3D Post-Stack Seismic Data Compression With a Deep Autoencoder

Ana Paula Schiavon, Kevyn Swhants dos Santos Ribeiro, João Paulo Navarro,
Marcelo Bernardes Vieira, and Pedro Mário Cruz e Silva

***Abstract*—We approach the problem of 3D post-stack seismic data compression by training a model based on a deep autoencoder. Our network architecture is trained to consider the similarity between 3D seismic sections drawn from one or multiple seismic volumes. A whole seismic volume is compressed with the latent representations of each of its composing volumetric sections. The goal is to compress the seismic data at very low bit rates with high-quality reconstruction. Our model is suitable for training general compressors from multiple seismic surveys or for specialized compression of a single seismic volume. Results show that our method can compress seismic data with extremely low bit rates, below 0.3 bpv (bits-per-voxel), while yielding Peak Signal-to-Noise Ratio (PSNR) values over 40 dB.**

***Index Terms*—Seismic Data Compression, 3D Post-stack Data, Deep Learning, Autoencoder.**

I. Introduction

SEISMIC data are mappings from the Earth's subsurface that reveal a representation of geological structures present in the regions where they were acquired. The necessity of oil and gas exploration has led to improvements in the acquisition methods. Due to the high-quality of the acquisition sensors, the data needs hundreds of terabytes to be stored or transmitted, motivating their compression [1]–[3]. In general, the seismic data can be pre-stack or post-stack. Pre-stack contains a rather raw and redundant information. Post-stack data is a processed version in which the signal redundancies are attenuated. There exists a demand for pre-stack compression, as they are bigger and less processed than post-stack. However, the post-stack compression is of interest in applications that need huge processed seismic data transmission or storage. In this work, we only consider the compression of 3D post-stack volumes uniformly discretized in a 3D grid.

Several methods have been proposed to solve the problem of seismic data compression. Considered as transform-based methods, the works [4]–[6] approach the task by transforming the seismic data into a different domain whence the sparsity and correlation among coefficients provide a smaller representation. An approach based on dimensionality reduction was proposed by [7] where signal dimensions are reduced using a Principal Component Analysis (PCA) method. Taking into account the progress in video and image compression, some methods were proposed, by adapting the JPEG-XR [8] and the HEVC [9], [10] codecs for seismic data compression.

Considering the success of deep learning methods to solve computer vision [11] and geophysics tasks [12], [2] proposed to compress 3D post-stack seismic surveys through deep neural networks. Sustained on the premise that neural networks can learn important information directly from data for compression tasks, the main hypothesis is that CNNs can compress seismic data at low bit rates, preserving most of its underlying structural information. The work [1] adapted the method proposed by [3] to deal with the seismic domain, splitting a post-stack volume as a set of 2D slices. We refer to this method as 2D-based Seismic Data Compression (2DSC).

A.P. Schiavon, K.S. Ribeiro and M.B. Vieira are affiliated to UFJF - Universidade Federal de Juiz de Fora, Juiz de Fora, Minas Gerais, Brazil.
J.P. Navarro and P.M. Silva are affiliated to NVIDIA.

Seismic data can be compressed using 2D sections [1]. But higher compression is possible by exploiting the post-stack 3D seismic data as a volume. Assuming that seismic data is locally similar, our hypothesis is that a set of sections can be represented with the same latent representation of a single slice. As the main contribution, we propose the Multi-Channel Seismic Data Compression (MCSC) method. The proposal is to stack multiple consecutive 2D seismic slices extracted from the volume. The stack is thus a 3D seismic volume section from which the neural network can learn volumetric similarities. The goal is to extend the work from [1] to compress the seismic data at low bit rates. It is expected that the correlation among slices provide smaller bit rates compared to the 2DSC method.

II. Method

The basis of our approach was proposed by [3], and adapted by [1] to compress 2D seismic sections extracted from 3D post-stack data. In that work, bi-dimensional slices were extracted from the volume and each one was compressed separately. In the reverse process, the compressed slices coefficients are decompressed and the whole volume is reconstructed. In addition, training and inference schemes were proposed to allow the compression of seismic data having different characteristics. In this work, the approach proposed

