UNIVERSIDADE FEDERAL DE JUIZ DE FORA

INSTITUTO DE CIÊNCIAS EXATAS

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Ana Paula Schiavon

Post-stack Seismic Data Compression with Multidimensional Deep Autoencoders

Juiz de Fora

2020

Ana Paula Schiavon

Post-stack Seismic Data Compression with Multidimensional Deep Autoencoders

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação, do Instituto de Ciências Exatas da Universidade Federal de Juiz de Fora como requisito parcial para obtenção do título de Mestre em Ciência da Computação.

Orientador: Marcelo Bernardes Vieira

Juiz de Fora

2020

Ficha catalográfica elaborada através do Modelo Latex do CDC da UFJF
com os dados fornecidos pelo(a) autor(a)

Schiavon, Ana Paula.
Post-stack Seismic Data Compression with Multidimensional Deep Autoencoders / Ana Paula Schiavon. – 2020.
89 f. : il.

Orientador: Marcelo Bernardes Vieira
Dissertação (Mestrado) – Universidade Federal de Juiz de Fora, Instituto de Ciências Exatas. Programa de Pós-Graduação em Ciência da Computação, 2020.

1. Seismic Data Compression. 2. Deep Learning. 3. Autoencoder. 4. 3D Post-Stack Seismic Data. 5. Geophysical Image Processing. I. Vieira, Marcelo Bernardes, orient. II. Título.

Ana Paula Schiavon

“Post-Stack Seismic Data Compression with Multidimensional Deep Autoencoders”

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de Juiz de Fora como requisito parcial à obtenção do grau de Mestre em Ciência da Computação.

Aprovada em 13 de março de 2020.

AGRADECIMENTOS

Primeiramente agradeço a Deus.

Agradeço também à minha família, sobretudo à minha mãe Luzia, por todo o apoio empregado durante não só o mestrado, mas desde o início da minha jornada em Juiz de Fora.

Aos amigos que sempre estiveram presentes apoiando e incentivando meus estudos. Em especial ao Fernando, pelo apoio e amizade incondicionais.

Ao Kevyn, por toda a ajuda prestada com os experimentos durante os últimos meses do mestrado.

Ao professor Marcelo, pela paciência e esforço empregados na minha formação desde a graduação.

À NVIDIA, principalmente ao João Paulo e ao Pedro Mário, por fornecerem as placas de vídeo utilizadas durante o desenvolvimento deste trabalho.

Por fim, agradeço à UFJF e ao PGCC por todo o suporte e à CAPES, pelo apoio financeiro sem o qual este trabalho não poderia ser realizado.

“Creator ineffabilis (...) Da mihi intelligendi acumen, retinendi capacitatem, addiscendi modum et facilitatem, interpretandi subtilitatem, loquendi gratiam copiosam; ingressum instruas, progressum dirigas, egressum compleas.”(BRUN-GOUANVIC et al., 1996)

RESUMO

Dados sísmicos são mapeamentos da subsuperfície terrestre que têm como objetivo representar as características geofísicas da região onde eles foram obtidos de forma que possam ser interpretados. Esses dados podem ocupar centenas de Gigabytes de armazenamento, motivando sua compressão. Neste trabalho o problema de compressão de dados sísmicos tridimensionais pós-pilha é abordado usando modelos baseados em autocodificadores profundos. O autocodificador profundo é uma rede neural que permite representar a maior parte da informação contida em um dado sísmico com um custo menor que sua representação original. De acordo com nosso conhecimento, este é o primeiro trabalho a lidar com compressão de dados sísmicos utilizando aprendizado profundo. Dessa forma, através de aproximações sucessivas, são propostos quatro métodos de compressão de dados tridimensionais pós-pilha: dois baseados em compressão bidimensional, chamados Método de Compressão 2D de Dado Sísmico (2DSC) e Método de Compressão 2D de Dado Sísmico usando Multi-resolução (2DSC-MR), e dois baseados em compressão tridimensional, chamados Método de Compressão 3D de Dado Sísmico (3DSC) e Método de Compressão 3D de Dado Sísmico usando Quantização Vetorial (3DSC-VQ). O método 2DSC é o nosso método de compressão do dado sísmico mais simples, onde o volume é comprimido a partir de suas seções bidimensionais. O método 2DSC-MR estende o método anterior introduzindo a compressão do dado em múltiplas resoluções. O método 3DSC estende o método 2DSC permitindo a compressão do dado sísmico em sua forma volumétrica, considerando a similaridade entre seções para representar um volume inteiro com o custo de apenas uma seção. O método 3DSC-VQ utiliza quantização vetorial para relaxar a etapa de codificação do método anterior, dando maior liberdade à rede para extrair informação dos volumes sísmicos. O objetivo deste trabalho é comprimir o dado sísmico a baixas taxas de bits e com alta qualidade de reconstrução em termos de PSNR e bits-por-voxel (bpv). Experimentos mostram que os quatro métodos podem comprimir o dado sísmico fornecendo valores de PSNR acima de 40 dB a taxas de bits abaixo de 1.0 bpv.

Palavras-chave: Compressão de Dado Sísmico. Aprendizado Profundo. Autocodificador. Dado Sísmico Tridimensional Pós-Pilha. Processamento de Imagem Geofísica.

