optimizing neural network models for both performance and storage. By approximating the continuous-valued weights and biases, quantization methods facilitate the efficient deployment of machine learning models, particularly in resource-constrained environments. This section delves into the three primary categories of quantization techniques: Dynamic Quantization, Static Quantization, and Quantization-Aware Training. Each technique offers distinct advantages and trade-offs that are crucial for effective model compression and deployment.

2.4.1 Static Quantization

Static quantization, as the name suggests, involves the quantization of model parameters prior to the inference stage. Unlike dynamic quantization, which performs quantization during inference, static quantization transforms both the weights and the activations in the neural network during the training or pre-deployment phase. This results in a model that is fully quantized and ready for efficient execution, with no additional computational overhead during inference.

The primary advantage of static quantization lies in its performance gains. By quantizing the model beforehand, it eliminates the need for on-the-fly calculations, making it highly suitable for real-time and latency-sensitive applications. Additionally, static quantization often involves fine-tuning the quantized model, which helps in preserving the accuracy to a significant extent.

However, one drawback of static quantization is its lack of adaptability to varying data distributions. Since the quantization is performed before inference, the model may not be as flexible in handling inputs with different statistical properties, which can be a limitation in scenarios involving complex data types like 3D point clouds.

Static quantization has been widely adopted in various machine learning applications, ranging from computer vision to natural language processing and has recently found applications in point cloud compression as well.

2.4.2 Dynamic Quantization

Dynamic quantization is a technique employed predominantly during the inference phase of neural network models. Unlike static quantization, which quantizes the weights before inference, dynamic quantization performs this operation on-the-fly as the model processes the input data. The primary advantage of this approach is its ability to adapt to the statistical properties of the input, thereby achieving more efficient compression without significant loss of quality.

One of the key benefits of dynamic quantization is its minimal impact on model

accuracy. Since the quantization occurs during inference and adapts to the specific input, it allows for a more nuanced approximation of continuous-valued parameters. This flexibility makes it particularly useful for models dealing with heterogeneous or complex data distributions, such as 3D point clouds.

However, dynamic quantization is not without its limitations. The on-the-fly nature of this technique can introduce computational overhead, making it less suitable for real-time applications or systems with stringent latency requirements.

Recent advances in dynamic quantization have explored its application in various domains, including image and video compression, natural language processing, and, importantly for this thesis, point cloud compression.

2.4.3 Quantization-Aware Training (QAT)

Quantization-Aware Training (QAT) is an advanced approach that incorporates the quantization process directly into the model training phase. Unlike static and dynamic quantization, which are applied either before or during inference, QAT aims to simulate the effects of quantization during training itself. This involves the use of fake quantization layers that mimic the behavior of quantized operations, thereby allowing the model to adapt to the reduced numerical precision in a more robust manner.

One of the key advantages of QAT is its ability to fine-tune the model for quantization while it is being trained. This usually results in better model accuracy compared to post-training quantization methods. It is particularly useful in scenarios where the model needs to be highly optimized for both speed and accuracy, as in the case of point cloud compression algorithms that deal with complex 3D geometries.

However, the downside of QAT is that it may prolong the training process, as the model has to adapt to additional constraints imposed by the fake quantization layers. But this trade-off is often justified by the gains in model performance and the reduced need for model fine-tuning post-quantization.

QAT has been effectively employed in various machine learning applications, including neural networks designed for point cloud compression, and has shown to yield impressive results in terms of model efficiency and accuracy.

3

Methodology

This chapter presents the various algorithms and methodologies developed to extend and improve the baseline approach.

3.1 Dataset

An understanding of the dataset forms the foundation for the deep learning model's training. This section offers an overview of the dataset's characteristics and sources.

The point clouds (PCs) analyzed in this study were sourced using diverse acquisition methods to ensure the representation of various noise types that a universal codec might confront.

The dataset samples are categorized as follows:

- **High-Quality Models:** The dataset includes 6 superior quality PCs as depicted in Figure 3.1. These were extracted from the MPEG dataset [32]. A set of 39 synchronized RGB cameras captured dynamic PCs, producing 300 frames at 30 fps. Only one frame was selected for the static model.

- **Computer-Generated Models:** Given that a significant portion of 3D models in the industry is computer-generated, a PC in the dataset was derived by sampling a mesh (see Figure 3.2). This synthetic signal is anticipated to be the least noisy and thus, should present fewer challenges for the codec.

- **Structure from Motion (S.f.M.) PCs:** S.f.M. is a predominant PC acquisition method known to yield sparse and noisy data, as showcased in Figure 3.3. Including these samples ensures the neural network is trained to manage such data efficiently.

- **Laser Scanner PCs:** Recognized for their precision, laser scanners are extensively used in various fields, such as architectural acquisition. Therefore, PCs derived using this method are incorporated in the dataset, with examples provided in Figure 3.4.

Despite the presence of color information in these point clouds (PCs), our focus is primarily on their geometric structures. Hence, for the purposes of this study and the input to our model, we exclusively consider the spatial positions of the points, disregarding their associated color data.

(a) long **(b)** redandblack **(c)** soldier

(d) thai dancer **(e)** boxer **(f)** loot

Figure 3.1: Point Clouds from the MPEG dataset

Figure 3.2: Queen: PC sampled from a 3D mesh

3.2 DiGS Network Model Review

The representation and processing of 3D point cloud data have become increasingly important in various fields, including computer vision, robotics, and graphics. While advancements in neural networks have facilitated progress in point cloud processing, a significant challenge has been the accurate and efficient representation of unoriented point clouds that lack consistent orientation or normal data. Addressing this challenge, the paper "DiGS: Divergence Guided Shape Implicit Neural Representation for

(a) Facade15 **(b)** House

Figure 3.3: Point Clouds obtained with the structure from motion technique

(a) Facade09 **(b)** Facade64

(c) Shiva **(d)** Frog

Figure 3.4: Point Clouds obtained by using a laser scanner.

Unoriented Point Clouds" by Ben-Shabat et al [15]. presents an innovative model that combines the strengths of implicit neural representations with divergence-guided techniques. This section delves into the nuances of the DiGS model, its methodology, and its significance in the realm of point cloud representation.

3.2.1 Objective of the Paper

The heart of the paper lies in the introduction and exploration of the DiGS (Divergence Guided Shape) model. Point clouds, while being a rich source of 3D information, often come without consistent orientation or normal data. Such unoriented point clouds pose a unique set of challenges in terms of representation and processing. The primary aim of the DiGS model is to offer a robust solution to this problem, enabling accurate representation of these point clouds using implicit neural representations. By focusing on unoriented point clouds, the authors address a significant gap in the current literature and technologies, aiming to enhance the versatility and accuracy of point cloud processing techniques in various applications.

3.2.2 DiGS algorithm

As was anticipated in the previous chapters, an implicit representation for a 3D shape implements a function $f : \mathbb{R}^3 \to \mathbb{R}$, where the shape S is defined by the zero level-set of f, expressed as $S = \{x \in \mathbb{R}^3 | f(x) = 0\}$.

Neural networks, with their profound expressive capabilities, can be harnessed to model the function f. Such networks, when trained, accept a 3D coordinate as input and produce a scalar value in return. The training paradigm typically revolves around minimizing the discrepancy between the network's output and a predetermined scalar value for a set of 3D coordinates. For instance, inside the shape, the target could be -1, while outside, it might be +1.

The DiGS model integrates neural implicit representations as its foundational mechanism. However, it incorporates further innovations, like divergence-based analysis, to amplify the capabilities of standard neural implicit representations, particularly for unoriented point clouds (see Figure 3.5).

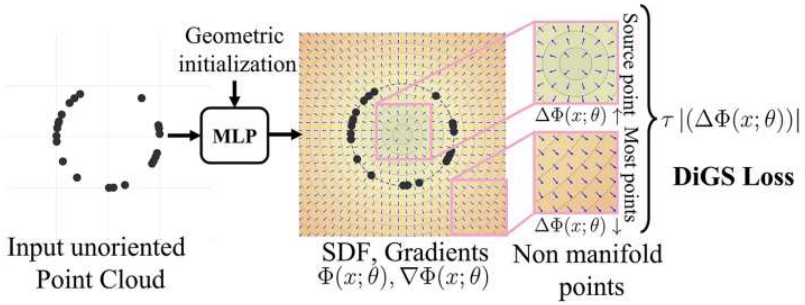


Figure 3.5: An example in 2D: An implicit neural representation of a shape is trained from an input point cloud that lacks orientation. The training combines an initial geometric setup with a divergence penalty loss, based on the notion that the divergence of the signed distance function is generally low across most areas.

By synergizing the flexibility of implicit representations with the prowess of neural networks, the DiGS model presents a robust and efficient modality for representing unoriented point clouds. The innate benefits of implicit representations, amalgamated with the paper's novel contributions, position the DiGS model as a pivotal stride in the

discipline.