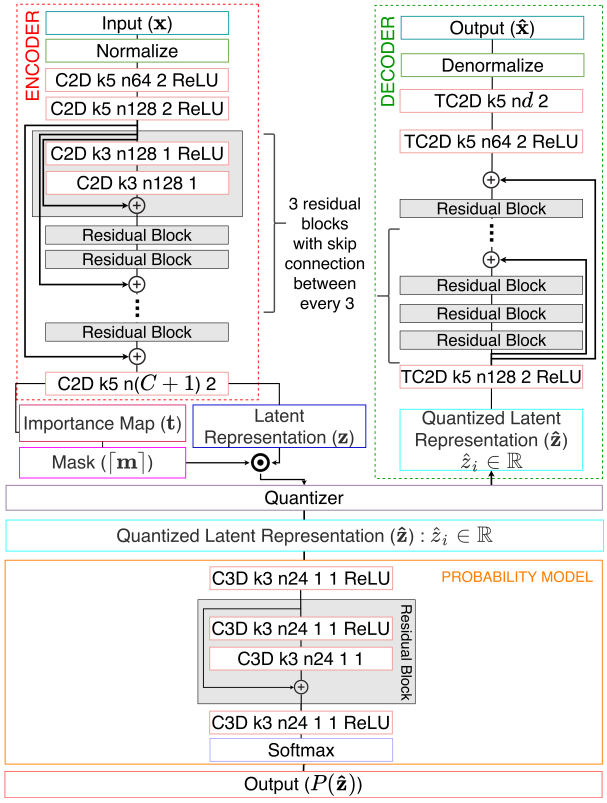

Fig. 1: Autoencoder and probability model architectures.

by [1] is extended to directly compress 3D regions of the volume. A set of 2D slices is concatenated, yielding a volume of dimensions $h \times w \times d$, where h and w are the spatial dimensions. The dimension d is the number of 2D slices that compose the volume and are stacked as input channels in Figure 1.

Figure 1 shows the architecture used in this work. It is composed of an encoder and a decoder. The "normalize" layer normalizes the input using the mean and variance of the training set and the "denormalize" performs the inverse operation. From the encoder, "C2D k5 n64 2 ReLU" is the convolution 2D with kernel size 5, 64 filters, stride 2 for both spatial dimensions w and h and ReLU as activation function. The last layer of the encoder performs a convolution with $C+1$ filters, yielding the latent representation $\mathbf{z} \in \mathbb{R}^{h/8 \times w/8 \times C}$ and an importance map $\mathbf{t} \in \mathbb{R}^{h/8 \times w/8 \times 1}$, where the value 8 is the fixed network subsampling factor for spatial dimensions w and h, and the parameter C indicates the number of feature maps of the latent representation. Then $\mathbf{z}$ is masked and quantized and its quantized representation $\hat{\mathbf{z}}$ is used to feed the decoder side and the probability model $\mathbf{P}$. The decoder mirrors the encoder but performing transposed convolutions to make output equals input. The transposed convolution is represented as "TC2D". All convolutional layers are normalized using batch normalization. The probability model $\mathbf{P}$ is a 3D CNN that estimates the bit rate through $\hat{\mathbf{z}}$.

The autoencoder is composed of an encoder $\mathbf{E} : \mathbb{R}^{h \times w \times d} \rightarrow \mathbb{R}^{h/8 \times w/8 \times C}$ that maps an input data $\mathbf{x}$ with dimensions $h \times w \times d$ to a latent representation $\mathbf{z} = \mathbf{E}(\mathbf{x})$. It differs from [1], in which a single slice with size $h \times w$ is used as input. The input volume of dimensions $h \times w \times d$ is mapped into a latent representation of dimensions $h/8 \times w/8 \times C$, that are the same from [1]. Therefore, we assume that the latent space is inherently capable of representing similarities of multiple slices. The quantizer $\mathbf{Q} : \mathbb{R}^{h/8 \times w/8 \times C} \rightarrow \mathcal{C}^{h/8 \times w/8 \times C}$ discretizes the entries of $\mathbf{z}$ using a finite set of L centroids $\mathcal{C} = \{c_1, \ldots, c_L\}, c_l \in \mathbb{R}$, yielding $\hat{\mathbf{z}} = \mathbf{Q}(\mathbf{z})$. The decoder $\mathbf{D} : \mathcal{C}^{h/8 \times w/8 \times C} \rightarrow \mathbb{R}^{h \times w \times d}$ then forms the reconstructed data $\hat{\mathbf{x}} = \mathbf{D}(\hat{\mathbf{z}})$. In this case, the decoder reconstructs the whole 3D stack of slices.