ABSTRACT

Seismic data are surveys from the Earth's subsurface with the goal of representing the geophysical characteristics from the region where they were obtained in order to be interpreted. These data can occupy hundreds of Gigabytes of storage, motivating their compression. In this work, we approach the problem of three-dimensional post-stack seismic data using models based on deep autoencoders. The deep autoencoder is a neural network that allows representing most of the information of a seismic data with a lower cost in comparison to its original representation. To the best of our knowledge, this is the first work to deal with seismic compression using deep learning. Four compression methods for post-stack data are proposed: two based on a bi-dimensional compression, named 2D-based Seismic Data Compression(2DSC) and 2D-based Seismic Data Compression using Multi-resolution (2DSC-MR), and two based on three-dimensional compression, named 3D-based Seismic Data Compression (3DSC) and 3D-based Seismic Data Compression using Vector Quantization (3DSC-VQ). The 2DSC is our simplest method for seismic compression, in which the volume is compressed through its bi-dimensional sections. The 2DSC-MR extends the previous method by introducing the data compression in multiple resolutions. The 3DSC extends the 2DSC method by allowing the seismic data compression by using the three-dimensional volume instead of 2D slices. This method considers the similarity between sections to compress a whole volume with the cost of a single section. The 3DSC-VQ uses vector quantization aiming to extract more information from the seismic volumes in the encoding part. Our main goal is to compress the seismic data at low bit rates, attaining a high quality reconstruction. Experiments show that our methods can compress seismic data yielding PSNR values over 40 dB and bit rates below 1.0 bpv.

Keywords: Seismic Data Compression. Deep Learning. Autoencoder. 3D Post-Stack Seismic Data. Geophysical Image Processing.

LIST OF FIGURES

LIST OF TABLES

LIST OF ACRONYMS

2DSC	2D-based Seismic Data Compression
2DSC-MR	2D-based Seismic Data Compression using Multi-resolution
3DSC	3D-based Seismic Data Compression
3DSC-VQ	2D-based Seismic Data Compression using Vector Quantization
BPG	Better Portable Graphics
CNN	Convolutional Neural Network
DCT	Discrete Cosine Transform
DFT	Discrete Fourier Transform
DNN	Deep Neural Network
FWT	Fast Wavelet Transform
GAN	Generative Adversarial Networks
GDN	Generalized Divisive Normalization
GSM	Gaussian Scale Mixture
HEVC	High Efficiency Video Coding
JPEG	Joint Photographic Experts Group
JPEG2000	Joint Photographic Experts Group 2000
JPEG-XR	Joint Photographic Experts Group Extended Range
KLT	Karhunen-Loève Transform
LCT	Local Cosine Transform
MSE	Mean Squared Error
MS-SSIM	Multi-scale Structural Similarity
PCA	Principal Component Analysis
PSNR	Peak Signal-to-Noise Ratio
RNN	Recurrent Neural Network
SE	Squeeze-Excitation

SGD	Stochastic Gradient Descent
SNR	Signal-to-Noise Ratio
SSIM	Structural Similarity
TCNN	Trimmed Convolutional Network
VAE	Variational Autoencoder
WHT	Walsh-Hadamard Transform

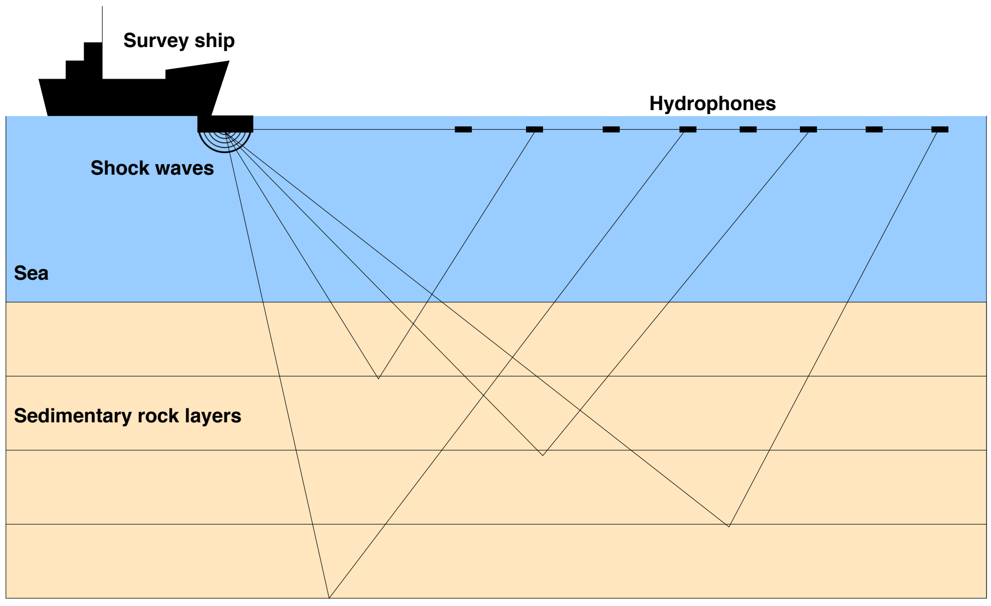
SUMMARY

1 INTRODUCTION

The first attempts to the oil and gas exploration were marked by laborious and expensive mappings from the Earth's subsurface. At that point, oil reservoirs were found through leaks identified at the ocean and drilling regions with high potential to contain them. Besides expensive, it led to inaccurate results, making the process of exploration too slow to be performed. In this sense, it was necessary to develop new techniques aiming to extract the subsurface information in a cheaper and faster way.