The DiGS model introduces a systematic approach to constructing neural implicit representations for unoriented point clouds. They use a smooth-to-sharp approach that keeps the gradient vector field highly consistent during training (see Figure 3.6). This improves the convergence speed of the model and acts as a regularizer allowing it to generally obtain more precise surfaces. This approach, which hinges on divergence-based analysis, can be broken down into four distinct steps:

1. Shape Initialization

The first step involves initializing a shape implicit function from the given unoriented point cloud. This process employs the distances from the point cloud to a base shape, often a sphere, ensuring that the initialized function is well-behaved and has a meaningful gradient.

2. Neural Network Parameterization with SIREN

To represent the initialized shape implicit function, the DiGS model leverages the Sinusoidal Representation Networks (SIREN). SIREN is specifically designed to represent intricate functions, making it an ideal choice for point cloud data. It uses sinusoidal activation functions, ensuring that the network remains highly expressive and can effectively capture the details of the point cloud.

3. Divergence Regularization

With the network in place, the next step is the actual training process. A critical aspect of this training is divergence regularization. By adding a penalty based on the divergence of the gradient of the implicit function, the model ensures that the implicit function accurately captures the structure of the point cloud. This divergence-based regularization is instrumental in enabling the model to handle unoriented point clouds effectively.

4. Shape Refinement

Post-training, the model undergoes a refinement process. This step ensures that the resultant implicit function is not just a mere approximation but a precise representation of the input point cloud.

The entire process, from initialization to refinement, underscores the model's emphasis on capturing the intricacies of unoriented point clouds. The divergence-based approach, combined with the power of SIREN, sets the DiGS model apart, offering a robust and efficient solution for point cloud representation.

3.3 Rationale for Choosing the DiGS Network Model

The selection of an appropriate neural network model is crucial for the success of any machine learning-based project. In the context of point cloud geometry compression, it

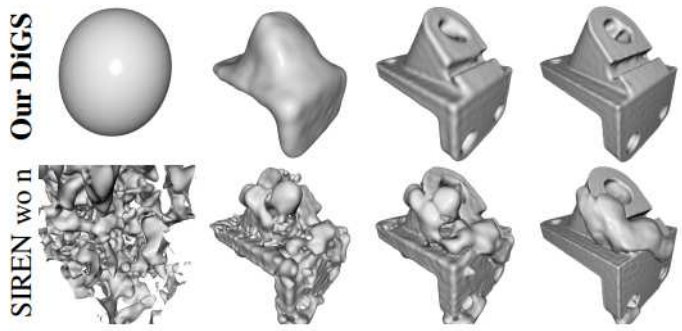


Figure 3.6: Progressive results across four training iterations for both DiGS (upper row) and SIREN without normalization (lower row), moving from earlier iterations on the left to later ones on the right.

becomes especially vital given the high computational requirements and the intricacies involved in 3D data handling. After thorough consideration and experimentation, we chose the DiGS Network model as the foundational architecture for our work. Below, we elaborate on the specific reasons for this selection.

Convergence Speed

One of the most critical factors in the choice of a neural network model is its ability to converge quickly. Slow convergence can dramatically increase the time required for model training, thereby slowing down the entire research and development process. DiGS Network was originally designed with convergence speed in mind, in particular, the weight initialization strategy and the regularizers allow it to learn the required shape very quickly. Because the training time is in fact the encoding time this property of the DiGS model makes it the ideal choice for this project.

Versatility in Handling Unoriented Point Clouds

Our project's scope necessitates a model capable of handling unoriented point clouds, a common form of 3D data. The DiGS Network's inherent ability to accept unoriented point clouds as input and construct a Neural Implicit Representation (NIR) allows for considerably increased decoding speed, w.r.t. a network that simply predicts occupancy because it allows to avoid sampling in regions where the predicted distance from the surface is high. Furthermore, the original DiGS model outputs the reconstructed surface as a Mesh, aligning closely with our project's requirements. However, we adapted the output from Mesh to Point Cloud to better suit our specific needs.

In summary, the DiGS Network model was selected due to its fast convergence, efficient training time, and flexibility it offers in handling various forms of 3D data, particularly unoriented point clouds. These characteristics make it an excellent fit for our research objectives, and we anticipate that it will significantly contribute to the successful completion of this project.

3.4 Reconstruction of Point Cloud

The reconstruction process plays a critical role in the conversion of Neural Implicit Representation (NIR) back to the point cloud (PC) format. The original DiGS Network model was designed to produce meshes as the output. However, for the needs of our research, we have adapted the model to generate point clouds. The function responsible for this transformation is `Implicit2pc()`.

The `Implicit2pc()` function uses the Marching Cubes algorithm for extracting a polygonal mesh of an isosurface from a three-dimensional discrete scalar field.

The implementation is adapted from a paper by Atzmon and Lipman [33], titled "SAL: Sign Agnostic Learning of Shapes from Raw Data." The original code for the SAL paper is available on GitHub.

3.4.1 Arbitrary Resolution Reconstruction

One of the major advantages of using the DiGS Network for point cloud reconstruction is the ability to generate point clouds at arbitrary resolutions. This flexibility allows us to make the point cloud as dense as we wish, which is beneficial for applications requiring high fidelity. Conversely, there are scenarios where speed is more crucial than fidelity; in such cases, the model can be adjusted to perform faster reconstructions at the cost of lower density.

The flexibility to control the density of the reconstructed point cloud is particularly advantageous in real-world applications such as Augmented Reality (AR) and Virtual Reality (VR). Consider a scenario where a viewer is using AR/VR glasses to interact with a point cloud object. When the viewer is at a distance from the object, a high-resolution point cloud is unnecessary because the details would not be discernible to the viewer. In this case, a lower-density point cloud would suffice, speeding up the reconstruction process and reducing computational load.

However, as the viewer approaches the object, the need for a more detailed, high-resolution point cloud becomes evident. Being able to adjust the resolution of the point cloud dynamically according to the viewer's proximity to the object is therefore an essential feature. This adaptability in resolution is made possible through our arbitrary resolution reconstruction approach, which allows for the generation of point clouds with varying densities to suit the specific requirements of the application.

3.5 Dynamic Quantization of the DiGS Model

Dynamic quantization serves as a powerful tool for model compression without significantly compromising the performance or accuracy of the model. In the context of the DiGS Network, the primary focus of utilizing dynamic quantization lies in reducing the size of the trained model. This becomes particularly important when dealing with large and complex 3D point clouds, where both computational cost and storage requirements are substantial.

3.5.1 Implementation Details

Dynamic quantization allows us to adjust the quantization level arbitrarily, enabling us to control the bit-depth allocated for each value in the model. Originally, each value is represented using 32 bits. By quantizing these values to fewer bits, we achieve a substantial reduction in the model size. Specifically, we can set the bit-depth to any arbitrary number of bits, thereby offering flexibility in determining the compression level.

This process involves two steps:

1. **Quantization**: In this phase, the 32-bit floating-point numbers in the model are quantized to a lower bit-width. This quantized model is then used for the compression of the point cloud.

2. **Storing Metadata for Dequantization**: Alongside the quantized model, it's essential to store metadata that holds the boundary values for each quantized parameter. This metadata is critical for the dequantization process, allowing us to reconstruct the 32-bit values from their quantized forms.

3.5.2 Bitrate Considerations

The size of the quantized model divided by the number of points in the original point cloud yields the 'bits per point' (bpp) metric. This value serves as a measure of the compression efficiency, indicating how many bits are used to represent each point in the compressed point cloud, and therefore it is used as the measure for the rate.

By employing dynamic quantization in the DiGS model, we gain the ability to control the rate, thereby optimizing the trade-off between model size and reconstruction quality.

3.6 Evaluation Metrics and Assessment Methodology

3.6.1 PSNR as a Metric

Peak Signal-to-Noise Ratio (PSNR) is a commonly used metric for assessing the quality of reconstruction in image and signal processing. In the context of 3D point clouds, PSNR serves to quantify the quality of the reconstructed point cloud as compared to the original.

In particular, in point clouds, one of the most adopted metrics is the PSNR_D1 which can be calculated using the following mathematical formulation:

Let PC1 and PC2 be the two point clouds being compared, each containing n points in $\mathbb{R}^3$. Let scale be an optional scale factor. If scale is not provided, it is calculated as the maximum coordinate range of PC1.

The distance between each point in PC2 and its nearest neighbor in PC1, denoted as d_{1i}, is calculated. Similarly, the distance between each point in PC1 and its nearest neighbor in PC2, denoted as d_{2i}, is calculated.