The quantization, proposed by [3], is performed in two steps. Considering that the visual information is spatially variant in an image, a mask quantization is proposed to make the bit rate allocation locally adaptive. In this setting, a single channel, also called as importance map ($\mathbf{t}$), is added to the last layer of the encoder and a mask $\mathbf{m}$ is generated so that the network learns to weight the latent representation regions that provide the most relevant information for the reconstruction of the image. The latent representation $\mathbf{z} \in \mathbb{R}$ is a mapped version of the 3D seismic data input already having several values close to zero. The quantization step is performed with a finite set $\mathcal{C} \subset \mathbb{R}$ with L scalar centroids learned by the autoencoder. The final discretization takes place by assigning each latent value z_i to the nearest learned centroid in $\mathcal{C}$, forming the quantized entries $\hat{z}_i \in \mathcal{C}$.

The conditional distribution of the quantized latent representation $\hat{\mathbf{z}}$ is needed to estimate the obtained bit rate:

$$p(\hat{\mathbf{z}}) = p(\hat{z}_1, \ldots, \hat{z}_n) = \prod_{i=1}^{n} p(\hat{z}_i | \hat{z}_{i-1}, \ldots, \hat{z}_1), \tag{1}$$

where $n = w/8 \cdot h/8 \cdot C$. A model that combines the proposals [13], [14] was presented by [3] to compute the bit rate needed to compress $\hat{\mathbf{z}}$. The probability model $\mathbf{P}$ is used to estimate the joint distribution $p(\hat{\mathbf{z}})$ of the quantized latent representation $\hat{\mathbf{z}}$.

The problem of learning a conditional distribution can also be formulated as a classification task. Given a sample $\hat{z}_i$ and a set of centroids $\mathcal{C} = \{c_1, \ldots, c_L\}$, we want to predict the probabilities of matching $\hat{z}_i$ to each centroid c_l from $\mathcal{C}$, according to their similarity. Thereby, each distribution $p(\hat{z}_i | \hat{z}_{i-1}, \ldots, \hat{z}_1)$ provides the probability of classifying $\hat{z}_i$ as each of the L centroids given the previous values.

A 3D CNN $\mathbf{P} : \mathcal{C}^{w/8 \times h/8 \times C} \rightarrow \mathbb{R}^{w/8 \times h/8 \times C \times L}$ is used to estimate each term $p(\hat{z}_i | \hat{z}_{i-1}, \ldots, \hat{z}_1)$:

$$P_{i,l}(\hat{\mathbf{z}}) \approx p(\hat{z}_i = c_l | \hat{z}_{i-1}, \ldots, \hat{z}_1), \tag{2}$$

where $P_{i,l}$ is the probability of each symbol $\hat{z}_i$ to be assigned to each centroid c_l of $\mathcal{C}$. The model $\mathbf{P}$ is trained with a loss:

$$\mathcal{L}_{\mathbf{P}} = \mathbf{d}(\mathbf{x}, \hat{\mathbf{x}}) - \beta \frac{1}{n} \sum_{i=1}^{n} \sum_{l=1}^{L} \mathbf{y}_{i,l} \log P_{i,l}(\hat{\mathbf{z}}), \tag{3}$$

where $\mathbf{d}$ is an equation that measures the distortion achieved when compressing $\mathbf{x}$ at a certain bit rate. The negative term is the cross-entropy loss that expresses the number of bits per pixel required to compress the latent representation $\hat{\mathbf{z}}$ using $\mathbf{P}$ as a probability model. The term β controls the trade-off between bit rate and distortion.

The autoencoder uses the probability model loss to deal with the rate-distortion trade-off. To control the bit rate, the autoencoder weights up the $P(\hat{\mathbf{z}})$ using the mask $\lceil \mathbf{m} \rceil$. The weighting is a way to easily control the coding cost, by increasing/decreasing the value of the importance map $\mathbf{t}$ for some spatial locations, obtaining fewer/more zero entries in $\lceil \mathbf{m} \rceil$. The loss function of the autoencoder $(\mathbf{E}, \mathbf{D})$ and the quantizer $\mathbf{Q}$ is given by:

$$\mathcal{L}_{\mathbf{E},\mathbf{D},\mathbf{Q}} = \mathbf{d}(\mathbf{x}, \hat{\mathbf{x}}) - \underbrace{\beta \frac{1}{n} \sum_{i=1}^{n} \sum_{l=1}^{L} \lceil m_i \rceil \, \mathbf{y}_{i,l} \log P_{i,l}(\hat{\mathbf{z}})}_{\mathbf{r}(\hat{\mathbf{z}})}. \tag{4}$$

Different from [3], we use the Peak Signal-to-Noise Ratio (PSNR) as distortion function instead of the proposed MS-SSIM. The MS-SSIM is a perceptual metric, suitable for image compression tasks. The PSNR is a generic metric that can be used to evaluate the similarity between two scalar fields, such as the post-stack data. As the results show, it was sufficient for compressing arbitrary seismic data. However, our proposals are not suitable for general-purpose 3D scalar field compression.