Seismic prospecting is the most important method used by industries. In this context, seismic surveys are acquired over regions of interest. They are mappings from the subsurface that reveal the geological structures and properties of the regions where they were obtained. Figure 1 shows the acquisition process. It is based on the seismic reflection process where a controlled source of energy is utilized to generate waves that propagate through the Earth's interior (EVANS, 1997). These waves are reflected and captured by sensors, and a processing process is performed to allow their interpretation. According to Yilmaz (2001), there are three main processing steps: deconvolution, stacking and migration. However, auxiliary processes can be used in some cases, to improve these steps. The seismic data can be stored as pre-stack or post-stack data, and the best strategy depends on the nature of the subsurface, in which analysis from an interpreter is needed. In general, pre-stack contains a lot of redundant information, while in post-stack these redundancies are attenuated. In this work, we approach only the compression of 3D post-stack volumes represented in a matricial form.

Figure 1 – Seismic survey acquisition process.



Source: created by the author.

Recently, advances in the quality of acquisition sensors have led to a substantial increase in data resolution. These surveys can require hundreds of terabytes or even petabytes of storage, yielding higher resolution signals to process, to store, and to transmit. The use of effective compressing algorithms plays an important role in seismic processing, aiming to deal with this increasing data size.

Given a signal, compression can be defined as the task of finding a less costly representation in terms of storage in comparison to the original. In other words, we are interested in finding a transformation that maps a signal to a space suitable for compression. This task can be performed using lossless methods, with perfect signal reconstruction, or in a lossy way, with a greater reduction in storage by allowing reconstruction distortions. Since we are focusing on compressing the seismic data at low bit rates, we approach the seismic compression problem by using lossy compression methods in this work.

The popularity of deep learning algorithms has increased considerably in recent years due to consistent advances in solving successfully complex Computer Science tasks such as image classification (KRIZHEVSKY; SUTSKEVER; HINTON, 2012), person re-identification (LI et al., 2014), action recognition (CARREIRA; ZISSERMAN, 2017), and image compression (MENTZER et al., 2018). In Geophysics, these methods have been used to solve problems such as automatic seismic interpretation, including fault detection (WU et al., 2019), salt detection (SHI; WU; FOMEL, 2018) and Relative Geological Time estimation (GENG et al., 2019). To the best of our knowledge, these methods have not been explored in the seismic data compression field.

The challenge of working with compression for the seismic domain relies on the difficulty of detaching parts of the signal that represent physical properties from those that do not. The variances and inconsistencies that may be present in seismic signals such as noises, interferences, and processing inaccuracies are hard to be captured by handcrafted approaches. We conjecture that deep learning methods are a viable way to face this problem since it has shown to deal efficiently with pattern recognition and image compression. Deep neural networks, mainly the autoencoder-based, are known to be powerful tools to perform signal compression since they can learn adequately the most important features.

Against this background, we propose to explore deep learning-based seismic compression methods. Taking into account the progress of deep learning to solve Computer Vision and Geophysics tasks, we propose to lossy compress 3D post-stack seismic data through deep neural networks. Our main hypothesis is that these networks can compress seismic data at low bit rates preserving most of its underlying structural information. Intending to validate it, we adapt the method proposed by Mentzer et al. (2018). With competitive performance to the state of the art results, they proposed a general-purpose image compression approach. Since image and seismic domains are different, the original method needs to be adapted. As the post-stack data is represented in a 3D matricial form, 2D and 3D approaches can be designed to perform the seismic compression. Bi-dimensional methods consider that the volume can be detached into a set of seismic sections similar to images in which each section is compressed separately. The three-dimensional ones benefit from the correlation between sections by compressing a volume composed of many of them. In this work, we propose four models for post-stack seismic data compression: *2D-based Seismic Data Compression* (2DSC), *2D-based Seismic Data Compression using Multi-resolution* (2DSC-MR), *3D-based Seismic Data Compression* (3DSC) and *3D-based Seismic Data Compression using Vector Quantization* (3DSC-VQ). The 2DSC and 2DSC-MR methods are based on 2D compression while 3DSC and 3DSC-VQ are based on 3D compression.

From the evidence that the method proposed by Mentzer et al. (2018) works well compressing general-purpose images, the most straightforward way to treat the 3D post-stack seismic data compression is considering the volume as a set of 2D slices. The 2DSC method is proposed by adapting the original model to compress seismic sections. Although this approach can be sufficient to compress the data, there is still room for improvement. Considering that the 2DSC method captures information on only one scale, we propose the 2DSC-MR method. Traditional methods that take only one single scale as input fail to capture the scale-dependent information. Thus, our second hypothesis is that the information across scales improves the compression. An architecture similar to Huang et al. (2019) is used, allowing a multi-resolution compression of the post-stack data. The idea is to arrange the data in two scales and progressively compress them from coarse to fine. The multi-resolution structure allows the network to deal with different scales learning the most important features across them.