The mean squared error (MSE) for these distances is calculated as:

$$\text{MSE1} = \frac{1}{n}\sum_{i=1}^{n} d_{1i}^2, \quad \text{MSE2} = \frac{1}{n}\sum_{i=1}^{n} d_{2i}^2$$

Finally, the PSNR_D1 is calculated as:

$$\text{PSNR\_D1} = 10\log_{10}\left(\frac{\text{scale}^2}{\max(\text{MSE1}, \text{MSE2})}\right)$$

This metric serves as a quantitative measure of the fidelity of the reconstructed point cloud in relation to the original.

3.6.2 Quality per Compression

To thoroughly evaluate the performance of our model, we plot PSNR against bpp. This plot serves to illustrate the trade-off between rate and distortion, allowing us to identify the optimal operating point where we achieve a balance between a small model size and a high reconstruction quality based on the application requirements.

3.7 Role of Residuals in Compression

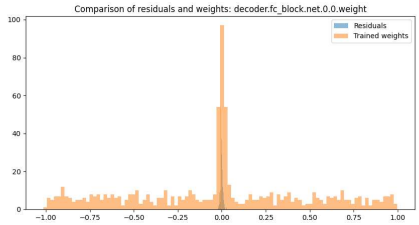
In our initial experiments, we focused on compressing the entire point cloud using neural implicit representations. However, the performance, as measured by PSNR versus bits-per-point (bpp), was not up to efficiently correlated to the human perception. This led us to explore alternative approaches to improve the efficiency of our compression algorithm.

After extensive experimentation, we shifted our focus to compressing residuals. In this context, residuals are defined as the differences between the initial model, which is initialized deterministically thanks to a fixed procedure and seed, in the DiGS Network (see Figure 3.6), and the final learned model. We found that residuals had lower entropy thus resulting in a lower rate when keeping distortion constant, thereby making the compression more effective.

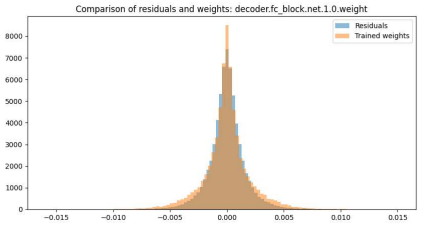
By compressing the residuals rather than the entire point cloud, we were able to achieve a more balanced and effective compression scheme, with improved PSNR per bpp metrics.

3.7.1 Why Use Residuals?

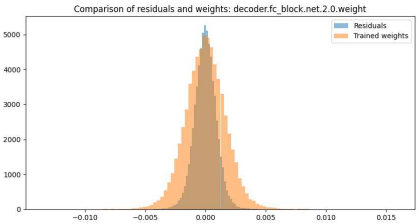
Quantization techniques are more effective when the range of data values is smaller. Because with the same amount of bits you can represent the distribution of values with smaller errors. During our experiments, we observed that the distribution of residuals has a smaller standard deviation compared to the distribution of actual model weights. This observation led us to the idea of compressing the residuals instead of the original model weights.



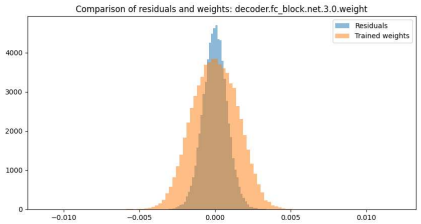
(a) decoder fc block net 0.0 weight



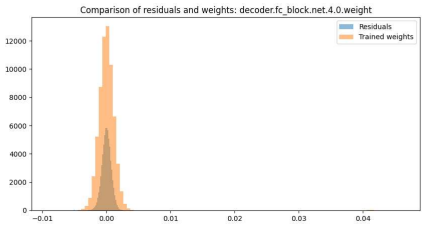
(b) decoder fc block net 1.0 weight



(c) decoder fc block net 2.0 weight



(d) decoder fc block net 3.0 weight



(e) decoder fc block net 4.0 weight

Figure 3.7: Comparison of residuals and actual weights.

3.7.2 Methodology

Our approach is as follows:

1. Compress the residuals using dynamic quantization.
2. Store these compressed residuals.
3. During the dequantization process, we decompress the residuals and add them back to the initial model, which is initialized deterministically thanks to a fixed procedure and seed, to reconstruct the final learned model.

3.7.3 Visual Evidence

We offer visual evidence to support our approach. Five charts comparing the distribution of residuals and actual weights for different layers are presented in Figure 3.7:

This clearly supports our choice of compressing residuals for more efficient and effective model compression. As you can see, for most layers, the range of residuals is smaller than the weights themselves.

4

Experiments and Results

In this chapter, we present a detailed account of the experiments conducted to validate the efficacy of our proposed method for point cloud geometry compression using neural implicit representations. The primary objective is to evaluate the performance of the trained DiGS network model, especially in terms of compression ratios and reconstruction quality. To this end, we provide an exhaustive explanation of the experimental setup, model configurations, applied quantization techniques, and the metrics used for assessment. Subsequently, we reveal the results and offer an in-depth analysis, comparing them with existing methods and drawing insightful conclusions.

4.1 Experimental Setup

The experimental setup serves as the foundation upon which the entire study is built. It is crucial to provide a thorough description to ensure the reproducibility of the results. We begin by elaborating on the hardware and software configurations employed during the experiments.

The computational experiments were conducted on a dedicated machine equipped with an Intel® Core™ i7-7700 CPU that operates at a clock speed of 3.60 GHz and offers 8 cores. The machine is further enhanced with an NVIDIA GeForce GTX 1070 Graphics Processing Unit (GPU), which significantly accelerates the model training and inference processes. To handle large datasets and facilitate complex computations, the system is provisioned with 32 GB of Random Access Memory (RAM).

Our implementation is built on Python 3.7, chosen for its extensive libraries and community support for scientific computing and machine learning tasks. Several specialized libraries were utilized to implement different aspects of the project. For numerical operations, we relied on the `numpy` library, version 1.19.2. Visualization tasks were accomplished using `matplotlib` version 3.3.4, while scientific computations leveraged the `scipy` library, version 1.6.2. We used machine learning utilities that were covered by `pytorch`.

The dataset used in this study has been elaborated upon in Chapter 3. We refer the reader to that chapter for a comprehensive understanding of its characteristics and how it aligns with the objectives of this research.

4.2 Model Configuration

Understanding the architecture and hyperparameters of the DiGS network is pivotal for comprehending the experimental results and ensuring the reproducibility of this study. In this section, we delineate the configuration of the DiGS network, focusing on its architecture, layers, and hyperparameters.

4.2.1 Architecture

The DiGS network is primarily composed of an optional encoder and a decoder. The encoder can either be an 'autodecoder' or be completely omitted ('none'), based on the configuration. The core of the model lies in its decoder, which is implemented as a fully connected neural network.

Decoder

The decoder is constructed as a fully connected network with varying numbers of hidden layers, controlled by the parameter `decoder_n_hidden_layers`. Each hidden layer is followed by a non-linearity, the type of which is specified by the parameter `nl`. The decoder takes as input the 3D coordinates concatenated with a latent vector, if an encoder is used. The output is a single channel, representing the implicit field value at the input coordinates.

4.2.2 Hyperparameters

The DiGS network allows for a variety of hyperparameters to be configured. Some of the key hyperparameters include:

- **Latent Size (`latent_size`)**: Determines the size of the latent vector.
- **Decoder Hidden Dimension (`decoder_hidden_dim`)**: Specifies the number of neurons in the hidden layers of the decoder.
- **Non-Linearity (`nl`)**: Defines the type of non-linear activation function used in the network.
- **Number of Hidden Layers in Decoder (`decoder_n_hidden_layers`)**: Controls the depth of the decoder network.
- **Initialization Type (`init_type`)**: Determines the weight initialization strategy for the network.
- **Number of Epochs (`num_epochs`)**: Specifies the number of training cycles.

Initialization

The initialization of the network weights is particularly crucial for the training dynamics and model performance. The DiGS network supports multiple initialization strategies

like 'siren', 'geometric_sine', and 'mfgi', each serving specific purposes and offering different characteristics during training.

4.2.3 Specific Configurations

For a concrete understanding, we specify the exact configurations used in our experiments. The following hyperparameters were chosen based on empirical testing and the requirements of the model:

Hyperparameter	**Value**
Latent Size (`latent_size`)	0 (for reconstruction)
Decoder Hidden Dimension (`decoder_hidden_dim`)	256
Non-Linearity (`nl`)	sine
Number of Hidden Layers in Decoder (`decoder_n_hidden_layers`)	4
Initialization Type (`init_type`)	mfgi
Number of Epochs (`num_epochs`)	5000

Table 4.1: Specific hyperparameters and configurations used in the experiments.

It is worth noting that a latent size of zero indicates that no latent variables were used in the model for the task of point cloud reconstruction. This configuration aligns with our focus on using the weights of the network as a mean to represent the point cloud and eliminates the influence of latent variables on the reconstruction task.

