Neural-network-based Volume Compression

Anže Kristan, Matija Marolt, Ciril Bohak, Žiga Lesar

University of Ljubljana, Faculty of Computer and Information Science
E-mail: ak4352@student.uni-lj.si, {matija.marolt, ciril.bohak, ziga.lesar}@fri.uni-lj.si

Abstract

Volumes are a common representation of volumetric data, but they often result in large data sizes due to their three-dimensional structure, which can include substantial empty space or repetitive patterns. This study explores the use of neural-network-based techniques to compress volumetric data efficiently. We implemented and tested several neural network models designed to encode small data blocks and reconstruct the original volumes. These models provide lossy compression, with reconstructions displaying various artifacts inherent to block-based approaches. Our approach was evaluated against traditional compression methods, demonstrating that our neural network models achieve significant compression ratios, although they require extended training periods. The presented neural-network-based compression is suitable for specific application domains where exact reconstruction precision is not critical.

1 Introduction

Volumes are the most common representation of volumetric data, capturing values such as color, intensity, or material density at points on a three-dimensional grid. Volumes typically arise from various 3D acquisition techniques, including computed tomography [1, 4], magnetic resonance imaging [17], and 3D ultrasound [6]. Their applications span medical imaging, material sciences, physics simulations, and entertainment industries.

Due to their three-dimensional nature, volumes can require significant storage space, especially with high-resolution grids or when storing each frame of a simulation or animation as a separate volume. General compression methods, such as run-length encoding (lossless) [12, 20] and wavelet transform (lossy) [18, 9], attempt to address these challenges but often struggle with achieving high compression ratios without substantial data loss, as they cannot take advantage of the spatial correlations between voxels.

In this work, we investigate neural networks for volume compression, focusing on achieving smaller file sizes even at the expense of less accurate or slower reconstructions. Neural networks' ability to learn complex patterns presents a promising solution for efficient compression. We tested various models with different parameters to encode small data blocks and reconstruct the original volumes, aiming to surpass traditional methods in compression ratio.

Our experiments, detailed in Section 4, involved three different volumes and multiple neural network configurations. We discuss the results, comparing the performance of neural-network-based compression with conventional technique (ZIP), and offer conclusions based on our findings. This approach provides a potential tool for efficiently managing large volumetric datasets, particularly when exact reconstruction is not critical.

2 Related work

One of the currently most well-known and popular formats for sparse volumetric data is the so-called VDB format [10] available as OpenVDB [15] implementation. It is a hierarchical dynamic grid structure, similar to a B+ tree, and is most useful when dealing with large, sparse, and animated volumes, such as level sets and cloud modeling. Jain et al. [3] described a deep learning technique for compressed volume rendering, maintaining high-quality rendering capabilities while achieving data compression, which demonstrates the early integration of neural networks in volumetric data compression.

Tang et al. [19] introduced an approach for volume compression via truncated signed distance fields using a block-based neural network architecture. Their method demonstrated higher quality outputs for the same bitrate compared to previous methods. Nagoor et al. [11] proposed a deep neural network with local sampling for lossless compression of volumetric medical images, significantly reducing storage requirements without compromising image quality, which is crucial for medical applications where data integrity is paramount.

Oblak et al. [13] explored lossy data compression of volumetric simulation data for web-based vascular flow simulation visualization, highlighting the feasibility of fast, real-time remote visualization. Recently, Lesar et al. [21] presented a web-friendly volume storage and compression solution, balancing compression efficiency with rapid data access on web platforms.

Building on these concepts, NeuralVDB [5] employs multiple hierarchical neural networks to compress sparse data, offering substantial improvements over OpenVDB with significant reduction factors.

3 Methodology

To select and train the best models, we qualitatively evaluated the results by visually inspecting the isosurface representation using the VPT [8] framework, compressed size, and training times. We further improved the most promising models by tweaking their parameters.

The selected models are autoencoders, which encode each block of the volume (a $k \times k \times k$ size subsection of the volume) as a separate code. The block size was defined as an input parameter (k), which influences the number of blocks the volume is split into and the size of the individual layers of the neural network models. The tested architectures were autoencoders with linear layers and architectures with handcrafted encodings with linear decoders.

