Volume 0 (1981), Number 0 pp. 1–4 COMPUTER GRAPHICS *forum*

Neural Volume Compression

Matej Bevec[1]

[1]University of Ljubljana, Faculty of Computer and Information Science

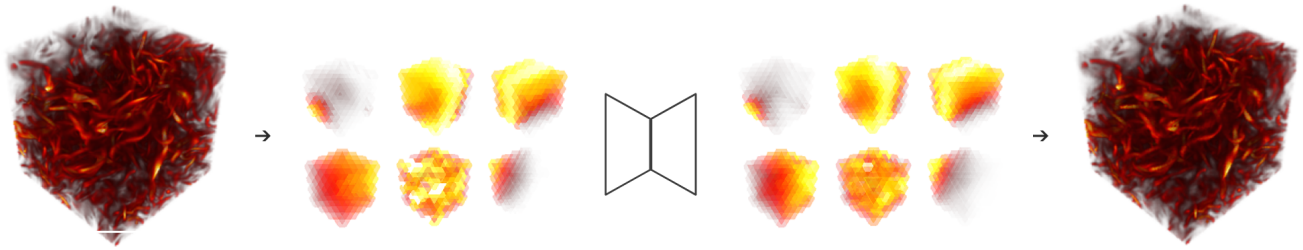

Figure 1: The VAE encoder-decoder system.

Abstract

Volumetric data, commonly used in many scientific fields, requires high spatial resolution which poses challenges in visualization, storage and bandwidth. Spatial compression can help address these issues and can be achieved in various ways.
We explore learning the compressed representation from data with a neural network based approach. A given volume is segmented into small uniform blocks, which are then reconstructed with an autoencoder architecture through a low-dimensional bottleneck, which can be used as the compressed representation. The artifacts emerging from block-wise operation are addressed with an overlapping margin approach. Despite a simple architecture and limited training resources, the implemented method produces faithful reconstructions with minimal errors at significant compression rates.

1. Introduction

Volumetric data is utilized in various fields and is often the product of numerical scientific simulations or biomedical scans, which require high spatial resolution for accurate operation. This large scale is often challenging for visualization, disk storage or bandwidth constrains. As such, volumetric data benefits greatly from spatial compression.

Much like two-dimensional images, volumetric data exhibits a level of redundancy, which may be exploited for compression. In fact, due to the power rule, we may expect an even larger gain from compression in three dimensions. One possible approach to achieving this is to segment a given volume into uniform blocks and represent them in Fourier/frequency space [YL95] or in a Wavelet-based space [WMB*11]. Here, the least significant frequencies can be discarded to reduce file size while maintaining a decent reconstruction.

We instead attempt to learn a compressed representation with a neural network. Assuming sufficiently small uniform blocks, we conjecture that their latent distribution across all encountered volumes is both finite and lower dimensional than their pixel count. This compressed representation is learned using an autoencoder architecture trained to reconstruct said blocks through a low-dimensional bottleneck.

The following sections describe the implemented method and present the results of our evaluation. An easy to use library providing pre-trained models as well as custom training utilities and scripts to reproduce presented results is available at https://github.com/MatejBevec/neural-volume-compression.

2. Method

2.1. Blockwise reconstruction

In our approach, a given three-dimensional volume is first segmented into small (eg. $8 \times 8 \times 8$) uniform blocks, which are then independently encoded by an autoencoder model. The encoded block

set can be stored as a compressed file. During the decoding step, the blocks are independently decoded and reconstructed (see Figure 1).

The autoencoder model is trained, using mean-square error (MSE), to reconstruct said blocks and has no information as to how they form a volume at large. Training reconstruction on small blocks vastly reduces the parameter space of the autoencoder model which can be trained to essentially overfit the training set. We assume there is a finite, not intractably-large number of different blocks that ever appear in volumes, or at least their distribution can be approximated by a finite set. Therefore, since an unseen volume does not introduce any new block types, memorizing (i.e. overfitting on) the block space is a valid strategy.

2.2. Autoencoder architecture

Since overfitting is an acceptable and even desired outcome in our case, we conjecture that using a fully-connected encoder-decoder architecture should be sufficient. Specifically we employ a dense variational autoencoder (VAE) architecture trained with MSE to reconstruct volumetric blocks and take the latent mean vector as volume's compressed (encoded) representation.