4.3 Results

This section presents the empirical results obtained from the experiments. We focus on multiple aspects including the performance of trained models before quantization, the effect of dynamic quantization on these models, and a comparative analysis with baseline methods. Each subsection is accompanied by tables and plots to provide a comprehensive view of the findings.

4.3.1 Performance Metrics After Training

To evaluate the initial reconstruction capabilities of the DiGS network model, we calculated the Peak Signal-to-Noise Ratio (PSNR) for different 3D models after training but before applying any quantization steps. These results serve as a baseline for understanding the inherent quality of the reconstructed point clouds.

The table and figure elucidate that the PSNR values vary depending on the complexity and characteristics of the 3D model. Higher PSNR values generally indicate better reconstruction quality, serving as an initial indicator of the model's performance.

4.3.2 Performance Metrics After Quantization and Dequantization

For each model, we have generated individual tables to delve deeper into their performances. These tables, like the one for the 'thai' model shown in Table 4.3, provide

3D Model	PSNR_D1 (dB)
House	35.193
Long	63.374
Soldier	54.253
Frog	39.238
Shiva	40.765
Thai	55.343
Boxer	60.250
Facade09	44.117
Facade15	39.747
Loot	47.834
Queen	50.088
RedandBlack	56.339

Table 4.2: PSNR values for different 3D models after training the DiGS network model, but before quantization.

a more granular view. The tables show the performance metrics at various bitrates, helping to ascertain the optimal operating point for each model.

Table 4.3: Performance Metrics of thai

Num Bits	Compressed Model Size (Bytes)	PSNR D1 (dB)	Bitrate (bpp)
2	181513	24.45	1.48
4	250424	30.03	2.04
8	447589	50.17	3.65
16	751972	50.46	6.13

Table 4.4: Performance Metrics of soldier

Num Bits	Compressed Model Size (Bytes)	PSNR D1 (dB)	Bitrate (bpp)
2	169256	24.32	1.24
4	232933	33.51	1.71
8	425053	48.62	3.12
16	741268	48.73	5.45

4.3.3 Comparison with Baselines

In our evaluations, we compared our models against established benchmarks such as DRACO and G-PCC. Figure 4.2 shows that both DRACO and G-PCC outperform our approach in terms of PSNR across different bitrates. One contributing factor to this is the quality of the original Neural Implicit Representation (NIR) generated by the DiGS model, which was not as high as anticipated.

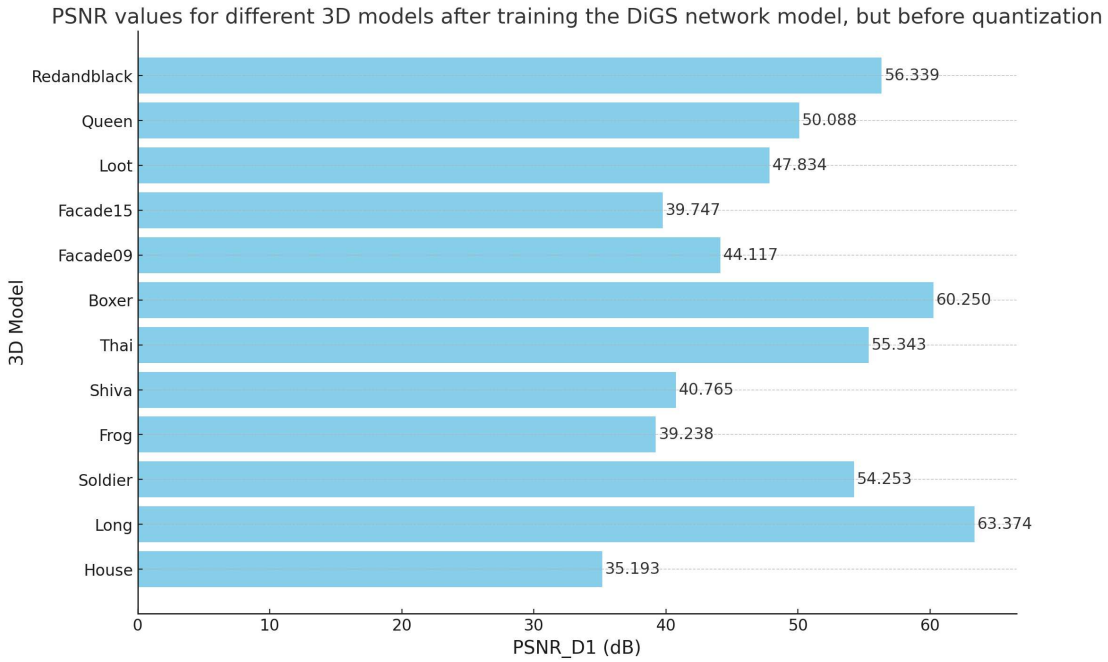


Figure 4.1: Graphical representation of PSNR values for different 3D models after training.

Table 4.5: Performance Metrics of shiva

Num Bits	Compressed Model Size (Bytes)	PSNR D1 (dB)	Bitrate (bpp)
2	170237	21.73	1.51
4	240926	36.85	2.14
8	429156	40.27	3.81
16	741416	40.29	6.59

4.3.4 Summary

Our work contributes to the point cloud compression literature by introducing a novel approach that focuses on the compression of Neural Implicit Representations (NIRs). However, as seen in Figure 4.2, our models have limitations in terms of PSNR when compared to traditional methods like DRACO and G-PCC. These findings suggest that improving the initial NIR quality could be a key direction for future research to enhance the performance of our approach.

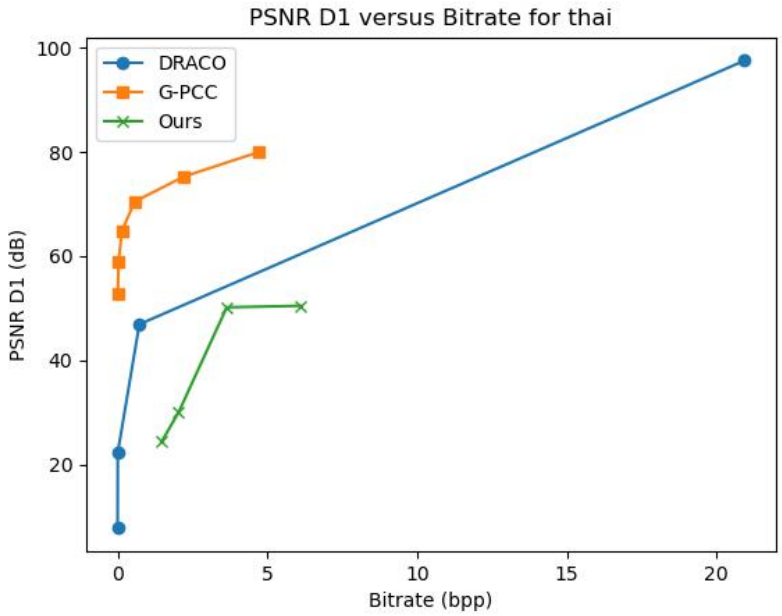


Figure 4.2: Performance measurement on thai.

Table 4.6: Performance Metrics of redandBlack

Num Bits	Compressed Model Size (Bytes)	PSNR D1 (dB)	Bitrate (bpp)
2	153060	23.84	1.62
4	245028	32.17	2.59
8	443213	50.86	4.68
16	750763	51.56	7.93

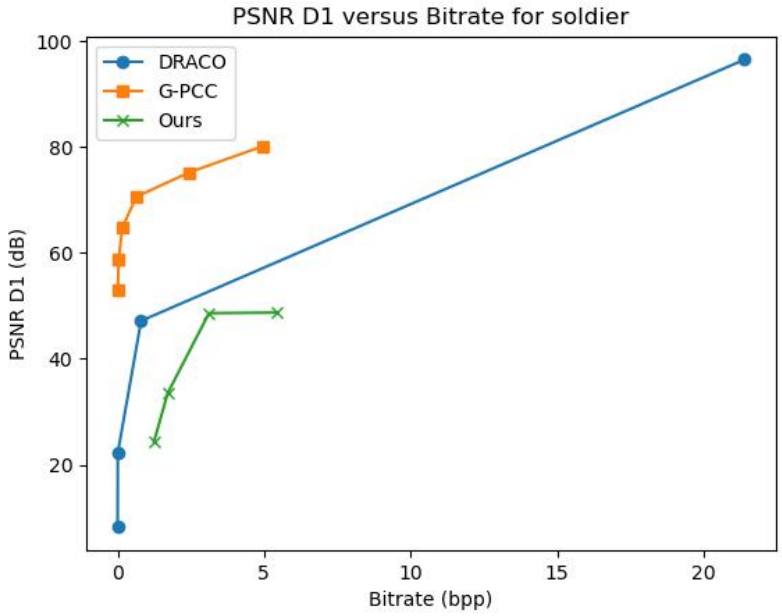


Figure 4.3: Performance measurement on soldier.