We tested the following architectures:

- **linear1**: the encoder receives a flattened block as the input (k^3) and is composed of 4 linear layers, each following layer is half the size of the previous, except the last layer. The last layer's output size (the encoding size) is determined by the parameter s. The decoder architecture is the same but in reverse.
- **linear2**: the encoder receives a flattened block as the input (k^3) and is composed of 4 linear layers. The output size of each layer is $\frac{1}{8}$-th the input size for the first two layers, then $\frac{1}{12}$-th, and finally, the size defined by the s parameter. The decoder architecture is the same but in reverse.
- **coordvarsvd**: the encoded features are handcrafted for each block and are as follows: 3 floats representing the coordinates of the current block normalized to [0,1], 3 floats representing the mean variances in (x,y,z) axis of the current block, and finally 3 floats representing the largest 3 singular values of this block. The decoder has the same architecture as in the linear1 architecture.
- **svd1**: similar to coordvarsvd, but only uses the 3 largest svd values as the encoded representation of each block. It uses the linear1 decoder architecture.
- **svd2**: similar to svd1, but uses the decoder architecture of linear2 instead.

All models were implemented in PyTorch [16] and used Mean Square Error as a loss function (MSELoss) and Stochastic Gradient Descent (SGD) for optimization. All encoded values were saved to files as 16-bit floats.

Each model was trained for 50 epochs. The only exception is the linear1_200ep model, which uses the linear1 architecture, but was trained for 200 epochs due to its fast training time. Even at the end of these 200 epochs, the loss was still decreasing, so there is still room for improvement for each model by simply training them further.

After we finished investigating the different architectures, we also investigated the effect of the encoding size (s) on the reconstruction and file sizes for model following using the linear1 and linear2 architectures. For this purpose, the created models were trained for 200 epochs.

Table 1: Details on analyzed volumes.

name	shape	raw (MB)	ZIP (MB)
tooth	$103 \times 94 \times 161$	1.48	0.88
bonsai	$256 \times 256 \times 256$	16.00	3.14
body	$512 \times 512 \times 226$	56.50	8.25

4 Results

We tested several architectures on 3 separate volumes [14] detailed in Table 1. All volumes are raw binary files containing `uint8` values and have to be reshaped to the correct dimensions. To remove some noise in the volume data, we filtered out (set to 0) the values below the empirically determined threshold κ for each volume (tooth: $\kappa = 105$, bonsai: $\kappa = 50$, body: $\kappa = 0$). We then resampled the volume values from $[0, 255] \rightarrow [0, 1]$. As the final preprocessing step before training each neural network model, we padded the input volume to be divisible by the model's specified block size parameter $k \rightarrow [k, k, k]$. All computing was done via Google Colab [2] on the free tier CPU runtime, using Python, PyTorch and NumPy. The code is available on GitHub [7].

4.1 Tooth

For the tooth volume, several different model architectures and parameters were tested. Table 2 shows the most promising models, their parameters, encoded file sizes, decoder file sizes, and MSE of the difference between the reconstructed file and input file. Table 2 also shows the times it took to train a model for one epoch (one iteration over the whole volume), the time to encode and save the volume to file, and the time to decode and save the volume to file. Figure 1 shows images of the original volume and its filtered version used for training and 4 reconstructions of the following models: linear1 ($k = 4$) (chosen for lowest MSE), linear1 ($k = 8$) (least time consuming), linear1_200ep ($k = 8$), svd2 ($k = 4$) (small file sizes). Visualizations were done using VPT with the isosurface extraction renderer, with the same isovalue for each visualization.