The relatively small input block size allows a shallow architecture with a single hidden layer in both the encoder and the decoder. This is convenient at inference time (i.e. when using the resulting codec) because forward passes are relatively fast. Another advantage is a low-dimensional bottleneck since the size of the latent vector is inversely proportional to the achieved compression factor.

2.3. Oversized blocks

While we can expect faithful reconstruction per-block, treating blocks independently introduces specific artifacts after the volume at large is reassembled. Most notably, we can see unwanted discontinuities between bordering blocks, where there are none in the original input. This is because the model is "unaware" of the expected continuity beyond a block's edge.

We evaluate one approach to alleviate this issue — introducing a margin to the segmented blocks. If a volume is split into $b \times b \times b$ blocks with margin m, each block is extended by an m wide border in each direction with data from the neighboring blocks. The autoencoder model then encodes the resulting $b + 2m$-wide blocks. During reconstruction the shared margins are either interpolated between neighboring decoded blocks, or only the margin-less section is cropped for a given block (see Figure 2)

3. Experiments

3.1. Dataset

The training dataset consists of 30 three-dimensional volumes sourced from [V23]. The examples vary in size and subject, spanning domains such medicinal CT scans, fluid dynamics data and scans of man-made physical objects.

The volumes are first converted to gray-scale (single channel) and normalized to a [0, 1] range. Then they are segmented into uniform blocks and padded with a margin as described. In the scope of this report, we consider $8 \times 8 \times 8$ blocks, possibly with a two-wide margin. In the edge cases, the volume may be zero-padded to ensure all blocks are of uniform size. Finally, the collection of all blocks from all volumes is shuffled prior to batching. All together, the dataset consist of approximately *1.7 million blocks*.

During training, we compute loss on the training set. However, we prepare an additional collection of 9 unseen volumes for the final evaluation.

3.2. Model variants and training

In this report, we focus on three fully-connected VAE variants, all of which are relatively shallow with a single hidden layer in both the encoder and decoder (see Table 1). The first model uses no margin while the other two use a two-wide margin. The first two models use 32-dimensional latent vectors, while the last uses a larger 64-dimensional vector. This corresponds to 4x and 2x compression respectively.

variant	block size	margin	input layer	hidden layer	latent layer
zero margin	8	0	8^3	128	32
two-wide margin	8	2	12^3	128	32
more parameters	8	2	12^3	256	64

Table 1: Model parameters for the three considered variants.

All the above variants are trained with the Adam optimizer, a batch size of 64 and a learning rate of $10^{-}4$. Due to time and computational resource constraints, we only train for 8 epochs. Since loss keeps decreasing until the last epoch we presume improvements could be expected from longer training times.

4. Results

4.1. Reconstruction loss

Table 3 presents the MSE reconstruction loss evaluated on the test set. Surprisingly, these results suggest that introducing a two-wide margin decreases the reconstruction accuracy and doesn't provide an advantage, as long as we are only interested in the average per-pixel difference.

4.2. Qualitative analysis

However, the per-pixel reconstruction loss does does not reveal the entire picture. Figures 3 and 4 showcase a number of examples, i.e. reconstructed volumes, at different zoom levels, while Figure 2 depicts the effect of using a block margin.

It is evident that all variants produce a faithful reconstruction with minimal noticeable artifacts or changes. In some cases, there is some loss of fine detail (softness) at higher zoom levels. A more noticeable artifact can be seen in the *no margin* model examples (see Figure 2, image *b*), where the borders between blocks are visible in the final reconstruction. The images *c* and *d* in the same figure show how introducing a two-pixel-wide overlapping margin can alleviate this issue. In image *c*, the model is trained on 12-wide

size	zero margin enc	zero margin dec	two-wide margin enc	two-wide margin dec	more parameters enc	more parameters dec
$192 \times 180 \times 168$	0.09	0.19	0.21	0.31	0.22	0.33
$512 \times 512 \times 32$	0.13	0.26	0.28	0.40	0.31	0.47
$256 \times 256 \times 256$	0.27	0.48	0.59	0.82	0.67	0.95
$280 \times 512 \times 174$	0.42	0.69	0.97	1.48	1.06	1.36
$512 \times 512 \times 512$	2.16	3.64	5.97	7.92	7.54	9.38
$832 \times 832 \times 494$	5.90	9.77	38.8	20.7	26.1	25.4