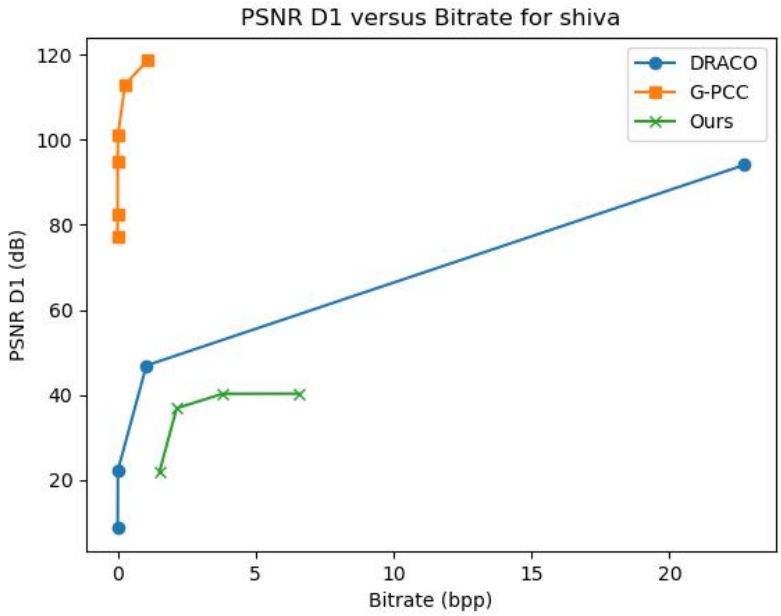


Figure 4.4: Performance measurement on shiva.

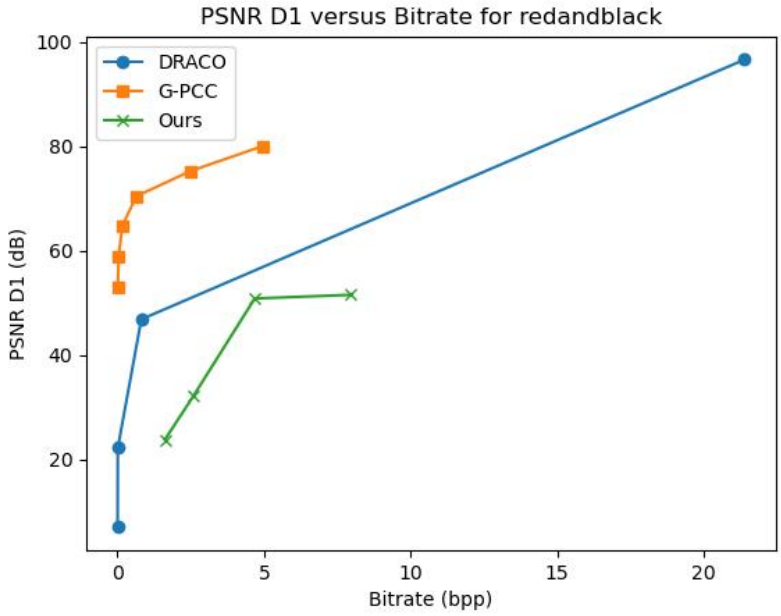


Figure 4.5: Performance measurement on redandBlack.

Table 4.7: Performance Metrics of queen

Num Bits	Compressed Model Size (Bytes)	PSNR D1 (dB)	Bitrate (bpp)
2	178397	25.70	1.43
4	240947	30.14	1.93
8	424278	48.73	3.39
16	741151	48.81	5.92

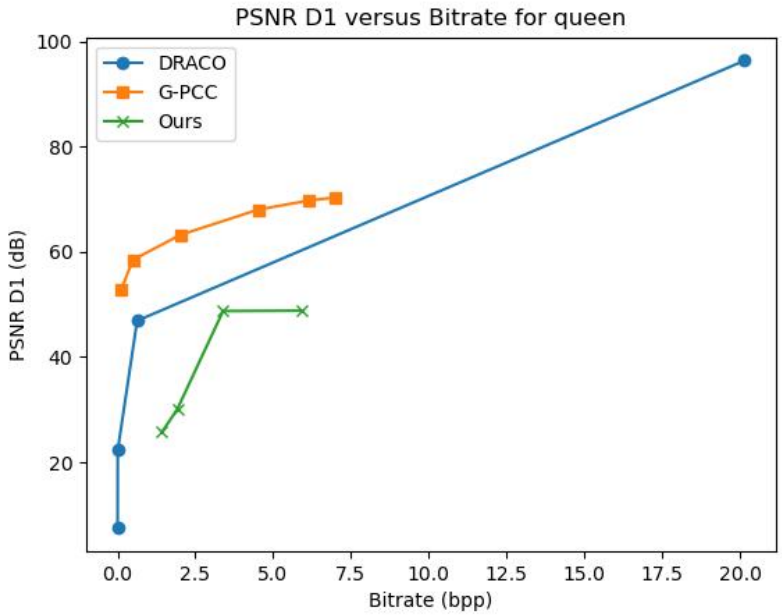


Figure 4.6: Performance measurement on queen.

Table 4.8: Performance Metrics of loot

Num Bits	Compressed Model Size (Bytes)	PSNR D1 (dB)	Bitrate (bpp)
2	166812	24.62	1.66
4	247830	31.42	2.46
8	439783	46.37	4.37
16	750556	46.44	7.46

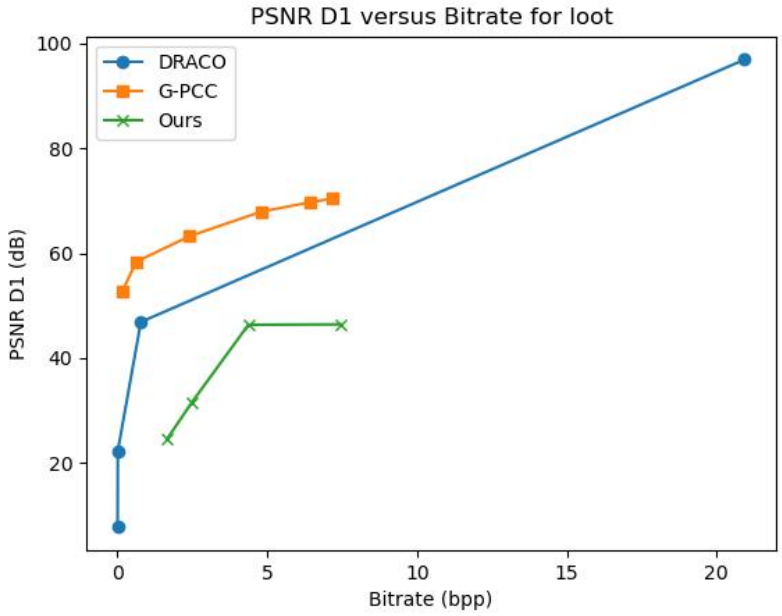


Figure 4.7: Performance measurement on loot.

Table 4.9: Performance Metrics of long

Num Bits	Compressed Model Size (Bytes)	PSNR D1 (dB)	Bitrate (bpp)
2	165733	28.22	1.55
4	240451	32.35	2.24
8	436023	54.49	4.07
16	747578	54.61	6.97

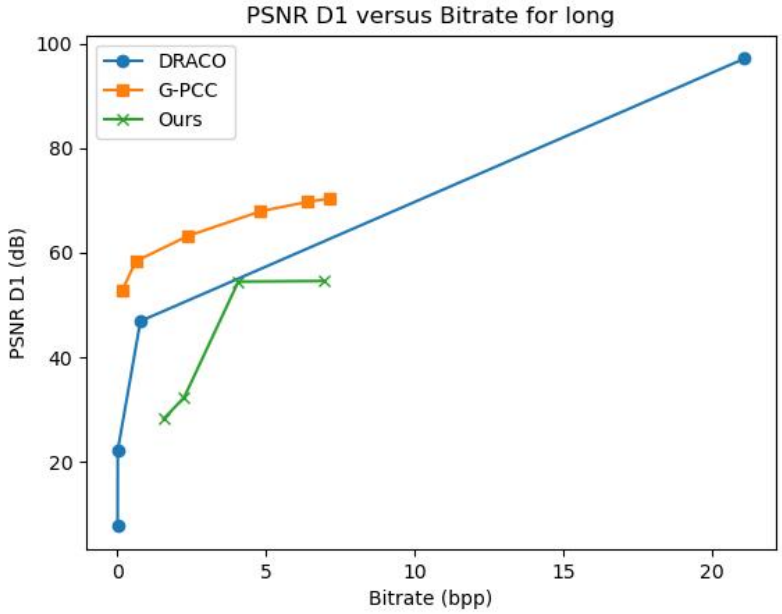


Figure 4.8: Performance measurement on long.