A three-dimensional compression is expected to yield better results than a bi-dimensional approach since there exists a spatial correlation in all post-stack data directions. The 3D counterpart can learn to capture information in the three directions at the same time. Thus, our third hypothesis is that three-dimensional models can generate representations more suitable for 3D post-stack data compression than bi-dimensional approaches. We propose 3DSC and 3DSC-VQ models aiming to compress directly the seismic volumes. The 3DSC considers the similarity between neighbouring slices, in which the neural network learns a mapping from three-dimensional volumes into bi-dimensional representations. However, some seismic volumes can have noises that make their sections quite different and the depth reduction can lead to high losses. Our fourth hypothesis is that the compression of the whole volume without depth reductions on encoding can improve the quality reached. The 3DSC-VQ method compress the whole volume without encoding reductions to prevent subsampling inaccuracies. The depth dimension is reduced by using vector quantization. The loss from vector quantization is expected to be lower than from the encoding part.

1.1 PROBLEM DEFINITION

The problem tackled in this dissertation is the compression of three-dimensional post-stack seismic surveys using deep learning at low bit rates and with low loss of information, tending to preserve the underlying structures as much as possible.

In this work, the 3D post-stack seismic data is associated with real amplitude values. Thus, we can define a seismic volume $\mathbf{v} \in \mathbb{R}^{\mathfrak{T}\times\mathfrak{C}\times\mathfrak{I}}$, where $\mathfrak{I}, \mathfrak{C}, \mathfrak{T}$ are related to the *inline (x)*, *crossline (y)* and *time-depth (z)* directions, respectively. The problem of lossy compressing a three-dimensional seismic data can be defined as follows: given a volume $\mathbf{v}$, we are interested in finding the functions:

$$f : \mathbb{R}^{\mathfrak{T}\times\mathfrak{C}\times\mathfrak{I}} \to \mathbb{R}^{n}, \tag{1.1}$$

$$q : \mathbb{R}^{n} \to \mathcal{C}^{n/v}, \tag{1.2}$$

and

$$g : \mathcal{C}^{n/v} \to \mathbb{R}^{\mathfrak{T}\times\mathfrak{C}\times\mathfrak{I}}, \tag{1.3}$$

so that $g(q(f(\mathbf{v}))) \approx \mathbf{v}$ and $\mathcal{C} = \{c_1, \ldots, c_L\}$, where $\mathcal{C}$ is a countable set of quantization cells $c_l \in \mathbb{R}^{v}$. The goal is to make $q(f(\mathbf{v}))$ less costly in terms of storage than $\mathbf{v}$.

To address this problem, two aspects need to be evaluated. The first is related to the quality of the reconstruction and the second is related to the cost of representing $\mathbf{v}$ with fewer bits. The quality can be measured with a distortion function. The distortion can be defined as the cost of representing $\mathbf{v}$ using $g(q(f(\mathbf{v})))$. The cost of representing $\mathbf{v}$ with fewer bits is the bit rate (or entropy). The entropy yields the number of bits needed to represent the information in its compressed form. Both distortion $\mathbf{d}$ and entropy $\mathbf{r}$ are minimized in the so-called rate-distortion trade-off $\mathbf{d} + \beta\mathbf{r}$, where β is a scalar that controls the balance trade-off. We aim to make the rate-distortion trade-off as close to zero as possible.

Neural networks are known to be function approximators (HORNIK; STINCHCOMBE; WHITE, 1989) and can be used to approximate f, q and g. In a deep learning approach, the problem can be tackled by designing a deep architecture that allows the seismic data compression. The parameters related to the functions f, q and g can be learned by minimizing the rate-distortion trade-off $\mathbf{d} + \beta\mathbf{r}$ through a set of seismic volumes.

1.2 OBJECTIVES

The main objective of this work is to compress three-dimensional post-stack seismic volumes at low bit rates preserving its qualitative and quantitative aspects as much as possible using deep learning models.

As a secondary objective, we intend to perform an individual validation of our models using real seismic data with different characteristics. Thus, we can experimentally evaluate the performance of our models under different aspects as the presence of noises and frequencies related to different underlying geological structures.

Another secondary objective is to compare the four proposed compression methods according to their compression capability. Besides that, we compare the overall performance to methods from literature.

1.3 HYPOTHESES

In the last years, we can notice the predominance of methods based on deep autoencoders to solve signal compression tasks (AGUSTSSON et al., 2019; RIPPEL et al., 2019; MINNEN; BALLÉ; TODERICI, 2018; TODERICI et al., 2017; THEIS et al., 2017). They are the most natural way of compressing information since they are a type of neural network that aims to approximate the identity function. Autoencoders are often used as dimensionality reducers or signal compressors. With the goal of compressing, it is composed by an encoder, a quantizer and a decoder that are related to the Equations (1.1), (1.2) and (1.3), respectively. The encoder maps the input to a latent representation that it is discretized by the quantizer. The quantizer output is used by the decoder to reconstruct the input. The goal is to represent information with fewer bits, ensuring the similarity between the input and its reconstruction.