Table 2: Encoding (*enc*) and decoding time (*dec*), in seconds, for different model variations on volumes of varying size. The times include the block decomposition and reconstruction steps.

model	compression factor	reconstruction loss
zero margin	4×	0.00205
two-wide margin	4×	0.00239
more parameters	2×	0.00213

Table 3: Reconstruction loss (MSE) for different model variations on the training set and a test set of unseen volumes. While loss is computed block-wise in training, the above results refer to reconstruction loss for entire volumes, thus taking into account e.g. margin interpolation.

blocks, but only the 8-wide margin-less regions are used in reconstruction. Already, the borders are hard to notice, although visible. In image *d*, the margin regions are interpolated between neighboring blocks during reconstruction. In this case, block borders are practically invisible.

4.3. Speed

Finally, we consider the encoder-decoder speed, which is another significant factor when it comes to its usability. We measure the encoding (compressing) and decoding (decompressing) speed of the evaluated models for increasingly large volumetric data (Table 2). The times include the block decomposition and reconstruction steps and are measured on a mid-range laptop.

All models operate in a practical time frame (0.09s to 38.8s), but there is an expected trade-off between the model size (and so its accuracy) and its speed. All models perform decoding and encoding steps in under a second for models up to $256 \times 256 \times 256$ in size.

5. Conclusion

In this seminar, we tackled the 3D volumetric data compression task by training a neural encoder-decoder network to reconstruct the data. An input volume is segmented into small uniform blocks. The neural network is then trained to compress these blocks and to decode their compressed representations. Despite a shallow architecture, limited training time and a small training dataset, this approach proves to be very successful and achieves a significant compression factor while producing faithful reconstruction with minimal error. Additionally, overlapping interpolated margins help alleviate the most significant artifacts that appear between block borders.

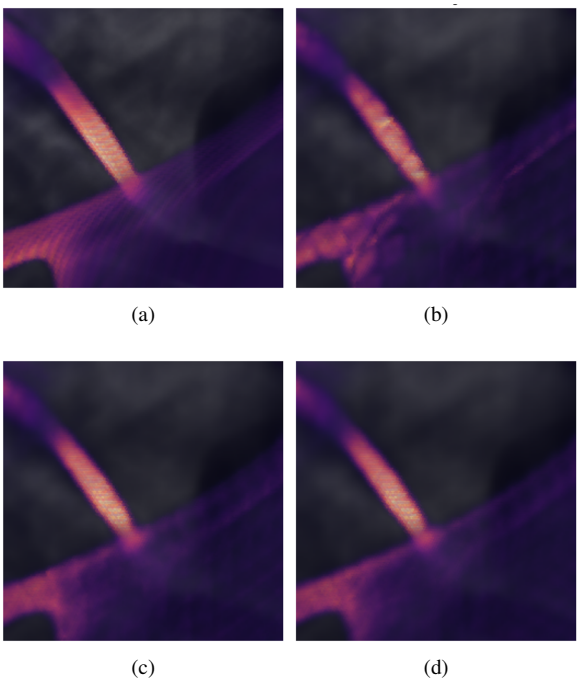

Figure 2: The effect of using an overlapping margin. The original (a) is reconstructed with no margin (b), a two-wide margin without interpolation (c) and a two-wide margin with interpolation (d).

However, there are many possible improvements to the implemented approach, which have remained out of the scope of this seminar. Firstly, larger models should be trained for longer on a larger dataset to ensure a complete fit on the conjectured block space. Similarly, different parameter configurations, such as different block sizes and latent dimensions, as well as different encoder-decoder architectures, such as convolutional AE, should be tested to identify the optimal trade-off between the compression factor, reconstruction loss and encoding/decoding speed. Of course, for a more legitimate evaluation, the implemented approach should also be tested against existing neural methods as well as baselines like a 3D discrete cosine transform (DCT) approach.