III. Training and Inference Procedures

The training and inference steps were performed similarly as described in [1]. As a pre-processing step, the seismic surveys are normalized to the [0,1] interval using its minimum and maximum values. It is needed because the seismic data is quantized with 32-bit floating-points and its range values are wider and the min-max values are arbitrary across different volumes. To make possible the training of a model capable of compressing different seismic surveys, we need to put all of them at the same conditions.

The batch generation step is performed as depicted in Figure 2. Initially, we extract a set of sub-volumes from the whole seismic volume $v \in \mathbb{R}^{H \times W \times D}$. The sub-volumes can be extracted by considering the inline, crossline and time-depth directions along the axes x, y and z, respectively. For inline sub-volumes, for instance, the samples are formed of d bidimensional crops of size $h \times w$ extracted along the x axis. Similarly, for crossline and time-depth are composed of samples along the axes y and z, respectively.

To extract sub-volumes from the inline direction, we consider the volume indexed according to the raster scan order in which the row is the y axis, the column is the z axis and the depth is the x axis. Using this order, we extract all non-overlapping sub-volumes of size $h \times w \times d$. If some dimension H, W, or D of the volume is not divisible by the corresponding dimension h, w, or d in the sub-volume, it is allowed the smallest overlapping sub-volume containing the remaining voxels. Figure 2 shows an example in which the dimension W is not divisible

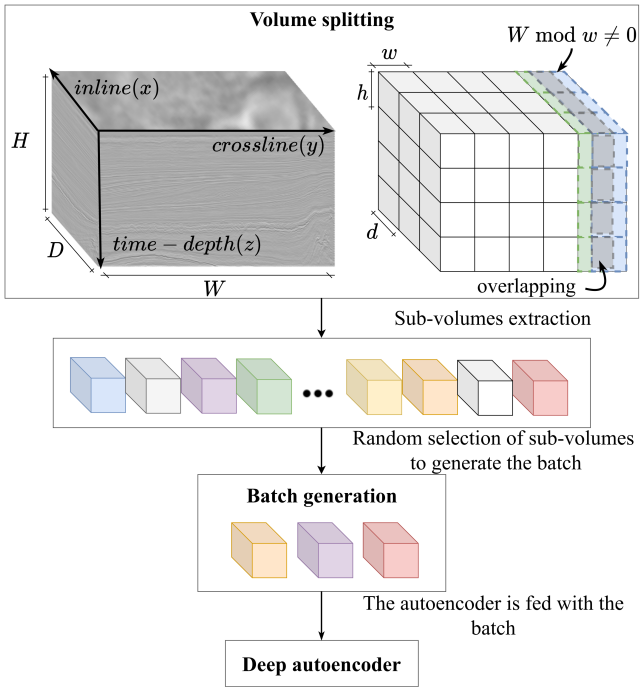

Fig. 2: Batch generation step. The volume is detached in a set of sub-volumes that are randomly selected to generate the batch.

by w. In this sense, the last sub-volumes extracted (blue) have $w - (W \bmod w)$ columns overlapping the previous sub-volume extracted (green). An analogous process is used to extract crossline and time-depth samples.

The batch is generated by randomly selecting sub-volumes extracted from the input volume. In the case of a training set composed of various datasets, we attempt to reduce the dataset bias by building the batch with samples from all of them at the same amount. The number of samples of a dataset can vary, and smaller datasets provide repeated samples.

We can use sub-volumes from all directions to train the network. We use only the inline and crossline directions to train our models due to their similarity. The time-depth direction has very different structures and is often too noisy when compared to the other directions. The batch has the same amount of sub-volumes for each direction. The validation and testing steps consider only the inline samples. With the batch generated, it is used to feed the autoencoder model. This process is repeated until the maximum number of iterations is reached.

Volumes of size $H \times W \times d$ are extracted for the inference. If their shapes are not divisible by the network subsampling factor, they are padded with a border extension. We propose the symmetric border extension since it better preserves the frequencies of the seismic volume. The autoencoder is fed with the volume and both input and output are unpadded to guarantee coherence of the metric evaluation. The volumes are denormalized to reconstruct the compressed seismic volume and the error between the original and reconstructed volumes is evaluated.