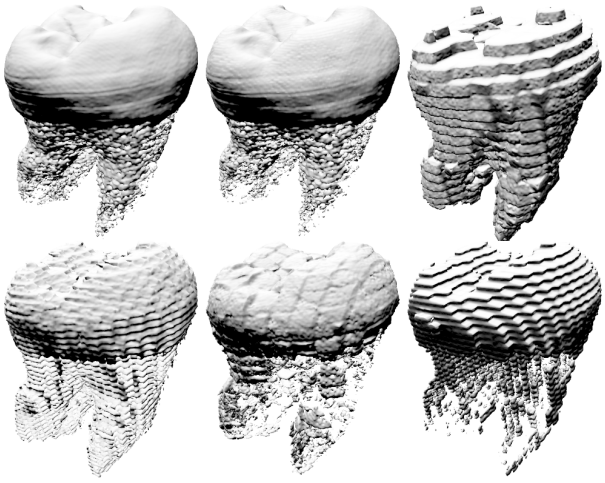

Figure 1: From left to right, top to bottom: original tooth volume, filtered tooth volume, and 4 reconstructions: linear1 ($k = 8$), linear1 ($k = 4$), linear1_200ep ($k = 8$), and svd2.

Table 2: Comparison of models for tooth volume, based on kernel size, encoding size (number of floats), the reconstructed volume MSE, file size of the decoder, file size of the encoded representation, training, encoding, and decoding times.

Name	k	s	MSE	decoder (kB)	encoded (kB)	epoch time (s)	enc. time (s)	dec. time (s)
linear1	8	8	393.17	682	51	**9.5**	2.5	2.7
linear1_200ep	8	8	141.60	682	51	-	-	-
linear1	4	8	**136.17**	15	400	49.2	9.3	6.6
linear2	8	8	907.28	163	51	13.1	**1.8**	**1.0**
linear2	4	8	276.79	**7**	400	50.8	9.4	6.9
coordvarsvd	4	9	297.17	15	450	46.2	10.2	8.5
svd1	4	3	307.30	15	150	45.3	5.3	9.4
svd2	4	3	301.07	**7**	**150**	41.7	5.1	8.7

Figure 2 shows the reconstructed volumes after filtering out low values and then passing them through a Gaussian blur with the parameters $\sigma = 1$ and $r = \frac{k}{2}$. The represented volumes are the original volume ($k = 8$), the linear1 ($k = 4$), linear1_200ep, and svd2.

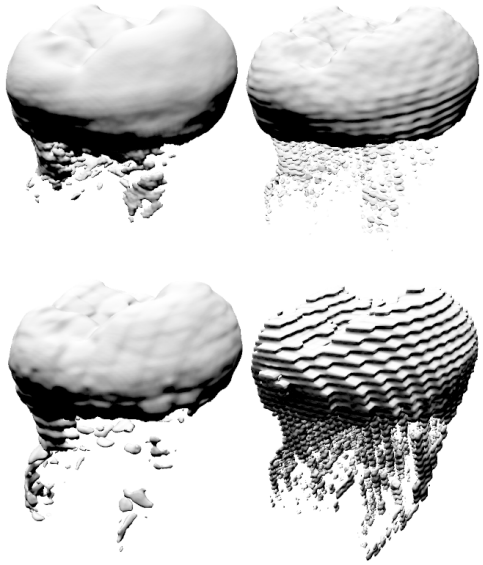

Figure 2: Volumes filtered with Gaussian blur. From left to right, top to bottom: original tooth volume, and 3 reconstructions: linear1 ($k = 4$), linear1_200ep ($k = 8$), and svd2.

Finally, as seen in Figure 3, we examined the effect of the different encoding layer sizes s. We trained models of two different architectures with different encoding layer sizes for 200 epochs each. The selected architectures were linear1 with parameters ($k = 8$) and $s \in \{1, 3, 5, 7\}$, and linear2 with parameters ($k = 8$) and $s \in \{5, 7\}$. In Table 3 are the MSE losses for the different models and file sizes (decoder + encoded representation).

Table 3: Comparison of the effect of enc_size parameter when training for 200 epochs.