References

[V23] Volumes. https://drive.google.com/drive/folders/1vxAHxXPUj9Z-xBkygZKXBKXbI4oDwGn3. Accessed: 2023-04-20. 2

[WMB*11] WOODRING J., MNISZEWSKI S., BRISLAWN C., DEMARLE D., AHRENS J.: Revisiting wavelet compression for large-scale climate data using jpeg 2000 and ensuring data precision. In *2011 IEEE symposium on large data analysis and visualization* (2011), IEEE, pp. 31–38. 1

[YL95] YEO B.-L., LIU B.: Volume rendering of dct-based compressed 3d scalar data. *IEEE Transactions on Visualization and Computer Graphics 1*, 1 (1995), 29–43. 1

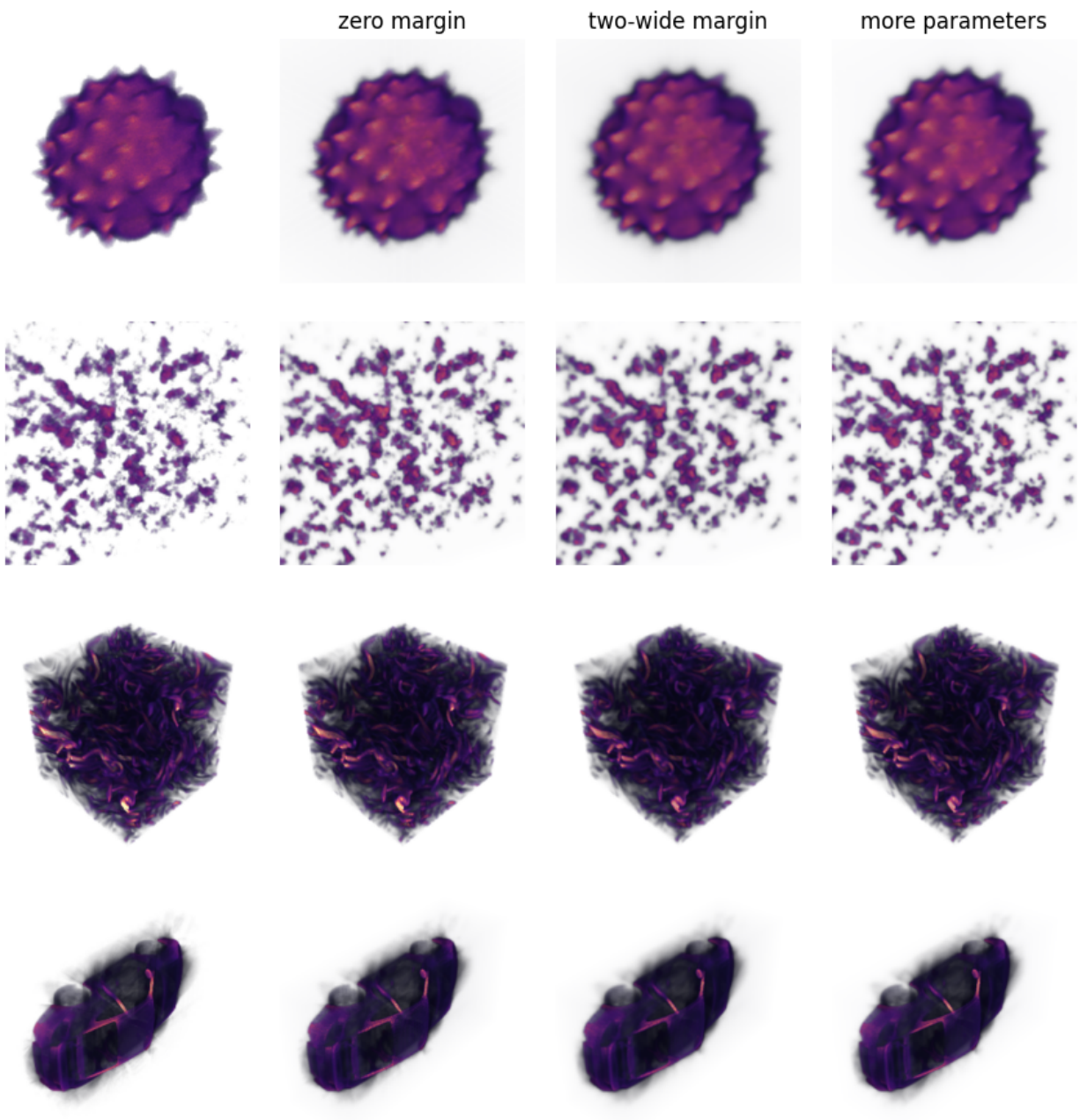

Figure 3: Different volumes (left column) reconstructed by the evaluated models.