Table 4.10: Performance Metrics of house

Num Bits	Compressed Model Size (Bytes)	PSNR D1 (dB)	Bitrate (bpp)
2	153765	17.89	0.71
4	219378	33.06	1.02
8	420082	34.86	1.95
16	739323	34.89	3.43

Table 4.11: Performance Metrics of frog

Num Bits	Compressed Model Size (Bytes)	PSNR D1 (dB)	Bitrate (bpp)
2	152600	20.48	0.77
4	232233	37.17	1.17
8	434068	39.08	2.20
16	746808	39.12	3.78

Table 4.12: Performance Metrics of facade15

Num Bits	Compressed Model Size (Bytes)	PSNR D1 (dB)	Bitrate (bpp)
2	175558	21.20	1.69
4	249943	39.18	2.40
8	435191	40.23	4.19
16	745874	40.22	7.18

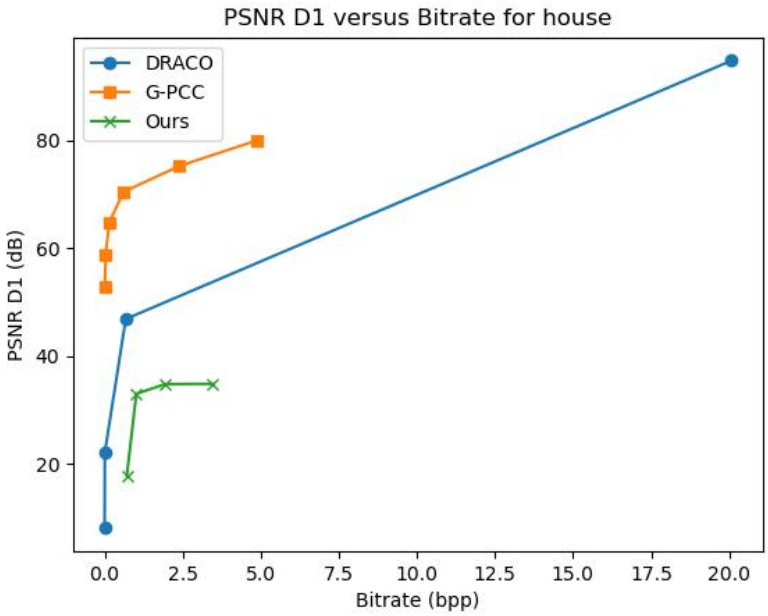


Figure 4.9: Performance measurement on house.

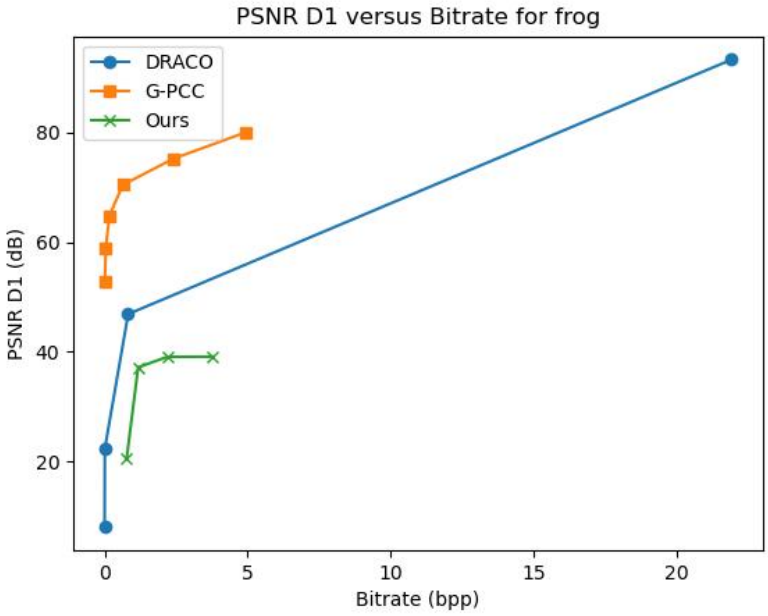


Figure 4.10: Performance measurement on frog.

Table 4.13: Performance Metrics of facade09

Num Bits	Compressed Model Size (Bytes)	PSNR D1 (dB)	Bitrate (bpp)
2	143241	24.18	1.29
4	220739	40.60	1.98
8	425511	44.42	3.82
16	739363	44.40	6.65

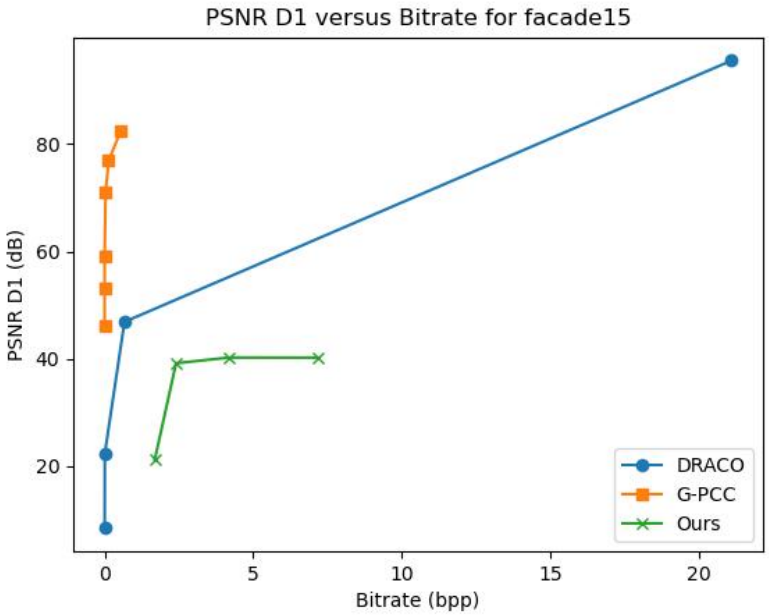


Figure 4.11: Performance measurement on facade15.

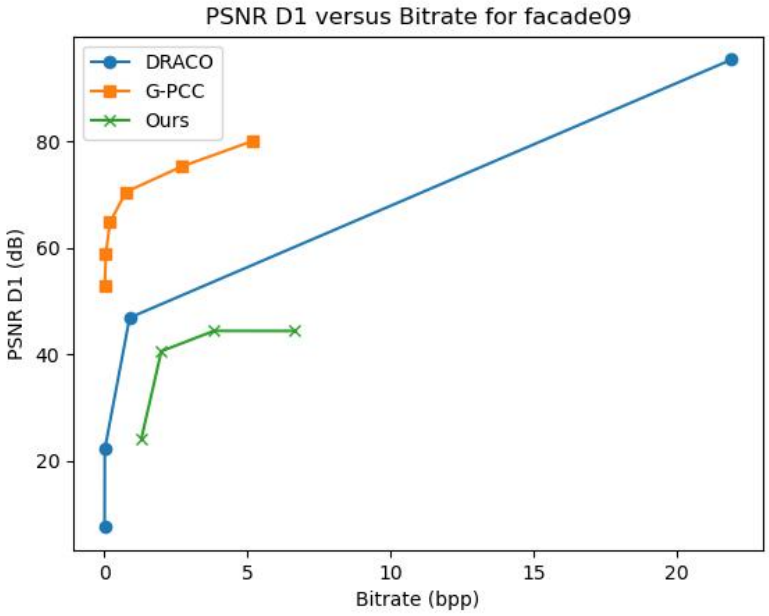


Figure 4.12: Performance measurement on facade09.

Table 4.14: Performance Metrics of boxer

Num Bits	Compressed Model Size (Bytes)	PSNR D1 (dB)	Bitrate (bpp)
2	160608	27.36	1.29
4	241871	33.59	1.94
8	438967	52.04	3.53
16	748941	52.10	6.02

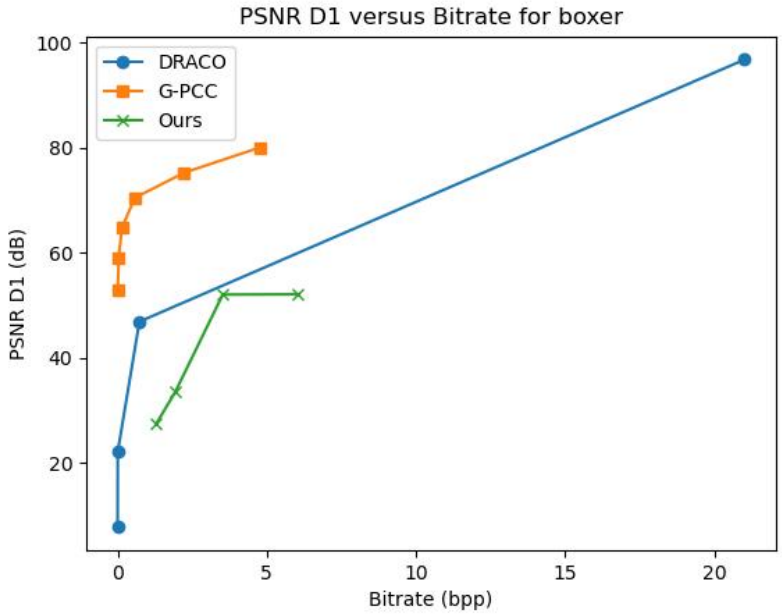


Figure 4.13: Performance measurement on boxer.