Aiming to compress images, Mentzer et al. (2018) proposed a method where two networks are trained simultaneously: a bi-dimensional compressive autoencoder and a three-dimensional probabilistic model. The first deals with the rate-distortion trade-off, controlling the distortion between the input and the output and the cost of representing the input using fewer bits. The second is used to estimate the bit rate associated with the autoencoder latent representation. Achieving competitive results to the state of the art, it inspired us to extend this approach to the seismic domain. Since this scheme was designed to deal with images, we adapt it to work with the post-stack seismic data represented as 3D matrices.

Our main hypothesis is that deep neural networks can compress seismic data at low bit rates preserving most of its underlying structural information. From the premise that the 3D post-stack data can be detached into 2D seismic sections, the most straightforward way to validate our hypothesis is by performing bi-dimensional compression. The model proposed by Mentzer et al. (2018) was designed for general-purpose image compression with 3-channels of 8-bit unsigned integers. The seismic sections are numerically represented as one channel 32-bit floating-points. It is possible to adapt the slices to the network input, but the quantization from 32-bit floating-points to 8-bit unsigned integers introduces precision errors that cannot be recovered. Moreover, the model was originally trained using a perceptual loss, and it does not preserve quantitative aspects that are important to the seismic domain. For this reason, we propose the 2DSC method, by adapting the network and all metrics to work with the post-stack seismic data. Although this approach is sufficient to compress the data, there is still room for improvement.

In the last years, multi-resolution methods have been used for image compression (SKODRAS; CHRISTOPOULOS; EBRAHIMI, 2001; NAKANISHI et al., 2018; HUANG et al., 2019). They work decomposing the signal into different scales such that each one has its feature extractor. Thus, a scale can provide information that is no easily captured by others. Our second hypothesis is that the information across scales improves the compression. Considering that the method proposed by Mentzer et al. (2018) captures information only on one scale, we propose the 2DSC-MR method, with an architecture similar to Huang et al. (2019) to allow a 2D multi-resolution compression of the post-stack data.

The most natural extension of the method proposed by Mentzer et al. (2018) points towards the 3D data compression. From the premise that there exists spatial correlation in all post-stack data directions, our third hypothesis is that three-dimensional models can generate representations more suitable to 3D post-stack data compression than bi-dimensional approaches. By dealing with volumes, the autoencoder becomes three-dimensional and the probabilistic model should be four-dimensional. However, due to memory constraints, a 4D neural network is still not practical. We propose two schemes to attend this problem without needing an expensive 4D model. The 3DSC method takes into account the similarity between slices, mapping a volume into a bi-dimensional representation. There is however a strong simplification in latent space to avoid 4D networks, and the depth reduction can lead to high losses in latent representations. Our fourth hypothesis is that the compression of the whole volume without depth reductions on encoding can improve the quality reached. The 3DSC-VQ uses a higher dimensional latent representation to improve compression. It reduces the latent representation depth dimension by using vector quantization. In this scheme, a set of vectorial centroids is learned and the columns of the volume are replaced by the indices of the nearest centroids.

1.4 RELATED WORK

This section presents the main works from the literature that are a background for this work. To the best of our knowledge, there is no work in seismic data compression that exploits deep learning methods. In this sense, the Subsection 1.4.1 describes only handcrafted approaches for seismic compression. The Subsection 1.4.2 presents deep learning-based methods for image and video compression.

1.4.1 Seismic Data Compression

Over the past decades, various compression methods have been proposed to deal with the amount of information collected from seismic surveys. The first attempts in seismic data compression considered the evaluation of transform-based methods. In general, a transform is applied to the seismic data to represent the input using a different domain and the sparsity and correlation among coefficients are exploited to provide smaller representations.

A comparative study of the transforms Discrete Fourier Transform (DFT), Discrete Cosine Transform (DCT), Walsh-Hadamard Transform (WHT) and Karhunen-Loève Transform (KLT) was proposed by Spanias, Jonsson e Stearns (1991). They used seismic data from the Norwegian Regional Seismic Array to perform their experiments. For each transform, the seismic data was compressed for different compression ratios. The DFT and WHT have higher error compared to the KLT and DCT. Although the KLT has achieved the smallest error, it is data-dependent and cannot generalize to multiple volumes. DCT has the best overall performance in terms of robustness and compression ratios.

However, the DCT can lead to blocking effects at high compression rates, a natural consequence of the individual processing of each block. To deal with block artifacts, a new method called Local Cosine Transform (LCT) was proposed by Aharoni et al. (1993) to improve compression. The idea is to apply smooth cutoff functions as a pre-processing step before performing the DCT such that the data becomes more suitable to be compressed.

A systematical evaluation of compression methods was performed by Averbuch et al. (2001). They evaluated eight combinations of methods, concerning the transform, quantization and entropy coding steps. Considering the transform, they evaluated the Fast Wavelet Transform (FWT) and the LCT. Due to its computational cost, the wavelet transform offers some advantages over other transforms while the LTC is superior in terms of signal-noise ratio.

An optimized sub-band coding is proposed by Røsten, Ramstad e Amundsen (2004). In this work, the input signal is decomposed into a set of sub-bands by a filter bank. These sub-bands are evaluated so that the energy of the input signal is concentrated/focused into a small number of sub-bands. They are downsampled, and a quantizer is used to allow compression using the entropy coding. The decoder performs the inverse, aiming to reconstruct the original signal.