A possible end-user pipeline can be as follows: the network is trained from scratch with the seismic volume. In a generalist approach, the training data is a volume (or multiple volumes) that can generalize well the testing domain. In this case, the volume of interest to be compressed is not used to train the network. In a specialized compression scheme, the volume used to train is the same to be compressed. The training step considers a percentage $\gamma \in [0, 100]$ from all sub-volumes extracted from the input data. The parameter γ can be arbitrary or even 100%. After training, the compression is performed in the whole volume of interest. The compressed representation is stored as well as the decoder weights. To decompress, the decoder weights are recovered and the inverse transformation is applied.

IV. Experimental Results and Discussion

Our method was implemented using the TensorFlow framework, and all runs performed on GPUs NVIDIA Tesla V100 of 32GB. Our models have $950,534$ trainable parameters in total, requiring 42MB to be stored. The hyper-parameter space was: Adam optimizer, batch-size of 30, initial learning rate of $8 \cdot 10^{-5}$ with step decay of 0.05 every 10 epochs for both networks, volume size of $160 \times 160 \times d$, latent representation of size $C = 64$, $L = 50$ centroids and $\beta = 100$. Parameters $C = 64$, $L = 50$ were empirically chosen since they are not very sensitive. We trained our models for 30 epochs. The reconstruction quality is reported as PSNR in decibels (dB) due to its sensibility to small error variations, and compression rate as bits-per-voxel (bpv), expressing the average number of bits necessary to represent a single seismic amplitude value.

We selected eight real seismic surveys from SEG Open Data repository[1]: Kahu3D, Kerry3D, Netherlands F3-Block, Opunake3D, Parihaka3D, Penobscot3D, Poseidon3D, and Waihapa3D. Samples were extracted from the inline (x) and crossline (y) directions. The model with the smallest validation loss during the training step was selected.

As proposed by [3], a clipping is used on the entropy term to force the bit rate target r_t to be reached when training. The $\max(\beta \mathbf{r}(\hat{\mathbf{z}}), r_t)$ is used instead of $\beta \mathbf{r}(\hat{\mathbf{z}})$ in Equation 4 when optimizing the autoencoder. For an objective comparison, we set the bit rate target $r_t = 1.0$ bpv. For experimental evaluation, we select the parameters that provide the best PSNR. But the target bit rate might not be reached due to divergent network learning. Therefore, we exclude

[1] https://wiki.seg.org/wiki/Open_data

Train \ Test	Opunake3D	Penobscot3D	Kahu3D	Parihaka3D	Waihapa3D	N. F3-Block	Kerry3D	Poseidon3D
Opunake3D	47.66 / 1.12	46.85 / 1.15	44.81 / 1.12	48.32 / 1.14	41.85 / 1.10	37.66 / 1.14	32.35 / 1.11	29.45 / 1.12
Penobscot3D	47.10 / 1.28	35.63 / 1.18	44.60 / 1.30	44.20 / 1.28	44.49 / 1.28	34.57 / 1.26	31.06 / 1.43	26.14 / 1.57
Kahu3D	50.53 / 1.08	48.11 / 1.09	44.55 / 1.13	45.83 / 1.12	45.72 / 1.19	37.03 / 1.12	34.18 / 1.14	30.43 / 1.11
Parihaka3D	**54.87 / 0.89**	**49.60 / 0.95**	**49.48 / 1.00**	**51.57 / 0.95**	**46.57 / 1.05**	**38.77 / 1.01**	35.42 / 1.16	**30.85 / 1.19**
Waihapa3D	48.82 / 1.21	46.61 / 1.19	45.75 / 1.24	43.62 / 1.25	43.07 / 1.19	35.83 / 1.25	32.31 / 1.35	29.09 / 1.45
N. F3-Block	47.77 / 1.20	46.34 / 1.16	46.33 / 1.16	48.49 / 1.15	43.08 / 1.17	38.73 / 1.09	37.24 / 1.15	33.26 / 1.22
Kerry3D	47.83 / 1.16	45.03 / 1.13	46.56 / 1.18	46.22 / 1.17	43.56 / 1.20	37.90 / 1.15	**37.88 / 1.12**	32.32 / 1.31
Poseidon3D	34.39 / 1.40	31.84 / 1.39	30.72 / 1.39	29.14 / 1.39	26.77 / 1.39	28.43 / 1.39	25.44 / 1.39	40.28 / 1.39

TABLE I: Leave-one-in protocol results for the MCSC method, reported as PSNR/bpv. Painted cells indicate the top 3 best results for each testing survey, with darker colors meaning a better result. Values in the diagonal result from reconstructions of the validation set and not from the reconstruction of the whole volume. We consider only results below the margin threshold (1.2 bpv) of acceptable bpv.