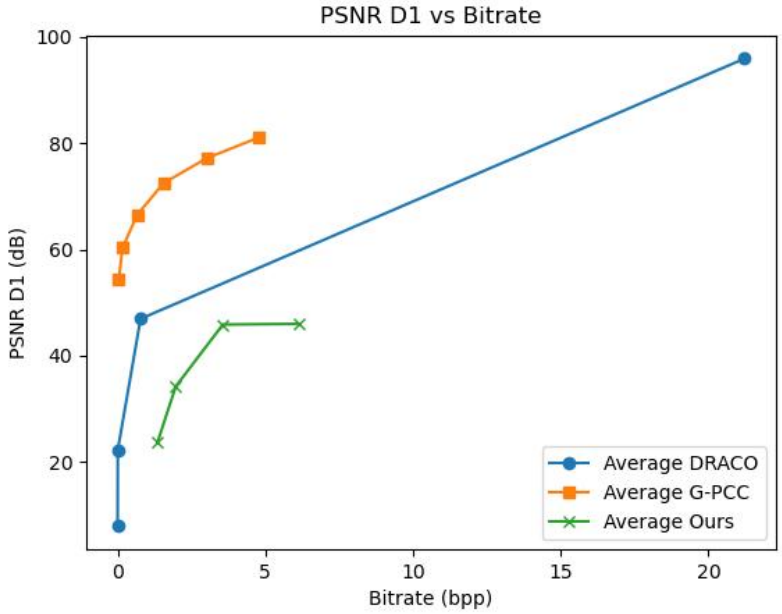
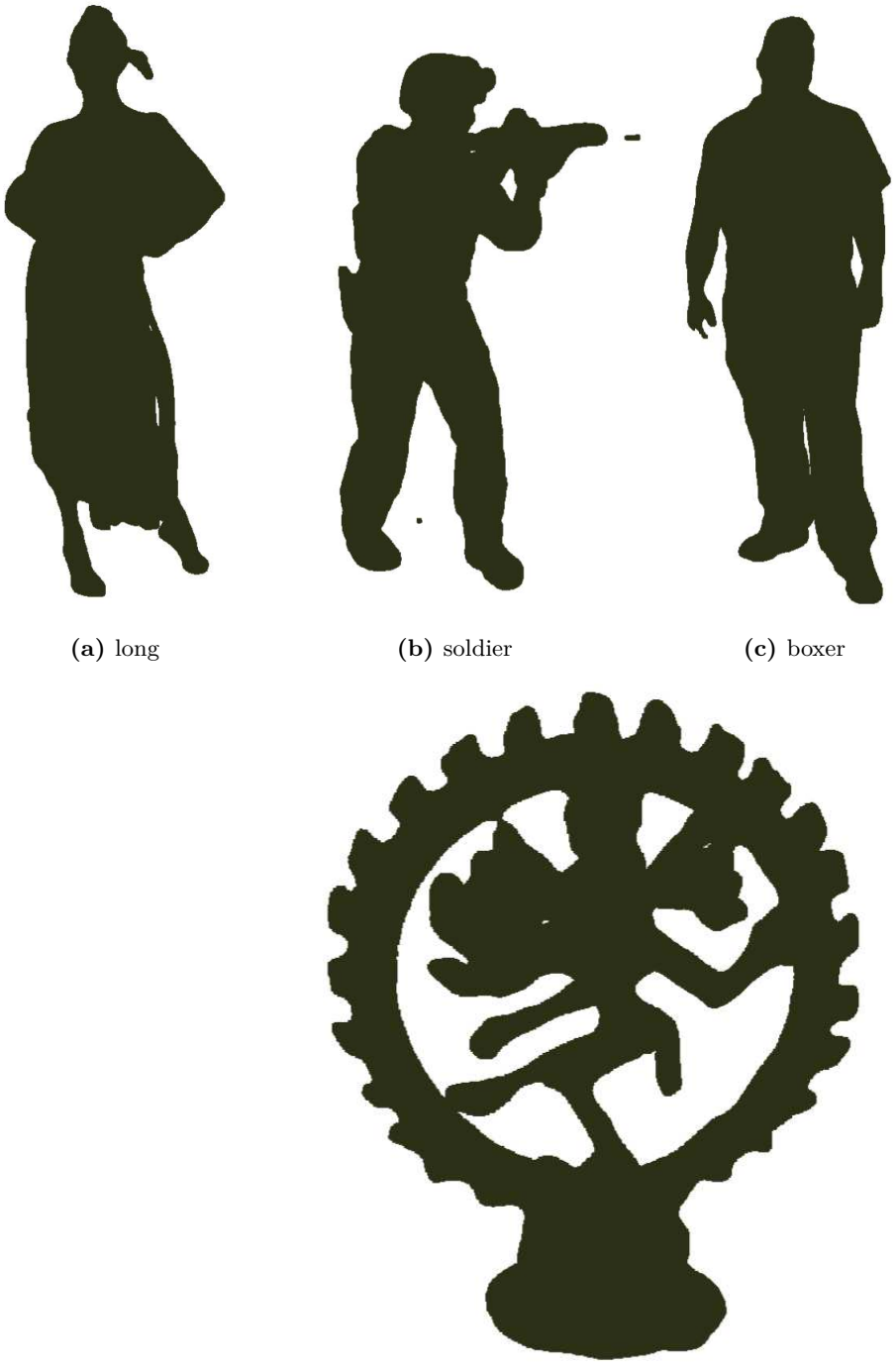


Figure 4.14: Average performance comparison.



(a) long **(b)** soldier **(c)** boxer

(d) shiva

Figure 4.15: Snapshots from some reconstructed PCs with 16 bits

5

Conclusions

This chapter presents the conclusions drawn from this thesis work.

5.1 Summary of Research Findings

The primary aim of this study was to explore the possibilities of compressing 3D point cloud data effectively, with a particular focus on arbitrary bitrates and dynamic resolution adjustments. While the PSNR metrics indicated that our compression method did not necessarily outperform existing methods in terms of preserving fine details, it still has greater flexibility in the choice of the decoded resolution and number of points which is due to the inherent characteristics of neural implicit representations.

A groundbreaking feature of our approach lies in its capacity for dynamic resolution adjustment. Leveraging the DiGS neural network model, we demonstrated that the point cloud's resolution could be increased or decreased as needed during the reconstruction phase. This offers significant advantages in real-world applications, such as augmented reality (AR). In an AR setting, the model's resolution can be dynamically adapted to the user's proximity to the object, allowing for a balance between detail and computational efficiency.

The ability to adjust the resolution of reconstructed point clouds dynamically has far-reaching implications for various fields. In Augmented Reality (AR) or Virtual Reality (VR), for instance, our methodology allows for an adaptive experience. As the user moves closer to or farther away from a 3D object, the system can modify the resolution, balancing visual detail and computational efficiency. This adaptability could lead to more immersive and responsive AR/VR applications.

Finally, the capability to function at arbitrary bitrates makes our approach highly versatile, potentially serving as a foundational technology for future compression algorithms that aim for both efficiency and adaptability.

While our method offers flexibility in resolution and bitrate, it does come with its own set of limitations. One of the primary drawbacks of the approach is the suboptimal PSNR values obtained using the uncompressed DiGS architecture which gives us a great disadvantage w.r.t. other methods since after compression distortion can only increase.

However, as NIR approaches improve this type of solution might become more and more viable given that the performance degradation introduced by compression is comparable with the one displayed by G-PCC and Draco. Additionally, the approach relies on neural networks, which require substantial computational resources for training. This could be a bottleneck for real-time applications where rapid encoding is necessary. However, by using the DiGS architecture we have already considerably reduced the training

time compared to simpler solutions.

Another challenge lies in the fine-tuning of parameters, including the choice of loss functions and quantization techniques, to strike an optimal balance between compression rate and reconstruction quality. The adaptability of the method, while a strength, also introduces complexity in finding the best set of parameters for any given application.