Model	enc_size	MSE	File Size (kB)
linear1	1	1757.65	686
	3	477.66	700
	5	**137.40**	713
	7	140.50	727
linear2	5	341.75	**194**
	7	171.13	207

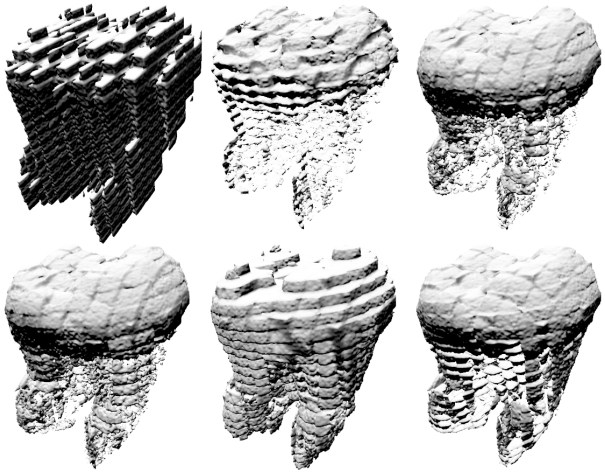

Figure 3: Tooth volume with different encoding layer sizes. From left to right, top to bottom: linear1 ($s = 1$), linear1 ($s = 3$), linear1 ($s = 5$), linear1 ($s = 7$), linear2 ($s = 5$), linear2 ($s = 7$).

4.2 Bonsai

Due to its size, we tried only a few models for the bonsai volume. The only model we considered even remotely successful with compression size and reconstruction visuals was svd2 ($k = 4$). The MSE for the reconstruction with this model was 79.92, and the output file sizes were 7kB decoder and 1.5MB encoded file. The training time for 1 epoch was 6m 56s, the encoding time was 54.9s, and the decoding time was 1m 19s.

4.3 Body

For the body volume, the only even remotely satisfactory model was the coordvarsvd ($k = 8$). The MSE for the reconstruction with this model was 1645.58, and the output file sizes were 682kB decoders and 2MB encoded files. The training time for 1 epoch was 5m 3s, the encoding time was 55.2s, and the decoding time was 59s.

5 Discussion

As we have taken a block-based approach to encoding the volumes, most of the problems with the reconstructions are that they lack smooth transitions between reconstructed blocks. Reconstructions of some models on the tooth volume achieved similar visual results to the original volume, especially after applying Gaussian blur to the reconstruction. However, as can be seen with the model

linear1_200ep, additional training of the models could increase the faithfulness of the reconstruction. Even after 200 epochs, the loss still diminishes, and the artifacts present in the reconstruction are less pronounced. We did not check whether this is true in general or may only be viable for some combinations of input volumes, models, and parameters.

There seems to be a large tradeoff between the model's kernel size and the size of the model. Larger kernel sizes lead to faster training, encoding, and decoding times, as well as smaller encoding file sizes, but they result in larger decoder file sizes and produce visible artifacts in the reconstruction when the number of training epochs is low. We also tried architectures utilizing 3D convolutional layers, which would allow for smaller decoder sizes. However, visual inspection of the reconstructions from these models was considered lacking, as there were large gaps between individual blocks.

An observation made during the training of the models was that after the first epoch, the general shape of the volume gets captured. However, it lacks details and smooth transitions between blocks. However, even after training for many epochs, the models still produce blocky reconstructions. Although this can be somewhat remedied with Gaussian blur to process the reconstructions, that can lead to loss of detail in itself. We might also consider using overlapping blocks.

From the results, we can conclude that selecting an optimal model depends on the specific input volume. Considering the long training and decoding time, models with a smaller kernel size may not be usable in real-world applications for bigger volumes. Combining this with the lossy nature of such neural-network-based compression, a simple zip file might satisfy many use cases since it does not need to be adapted to each individual volume. Moreover, there is a question of the tradeoff of the electricity and time spent on training such models for each volume versus simply using more storage/transfer bandwidth to hold the zip files of an arbitrary volume.

However, this field may still have potential, as shown in the related works. One option is to try a different volumetric representation rather than a simple volume. Another idea is to try to focus on a single model (that can be trained quickly and decodes the volume quickly) and an algorithm to estimate the model parameters for the target volume automatically.

6 Conclusion

Compression of volumetric data is challenging, and many ways exist to tackle it. The results of the use of different deep models with diverse sets of parameters explored in this work are inconclusive and do not provide us with a definite answer on their usability. As shown, it all depends on the speed of training/encoding/decoding, the compression factor, and the accuracy of the reconstructed volume desired in a particular use case.