An approach based on dimensionality reduction is proposed by Nuha et al. (2017). In their work, the trace dimensions are reduced using a Principal Component Analysis (PCA) method. They considered the similarity between traces to provide higher compression rates so that a trace can be approximated by shifting others.

Recently, some methods explored standard video and image compression techniques.

Liu et al. (2016) proposed to use the Joint Photographic Experts Group Extended Range (JPEG XR) standard to compress seismic sections represented as 32-bit floating-points. To this end, they convert the input to a 32-bit integer representation and compress the resulting image using the JPEG XR. The decoder turns back the representation to floating-point representation. They also proposed to control the Signal-to-Noise Ratio (SNR) and rate achieved by the method. This is achieved by adjusting parameters related to the quantization to achieve the SNR or bit rate target.

Radosavljević et al. (2016) proposed a method that extends the High-Efficiency Video Coding (HEVC) to compress seismic images. Almost all components of the HEVC codec (SULLIVAN et al., 2012) were redesigned. They used a lifting-based BinDCT instead of the DCT to achieve better reconstruction. This method was developed to compress 2D seismic data, and an extension was proposed by Radosavljević et al. (2017) aiming to explore the redundancy in three-dimensional data. The idea is to consider the 3D seismic data as a sequence of frames similar to videos. They used a block matching scheme, in which the motion estimation is computed over the volume. Parameters related to the quantizer were selected by using Lagrange multipliers.

1.4.2 Deep Learning for Image and Video Compression

Recently, Convolutional Neural Networks (CNN) have achieved impressive results in many Computer Vision tasks. Among the benefits of the use of CNN's to solve Computer Vision tasks, we can highlight their generalization power and ability to learn features from high amounts of data with little to no human efforts. One of the first attempts to use CNN's to image compression was performed by Ballé, Laparra e Simoncelli (2017). They proposed a method composed of analysis and synthesis modules, in which the analysis is referred to the encoder and the synthesis to the decoder parts of a convolutional autoencoder. The synthesis transform is the inverse of the analysis transform. These modules are composed of convolutional, downsampling/upsampling and Generalized Divisive Normalization (GDN) (BALLÉ; LAPARRA; SIMONCELLI, 2016) layers. The GDN is used to replace the batch normalization, since GDN can deal with local joint statistics of natural images, being spatially adaptive. To deal with the non-differentiability from quantization, during training they added uniform noise to the output of the encoder allow the network to be optimized using the Stochastic Gradient Descent (SGD) optimizer. For testing they used rounding. Their method outperforms the Joint Photographic Experts Group (JPEG) and Joint Photographic Experts Group 2000 (JPEG2000) codecs in terms of Peak Signal-to-Noise Ratio (PSNR) and Multi-Scale Structural Similarity (MS-SSIM) metrics.

The previous method was extended using a scale hyperprior to improve the entropy estimation (BALLÉ et al., 2018). They use a Gaussian scale mixture (GSM) as an entropy model, where the scale parameters are conditioned to a hyperprior. The idea is to send information concerning the image encoded from the encoder to decoder parts. The decoder first recovers the scale information and the latent representation, and it is used to reconstruct the input image. In this sense, the entropy model depends on the input image, allowing reduced bit rates. This method achieves comparable results to the state of the art in terms of PSNR and MS-SSIM.

Toderici et al. (2017) proposed a scheme based on Recurrent Neural Networks (RNN) for the variable rate compression on images without needing to retrain the network. The idea is to use RNN models to enable a progressive compression of an image, yielding variable bit rates. The input is encoded, and its latent representation is converted into a bitstream. The bitstream is encoded using an architecture similar to the one proposed by Oord, Kalchbrenner e Kavukcuoglu (2016). Their idea is to model the joint distribution over a pixel of an image using a CNN. The decoder uses it to reconstruct the image. This process is repeated using the residual between original and reconstructed images. The compression rate is determined by the number of iterations of the network, in which the compression of the residual is performed in different contexts.

Theis et al. (2017) proposed a compressive autoencoder system that uses mainly convolutional layers. To deal with quantization, they replaced the derivatives of the rounding function so that the quantization is performed as usual in the forward pass, and in the backward pass, a smooth approximation is used. They also used scale parameters to deal with variable bit rates without needing to retrain the model. The bit rate estimation is performed by using GSM with 6 scales in the training. The entropy is estimated by using Laplace-smoothed histograms. This method achieved similar results to the JPEG in terms of PSNR, MS-SSIM and Structural Similarity (SSIM).

Taking into account that the visual information in images is spatially variant, Li et al. (2018) proposed a model that weighs regions according to the detail level. High-frequency regions tend to demand more bits to be encoded than low-frequency. An importance map network is used to generate an importance map in which the bit rate allocation becomes locally adaptive. The latent representation was used to generate a mask used to weight the encoded information. This method is extended by Li et al. (2019), a Trimmed Convolutional Network (TCNN) is used to achieve low bit rates, predicting the current symbols from the previous available. An inclined TCNN is also proposed to partition the quantized latent representation into inclined planes to perform the decoding in parallel. This method outperforms state of the art results in terms of PSNR and achieves good results for MS-SSIM.