The research presented here opens several future improvements. One of the most immediate steps could be the exploration of other compression methods, such as Quantization Aware Training (QAT). Implementing QAT could potentially lead to better optimization and improved PSNR values, addressing one of the current limitations of our method.

Another exciting direction for future work could be the integration of our compression method with real-time applications, like augmented reality. This would involve overcoming the current limitations related to computational resources and training time.

Lastly, further research can also focus on improving the initial model quality, which is a key determinant in the fidelity of the reconstructed point cloud. Enhancing the initial model can potentially lead to better compression and reconstruction outcomes.

In summary, this thesis has explored the challenges and opportunities in point cloud compression, particularly focusing on neural implicit representations as a novel approach. While quality metrics like PSNR may not be comparable to traditional methods, its versatility and scalability offer significant advantages. The ability to reconstruct point clouds at arbitrary resolutions opens up new possibilities for applications ranging from entertainment to scientific research. As technology continues to evolve, the relevance of efficient and versatile point cloud compression methods will only increase, making the findings of this research both timely and impactful.

Bibliography

[1] E. Dupont, A. Goliński, M. Alizadeh, Y. W. Teh, and A. Doucet, "Coin: Compression with implicit neural representations," 2021.

[2] E. Dupont, H. Loya, M. Alizadeh, A. Goliński, Y. W. Teh, and A. Doucet, "Coin++: Neural compression across modalities," 2022.

[3] S. Milani, "Adae: Adversarial distributed source autoencoder for point cloud compression," in *2021 IEEE International Conference on Image Processing (ICIP)*, 2021, pp. 3078–3082.

[4] J. Wang, H. Zhu, H. Liu, and Z. Ma, "Lossy point cloud geometry compression via end-to-end learning," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 31, no. 12, pp. 4909–4923, dec 2021. [Online]. Available: https://doi.org/10.1109%2Ftcsvt.2021.3051377

[5] R. B. Rusu and S. Cousins, "3d is here: Point cloud library (pcl)," in *2011 IEEE International Conference on Robotics and Automation*. IEEE, 2011, pp. 1–4.

[6] H. E. Cline, W. E. Lorensen, S. Ludke, C. R. Crawford, and B. C. Teeter, "Two algorithms for the three-dimensional reconstruction of tomograms," *Medical Physics*, vol. 15, no. 3, pp. 320–327, 1988.

[7] H. Hoppe, "Progressive meshes," in *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, 1996, pp. 99–108.

[8] A. Asvadi, L. Garrote, C. Premebida, P. Peixoto, U. Nunes, and M. M. Peña, "Multimodal vehicle detection: fusing 3d-lidar and color camera data," *Pattern Recognition Letters*, vol. 115, pp. 20–29, 2017.

[9] M. Berger, A. Tagliasacchi, L. M. Seversky, P. Alliez, J. A. Levine, A. Sharf, and C. T. Silva, "A survey of surface reconstruction from point clouds," *Computer Graphics Forum*, vol. 36, no. 1, pp. 301–329, 2013.

[10] X. Tian, H. Choi, Y. Zhang, and K. Mueller, "Geometric compression for massively parallel visualization and computing," in *High Performance Graphics*, 2014, pp. 93–102.

[11] R. Schnabel, R. Wahl, and R. Klein, "Efficient ransac for point-cloud shape detection," *Computer Graphics Forum*, vol. 26, no. 2, pp. 214–226, 2007.

[12] T. Zhou, J. Fu, L. Xue, and L. Liu, "Real-time 3d shape segmentation through spectral neural networks," in *Graphics Interface*, 2018, pp. 41–48.

[13] T. Komura and M. Iwamoto, "Large scale processing of 3d scan data to achieve semantic segmentation," *Virtual Reality*, vol. 23, no. 1, pp. 13–24, 2019.

[14] P. Alliez and C. Gotsman, "Recent advances in compression of 3d meshes," *Advances in Multiresolution for Geometric Modelling*, pp. 3–26, 2005.

[15] Y. Ben-Shabat, C. H. Koneputugodage, and S. Gould, "Digs : Divergence guided shape implicit neural representation for unoriented point clouds," 2023.

[16] J. J. Park, P. Florence, J. Straub, R. Newcombe, and S. Lovegrove, "Deepsdf: Learning continuous signed distance functions for shape representation," *CVPR*, 2019.

[17] L. Mescheder, M. Oechsle, M. Niemeyer, S. Nowozin, and A. Geiger, "Occupancy networks: Learning 3d reconstruction in function space," *CVPR*, 2019.

[18] Z. Wang and A. Jacobson, "Hierarchical neural implicit surfaces," *SIGGRAPH*, 2020.

[19] V. Sitzmann, J. Martel, A. Bergman, D. Lindell, and G. Wetzstein, "Implicit neural representations with periodic activation functions," in *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, Eds., vol. 33. Curran Associates, Inc., 2020, pp. 7462–7473. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2020/file/53c04118df112c13a8c34b38343b9c10-Paper.pdf

[20] B. Isik, P. A. Chou, S. J. Hwang, N. Johnston, and G. Toderici, "Lvac: Learned volumetric attribute compression for point clouds using coordinate based networks," 2021.

[21] F. Pistilli, D. Valsesia, G. Fracastoro, and E. Magli, "Signal compression via neural implicit representations," in *ICASSP 2022 - 2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2022, pp. 3733–3737.

[22] S. Milani, "A syndrome-based autoencoder for point cloud geometry compression," in *2020 IEEE International Conference on Image Processing (ICIP)*, 2020, pp. 2686–2690.

[23] A. F. R. Guarda, N. M. M. Rodrigues, and F. Pereira, "Deep learning-based point cloud geometry coding with resolution scalability," in *2020 IEEE 22nd International Workshop on Multimedia Signal Processing (MMSP)*, 2020, pp. 1–6.

[24] ——, "Point cloud geometry scalable coding with a single end-to-end deep learning model," in *2020 IEEE International Conference on Image Processing (ICIP)*, 2020, pp. 3354–3358.

[25] ——, "Adaptive deep learning-based point cloud geometry coding," *IEEE Journal of Selected Topics in Signal Processing*, vol. 15, no. 2, pp. 415–430, 2021.

[26] J. Wang, H. Zhu, H. Liu, and Z. Ma, "Lossy point cloud geometry compression via end-to-end learning," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 31, no. 12, pp. 4909–4923, 2021.

[27] A. F. R. Guarda, N. M. M. Rodrigues, and F. Pereira, "Neighborhood adaptive loss function for deep learning-based point cloud coding with implicit and explicit quantization," *IEEE MultiMedia*, vol. 28, no. 3, pp. 107–116, 2021.

[28] S. Schwarz, M. Preda, V. Baroncini, M. Budagavi, P. Cesar, P. A. Chou, R. A. Cohen, M. Krivokuća, S. Lasserre, Z. Li, J. Llach, K. Mammou, R. Mekuria, O. Nakagami, E. Siahaan, A. Tabatabai, A. M. Tourapis, and V. Zakharchenko, "Emerging mpeg standards for point cloud compression," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 9, no. 1, pp. 133–148, 2019.

[29] M. Quach, G. Valenzise, and F. Dufaux, "Improved deep point cloud geometry compression," 2020.

[30] S. Milani, "Adae: Adversarial distributed source autoencoder for point cloud compression," in *2021 IEEE International Conference on Image Processing (ICIP)*, 2021, pp. 3078–3082.

[31] M. Quach, G. Valenzise, and F. Dufaux, "Learning convolutional transforms for lossy point cloud geometry compression," in *2019 IEEE International Conference on Image Processing (ICIP)*. IEEE, sep 2019. [Online]. Available: https://doi.org/10.1109%2Ficip.2019.8803413

[32] K. Maja, A. C. Philip, and S. Patrick, "8i Voxelized Surface Light Field (8iVSLF) Dataset," *ISO/IEC JTC1/SC29 WG11 (MPEG) input document m42914, Ljubljana*, July 2018.

[33] M. Atzmon and Y. Lipman, "Sal: Sign agnostic learning of shapes from raw data," 2020.

Acknowledgements

First and foremost, I would like to express my deepest gratitude to Professor Simone Milani, who supervised my work and accepted me as his student. His guidance and support have been invaluable to my research and academic growth.

Special thanks go to Daniele Mari, who co-supervised my work. Daniele's continuous support, willingness to answer all my questions, and prompt assistance have been crucial to my academic journey.

I owe a debt of gratitude to my family, who have always been my greatest supporters. Living abroad without them has been a challenge, but their love and encouragement have fueled my determination to do my best.

Lastly, I want to thank my friends who have made my experience abroad not only bearable but enjoyable. Their encouragement and support, especially during the tough times, have been indispensable.

Thank you all for playing a significant role in my academic journey.