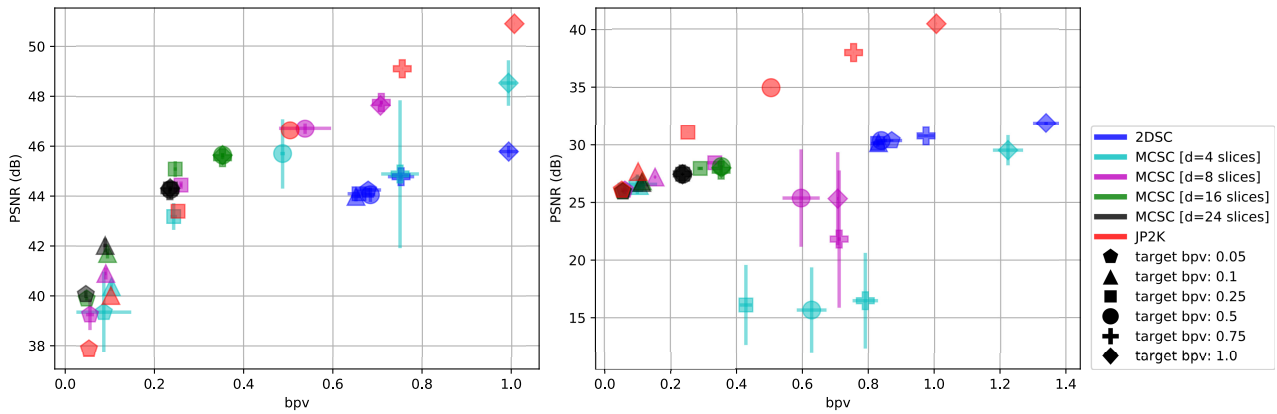

Fig. 3: Method comparison for Penobscot3D (left) and Poseidon3D (right) testing surveys. Lines indicate the standard deviation for PSNR (vertical) and bpv (horizontal).

from comparisons the results having bpv $> 1.2 \cdot r_t$ since they become inadequate even yielding a high PSNR. This margin is enough to deal with differences arising from network initialization.

To evaluate the compression capabilities of our method over different surveys, we use the leave-one-in protocol [15] such that one survey is used for training and the remaining ones are used for testing. The leave-one-out protocol could also be implemented, but we limited our scope to evaluate the generalization capabilities of our network by training with individual volumes. A future analysis is needed to define the best protocol to compress seismic data. For example, it is possible to train with multiple volumes to obtain a generalist, ready to use, compressor. Another possibility is to conduct specialist fine-tuning of the generalist network with the arbitrary target volume. Another possibility is to consider a specialized training for each survey.

For the leave-one-in protocol, we extract sub-volumes of depth $d = 4$ from the training survey and split the training set, so that the first 10% samples from the training data are used for validation, according to the order described in Section III. Table I shows the results of the leave-one-in protocol. Notice that the method performance depends on the survey used for training. In a generalist approach, the Parihaka3D was the best training survey, and the worse result was found using the Poseidon3D. In general, the presence of noise and high frequencies is directly related to compression quality. The Parihaka3D has a balance between high and low frequencies, providing enough information for a generalist training. In contrast, the Poseidon3D is a high frequency and noisy survey, which hinders the learning generalization, since the method becomes over-fitted. It occurs because our method considers that neighbor slices are similar and perform a mapping from a volume into a set of bi-dimensional feature maps. The noise leads to arbitrary patterns that are difficult to be recovered in a generalist approach. In this case, only with a specialist model it is possible to reach PSNR values up to 40dB.

To evaluate the performance of the method over different input depth sizes, we perform experiments with $d = 4$, 8, 16 and 24 slices. We set the Parihaka3D as the training survey. We perform experiments for a bit rate target $r_t = 0.05$, 0.10, 0.25, 0.50, 0.75 and 1.0 bpv. Figure 3 shows the results for the Penobscot3D and Poseidon3D testing surveys for each bit rate target with an average of 3 rounds. Lines indicate the standard deviation for PSNR (vertical) and bpv (horizontal). Notice that, for the Penobscot3D survey, as the bit rate target decreases, higher PSNR results are achieved by using a higher number of slices d. The same does not occur for the Poseidon3D survey, in which fewer slices provide higher PSNR values, considering smaller bit rates target. We believe that this is due to the presence of noise in these surveys. We conclude that the network performance tends to increase with d, but occasionally it does not occur in presence of noise. There is not an optimal value for d on all situations, but with $d = 16$ slices it is possible to achieve reasonable PSNR and bit rate values for most testing sets.