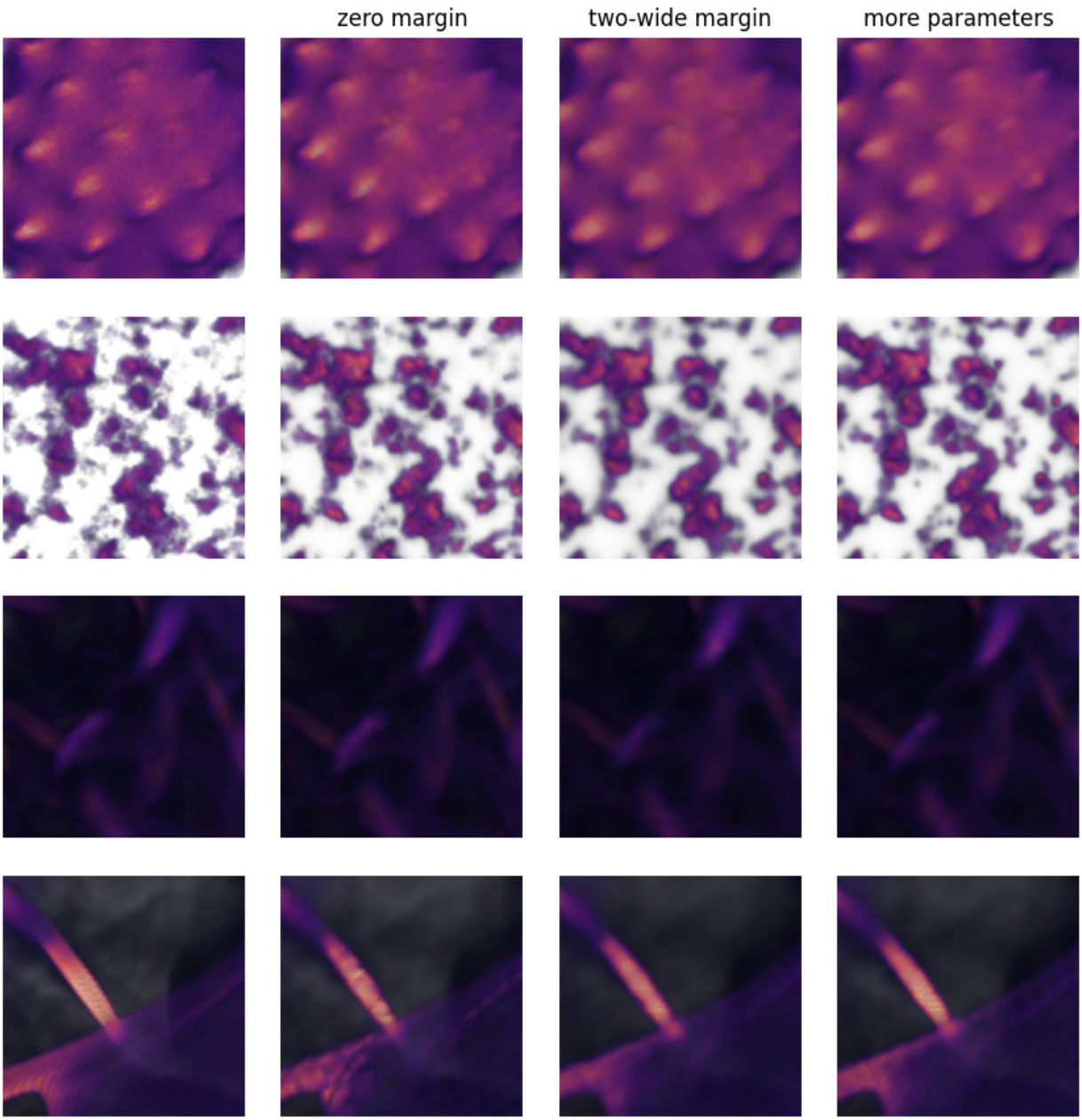

Figure 4: Different volumes (left column) reconstructed by the evaluated models at a higher zoom level.