One of the first attempts to use Generative Adversarial Networks (GAN) to improve image compression was performed by Rippel e Bourdev (2017). They used an adversarial loss to enhance the visual quality of the reconstructed image. In this setting, the encoder-decoder is the generator, and the discriminator learn to differentiate the original and reconstructed images. The feature extraction is based on pyramidal structured CNN's, to learn scale-dependent information. The latent representation is quantized using a bit plane decomposition to achieve further compression. Complementing an autoencoder with adversarial training, the proposal of Rippel e Bourdev (2017) to image compression outperforms JPEG, JPEG2000, Better Portable Graphics (BPG) and WebP codecs in terms of MS-SSIM metric, producing visually agreeable reconstructions for very low bit rates in real-time.

With competitive performance to state of the art results, Mentzer et al. (2018) proposed an image compression system based on two networks trained concurrently. They proposed an autoencoder inspired in the work of Theis et al. (2017) to deal with the trade-off between entropy and distortion. The entropy is estimated by using a probabilistic model that combines the proposals (OORD et al., 2016; OORD; KALCHBRENNER; KAVUKCUOGLU, 2016). They extended the work of Li et al. (2018) to generate an importance map without needing an extra network, by adding a channel in the last layer of the encoder. The quantization is performed similarly to the approach of Agustsson et al. (2017) but using scalars instead.

Habibian et al. (2019) presented a video compression method based on Variational Autoencoders (VAE). They argue that a stochastic encoder is not beneficial for compression since any noise added increases the bitrate. The main architecture is based on the works of Mentzer et al. (2018) and Ballé et al. (2018), using 2D and 3D autoencoders with a temporarily-gated PixelCNN as prior. The autoencoder used in this work is closely related to the proposed by us. Besides, extensions to semantic compression and adaptive compression are presented. Experiments were performed by evaluating 2D and 3D autoencoders and comparisons with state of the art methods shown that the method performs on par with the codec H.265/HEVC.

1.5 OUTLINE

The remainder of this work is organized as follows. The Chapter 2 presents a description of the main concepts needed for the comprehension of this work. Chapter 3 describes the proposed methods of this work that are evaluated in the Chapters 4 and 4.6. Chapter 5 presents the conclusion and future works.

2 FUNDAMENTALS

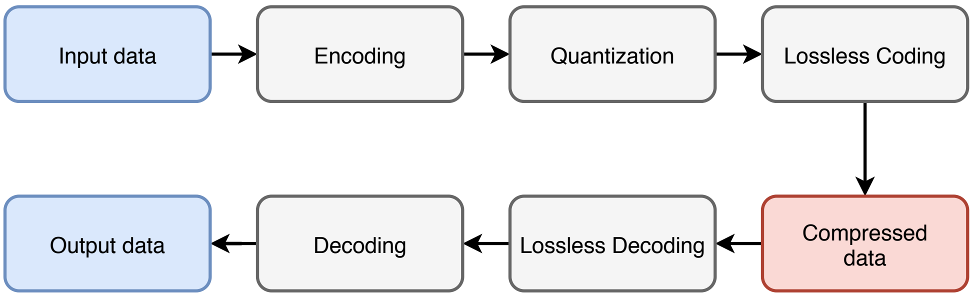
This chapter introduces the fundamentals needed for the comprehension of our work. We present the theory about data compression and the modules that compose it. We also explain the concepts about deep learning for data compression and the main structures that constitute the networks used: autoencoder, convolutional neural network (CNN) and residual blocks. Secondary structures such as batch normalization, softmax and activation functions are not explained since they are not the focus of our work. Moreover, it is presented a description of the work of Mentzer et al. (2018), on which we are based.

2.1 DATA COMPRESSION

According to Gonzalez e Wintz (1987), a compression system consists of two main blocks: an encoder and a decoder. Given a m-dimensional input signal $\mathbf{x} \in \mathbb{R}^m$, the encoder $\mathbf{E} : \mathbb{R}^m \to \mathbb{R}^n$, is a transformation that maps $\mathbf{x}$ to an encoded representation $\mathbf{z}$, suitable for compression. The decoder $\mathbf{D} : \mathbb{R}^n \to \mathbb{R}^m$ is a transformation that maps back $\mathbf{z}$ to the input space, yielding $\hat{\mathbf{x}}$ as output. If the reconstruction $\hat{\mathbf{x}}$ is equal to the input $\mathbf{x}$, the compression is called *lossless*. However, if the reconstruction $\hat{\mathbf{x}}$ approximates the input, the compression is called *lossy* and a quantizer block can be added between the encoder and the decoder to allow higher compression rates. In this work, we are focusing on lossy compression.

Figure 2 depicts a flowchart for a lossy compression scheme. Given the input data, the encoding is performed to produce a low-dimensional representation. The quantization discretizes the coordinates of the encoded representation and a lossless coding strategy is used to generate a bitstream. The bitstream is lossless decoded and the input is reconstructed.

Figure 2 – Lossy compression flowchart.



Source: created by the author.