Figure 4 shows a qualitative compression result of a 2D slice from the Penobscot3D testing survey, extracted in the inline direction. Even with an extreme compression rate of 320:1, the details are fairly preserved with PSNR = 42.50 dB.

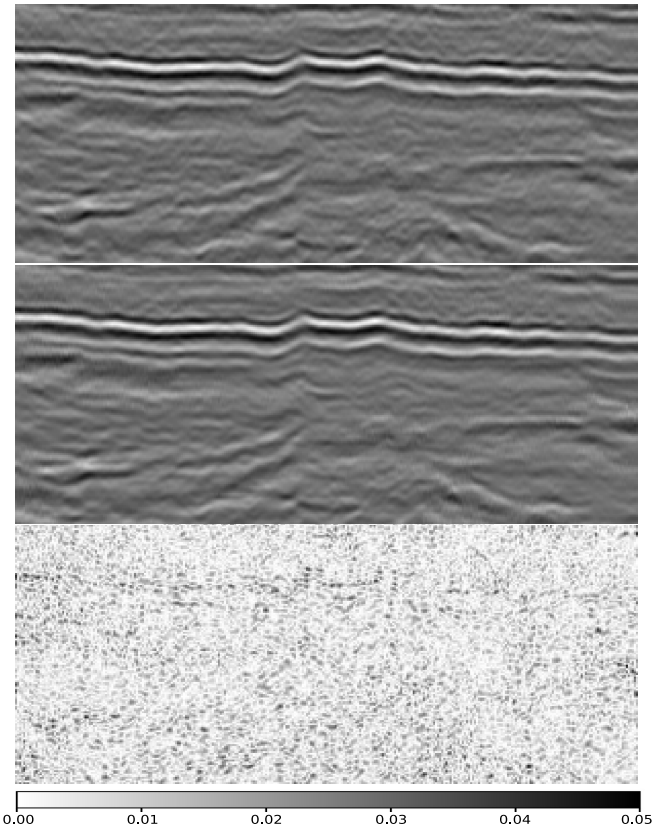

Fig. 4: Reconstruction of a 2D slice from the Penobscot3D survey. From top to bottom: original slice, reconstructed output and relative error between them. For a bit rate of 0.10 bpv, the network reaches a PSNR = 42.50 dB.

Aiming to compare our method to others in literature, we perform the same experiment considering the 2DSC and the JPEG2000 compressors. From Figure 3, we can see that for the Penobscot3D set and bit rates below 0.25 bpv, our method surpasses the JPEG2000 for all targets evaluated. From this point up, the JPEG2000 achieves higher PSNR. The 2DSC had the worst performance and only obtained bit rates greater than 0.6 bpv. For the noisy volume Poseidon3D, however, our method is competitive to the JPEG2000 only for the extremely low bit rate of 0.05 bpv. It occurs due to the presence of noise in the testing set. Nevertheless, it is an evidence that extreme bit rates are attainable by CNNs. The proposed MCSC setup was not sufficient to deal with high presence of noise in the testing surveys. It suggests that the training protocol and network setup need improvements to deal with surveys of different characteristics in a generalist approach. Similarly to Penobscot3D, the 2DSC could not compress with bit rates smaller than 0.8 bpv, but provided reasonable results with 4 and 8 slices if compared to the MCSC method.

Regarding the compression time, for a single sub-volume of size $160 \times 160 \times 4$, the MCSC takes 0.031 seconds to encode, while the JPEG2000 takes 0.036 seconds, on average. To decompress the sub-volume, the MCSC takes 0.026 seconds and the JPEG2000 takes 0.016 seconds. For the Parihaka3D survey, one training epoch takes about 326 seconds. In our experiments, it took about 3 hours to train a model for 30 epochs. These times were measured using a single NVIDIA Quadro GV100 with 32GB, and a Core i7 870 with 16GB of RAM.

The results evinced that we can surpass the JPEG2000 for very low compression rates, i.e. bit rates smaller than 0.3 bpv. In addition, our scheme can be adjusted to compress seismic volumes in a generalist or specialist way. However, as the network was designed to minimize the bit rate as much as possible, a major observed drawback is that it is very difficult to compress the seismic data in a wide range of target bit rates. In general, the performance worsens as the target bit rate increases. The network tends to find the best PSNR only below a given target bit rate. But it is possible to achieve better results by adapting the architecture to focus on higher bit rates instead of extreme compression. In such case, multiple compression streams could be used to find the best compression.