Acknowlegment

This research was conducted as part of the basic research project *Cell visualization of unified microscopic data and procedurally generated sub-cellular structures* [project number J2-50221], funded by the Slovenian Research and Innovation Agency (Javna agencija za znanstvenoraziskovalno in inovacijsko dejavnost RS) from the state budget.

References

[1] C. R. Crawford and K. F. King. Computed tomography scanning with simultaneous patient translation. *Medical Physics*, (17):967 – 982, 1990.

[2] Google colab. `https://colab.research.google.com`.

[3] Somay Jain, Wesley Griffin, Afzal Godil, Jeffrey W Bullard, Judith Terrill, and Amitabh Varshney. Compressed volume rendering using deep learning. In *Proceedings of the Large Scale Data Analysis and Visualization Symposium*, pages 1187–1194, 2017.

[4] W. A. Kalender, W. Seissler, E. Klotz, and P. Vock. Spiral volumetric CT with single-breath-hold technique, continuous transport and continuous scanner rotation. *Radiology*, (176):181 – 183, 1990.

[5] Doyub Kim, Minjae Lee, and Ken Museth. Neuralvdb: High-resolution sparse volume representation using hierarchical neural networks, 2024.

[6] D. Krakow, J. Williams, M. Poehl, D. L. Rimoin, and L. D. Platt. Use of three-dimensional ultrasound imaging in the diagnosis of prenatal-onset skeletal dysplasias. *Ultrasound in Obstetrics and Gynecology*, 21(5):467–472, 2003.

[7] Anže Kristan, Matija Marolt, Ciril Bohak, and Žiga Lesar. Neural-network-based volume compression. `https://github.com/UL-FRI-LGM/NN-based-volume-compression`.

[8] Žiga Lesar, Ciril Bohak, and Matija Marolt. Real-time interactive platform-agnostic volumetric path tracing in webgl 2.0. In *Proceedings of Web3D*, 2018.

[9] Stéphane Mallat. *A wavelet tour of signal processing: The Sparse Way*. Elsevier, 2009. 3rd Ed.

[10] Ken Museth. Vdb: High-resolution sparse volumes with dynamic topology. *ACM TOG*, 32(3):1–22, 2013.

[11] O. H. Nagoor, J. Whittle, J. Deng, B. Mora, and M. W. Jones. Lossless compression for volumetric medical images using deep neural network with local sampling. In *2020 IEEE ICIP*, pages 2815–2819, 2020.

[12] Mark Nelson and Jean-Loup Gailly. The data compression book 2nd edition. *M & T Books, New York, NY*, 1995.

[13] Rok Oblak, Ciril Bohak, and Matija Marolt. Web-based vascular flow simulation visualization with lossy data compression for fast transmission. In *Augmented reality, virtual reality, and computer graphics : proceedings*, pages 3–17, 2018.

[14] Open scivis datasets. `https://klacansky.com/open-scivis-datasets/`.

[15] Openvdb. `https://www.openvdb.org`.

[16] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017.

[17] P. A. Rinck. *Magnetic Resonance in Medicine. The Basic Textbook of the European Magnetic Resonance Forum. 9th edition*, volume 9.1. TRTF, 2016.

[18] Gilbert Strang and Truong Nguyen. *Wavelets and filter banks*. SIAM, 1996.

[19] Danhang Tang, Saurabh Singh, Philip A Chou, Christian Hane, Mingsong Dou, Sean Fanello, Jonathan Taylor, Philip Davidson, Onur G Guleryuz, Yinda Zhang, et al. Deep implicit volume compression. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 1293–1303, 2020.

[20] Ian H Witten, Alistair Moffat, and Timothy C Bell. *Managing gigabytes: compressing and indexing documents and images*. Morgan Kaufmann, 1999.

[21] Žiga Lesar, Ciril Bohak, and Matija Marolt. Blocky volume package : a web-friendly volume storage and compression solution. In *Proceedings of WSCG 2023*, pages 213–221, 2023.