V. Conclusion

This work tackles the 3D post-stack seismic data compression problem with very low bit rates $bpv < 0.3$. We propose a model and training protocol based on a deep autoencoder to find redundancy in volumetric seismic sections drawn from one or multiple seismic volumes. Our method can be tuned as a generalist or a specialist compressor. The performance in both cases depends on the surveys and protocol used for learning. Our results show fair evidences that CNN based compression can overcome state-of-the-art compressors like JPEG2000 for very low bit rates.

A future investigation is needed to evaluate the best protocol to compress seismic data by using the proposed neural network. It is possible to perform a generalist training with multiple volumes, a specialist fine-tuning for a given volume, or even use a trained network for each survey setup. Also, the autoencoder architecture can be adapted to perform multi-resolution compression or to model a known residual error distribution, such as the white noise, through generative networks. These improvements can make the method less sensitive to noise and less prone to generate reconstruction distortions.

Acknowledgment

Authors thank CAPES and FAPEMIG for the financial support, and NVIDIA Corporation for the GPU Grant Program.

References

[1] A. P. Schiavon, J. P. Navarro, M. Vieira, and P. M. C. e Silva, "Low bit rate 2d seismic image compression with deep autoencoders," in *Int. Conf. on Computational Science and Its Applications*, 2019, pp. 397–407.

[2] J. Navarro, A. Schiavon, M. Vieira, and P. Silva, "Deep seismic compression," in *81st EAGE Conference*, vol. 2019, no. 1. European Association of Geoscientists & Engineers, 2019, pp. 1–5.

[3] F. Mentzer, E. Agustsson, M. Tschannen, R. Timofte, and L. Van Gool, "Conditional probability models for deep image compression," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[4] A. S. Spanias, S. B. Jonsson, and S. D. Stearns, "Transform methods for seismic data compression," *IEEE Transactions on Geoscience and Remote sensing*, vol. 29, no. 3, pp. 407–416, 1991.

[5] G. Aharoni, A. Averbuch, R. Coifman, and M. Israeli, "Local cosine transform - a method for the reduction of the blocking effect in jpeg," in *Wavelet Theory and Application*. Springer, 1993, pp. 7–38.

[6] A. Z. Averbuch, F. Meyer, J. . Stromberg, R. Coifman, and A. Vassiliou, "Low bit-rate efficient compression for seismic data," *IEEE Transactions on Image Processing*, vol. 10, no. 12, pp. 1801–1814, Dec 2001.

[7] H. H. Nuha, B. Liu, M. Mohandes, and M. Deriche, "Seismic data compression using signal alignment and pca," in *2017 9th IEEE-GCC Conference and Exhibition (GCCCE)*. IEEE, 2017, pp. 1–6.

[8] Y. Liu, Z. Xiong, L. Lu, and D. Hohl, "Fast snr and rate control for jpeg xr," in *Signal Processing and Communication Systems (ICSPCS), 2016 10th International Conference on*. IEEE, 2016, pp. 1–7.

[9] M. Radosavljević, Z. Xiong, L. Lu, and D. Vukobratović, "High bit-depth image compression with application to seismic data," in *Visual Communications and Image Processing*. IEEE, 2016, pp. 1–4.

[10] M. Radosavljević, Z. Xiong, L. Lu, D. Hohl, and D. Vukobratović, "Hevc-based compression of high bit-depth 3d seismic data," in *IEEE International Conference on Image Processing*. IEEE, 2017, pp. 4028–4032.

[11] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems 25*. Curran Associates, Inc., 2012, pp. 1097–1105.

[12] Y. Shi, X. Wu, and S. , "Automatic salt-body classification using a deep convolutional neural network," in *SEG Technical Program*. Society of Exploration Geophysicists, 2018, pp. 1971–1975.

[13] A. Van Den Oord, N. Kalchbrenner, and K. Kavukcuoglu, "Pixel recurrent neural networks," in *Intern. Conf. on Machine Learning*, 2016, pp. 1747–1756.

[14] A. Van Den Oord, N. Kalchbrenner, L. Espeholt, O. Vinyals, A. Graves *et al.*, "Conditional image generation with pixelcnn decoders," in *Advances in Neural Information Processing Systems*, 2016, pp. 4790–4798.

[15] L. Trippa, L. Waldron, C. Huttenhower, G. Parmigiani *et al.*, "Bayesian nonparametric cross-study validation of prediction methods," *The Annals of Applied Statistics*, no. 1, pp. 402–428, 2015.