Since $\mathbf{z} \in \mathbb{R}^n$, infinite elements are needed to represent it with the highest precision. As it is not possible, the quantizer $\mathbf{Q} : \mathbb{R}^n \to \mathcal{C}^{n/v}$ must solve the problem of representing a continuous variable using a limited number of elements. The goal is to find a finite set $\mathcal{C} = \{c_1, \ldots, c_L\}$, $c_l \in \mathbb{R}^v$ of quantization cells that accurately represent $\mathbf{z}$. If $v = 1$, the quantization is called *scalar*, where each quantization cell c_l corresponds to a scalar value. The quantization is called *vectorial* if the quantization cells have vectorial shape, with $v > 1$. For lossy compression with a quantizer block, the decoder must be redefined, becoming $\mathbf{D} : \mathcal{C}^{n/v} \to \mathbb{R}^m$ and yielding the quantized latent representation $\hat{\mathbf{z}}$ as output. As quantization is irreversible, some distortions can appear on the reconstructed side.

The objective of lossy compression systems is to represent information with fewer bits than its original form. With this goal, it is necessary the use of mathematical tools to measure the compression capacity of a system. From the rate-distortion theory (SHANNON, 1959), the distortion $\mathbf{d}$ between the input $\mathbf{x}$ and output $\hat{\mathbf{x}}$ and the bit rate (entropy) $\mathbf{r}$ of the encoded information $\hat{\mathbf{z}}$ can jointly be evaluated through the trade-off:

$$\mathbf{d}(\mathbf{x}, \hat{\mathbf{x}}) + \beta \mathbf{r}(\hat{\mathbf{z}}). \tag{2.1}$$

At the same time, small distortions and rate are desirable such that the β controls the trade-off between them.

The distortion $\mathbf{d}(\mathbf{x}, \hat{\mathbf{x}})$ can be defined as the cost of representing $\mathbf{x}$ using $\hat{\mathbf{x}}$. It can be measured by some metrics such as PSNR or MS-SSIM. Choosing the appropriate metric is of great importance as it quantifies the type of information that is preserved. A perceptual metric, such as MS-SSIM, measures the amount of lost information that the human visual system can capture. Perceptual metrics do not consider the differences between pixel values from original and reconstructed images, but only their visual similarities. In contrast, a pixel-wise metric, such as PSNR, measures the distortion between the original and reconstructed pixel values. Although it can not detect the presence of some visual artifacts, most of the visual information is preserved for high values of PSNR (> 40 dB). Since we are interested in compressing the seismic data preserving both its qualitative and quantitative aspects as much as possible, we chose to use the PSNR metric in this work. The PSNR (in decibels) between $\mathbf{x}$ and $\hat{\mathbf{x}}$ is given by:

$$\mathbf{d}(\mathbf{x}, \hat{\mathbf{x}}) = \mathrm{PSNR}(\mathbf{x}, \hat{\mathbf{x}}) = 10 \cdot \log_{10} \frac{\mathrm{MAX}^2}{\mathrm{MSE}(\mathbf{x}, \hat{\mathbf{x}})}, \tag{2.2}$$

where MAX is the maximum value possible and MSE is the mean squares error associated to $\mathbf{x}$ and $\hat{\mathbf{x}}$. If $\mathbf{x}$ is normalized in the range [0,1], the MAX value used is 1.

The bit rate $\mathbf{r}(\hat{\mathbf{z}})$ of the trade-off is related to the coding cost (or entropy) of the compressed representation $\hat{\mathbf{z}}$. From Cover e Thomas (2006), the entropy of a random variable is defined as a lower bound on the average number of bits required to represent it. Considering $\hat{\mathbf{z}}$ as a sequence of discrete random variables $\hat{\mathbf{z}} = \hat{\mathbf{z}}_1, \ldots, \hat{\mathbf{z}}_n$ and a joint probability mass function $p(\hat{z}_1, \ldots, \hat{z}_n)$, the entropy of $\hat{\mathbf{z}}$ is given by:

$$\mathbf{r}(\hat{\mathbf{z}}) = r(\hat{\mathbf{z}}_1, \ldots, \hat{\mathbf{z}}_n) = -\sum_{\hat{z}_1, \ldots, \hat{z}_n} p(\hat{z}_1, \ldots, \hat{z}_n) \log p(\hat{z}_1, \ldots, \hat{z}_n). \tag{2.3}$$

If the log has base 2, the entropy expresses the number of bits, on average, required to describe $\hat{\mathbf{z}}$.

2.2 DEEP LEARNING

Deep learning is a Machine Learning subfield that aims to learn approximations of functions by sequentially stacking many layers in a neural network. A network with this characteristic is known as a Deep Neural Network (DNN). Its recent success can be assigned to a set of factors, including the availability of massive datasets and the efficient GPU computing hardware.

According to Hornik, Stinchcombe e White (1989), neural networks with a single hidden layer and sigmoidal activations can approximate any function to any degree of accuracy once sufficient many hidden units are available. However, in many applications, the performance of deep architectures outperforms the shallow ones. Compared to shallow, deeper architectures better capture invariant properties of the data, allowing to extract high order features (MHASKAR; POGGIO, 2016).

The following subsections describe the main structures that compose our networks: autoencoder, CNN and residual block. The autoencoder defines the architecture used to solve data compression tasks. Convolutional networks are useful to capture local patterns that are not possible using the fully connected ones. The residual block allows us to construct deeper architectures. Jointly, these structures enable us to develop a deep compression system for post-stack data.

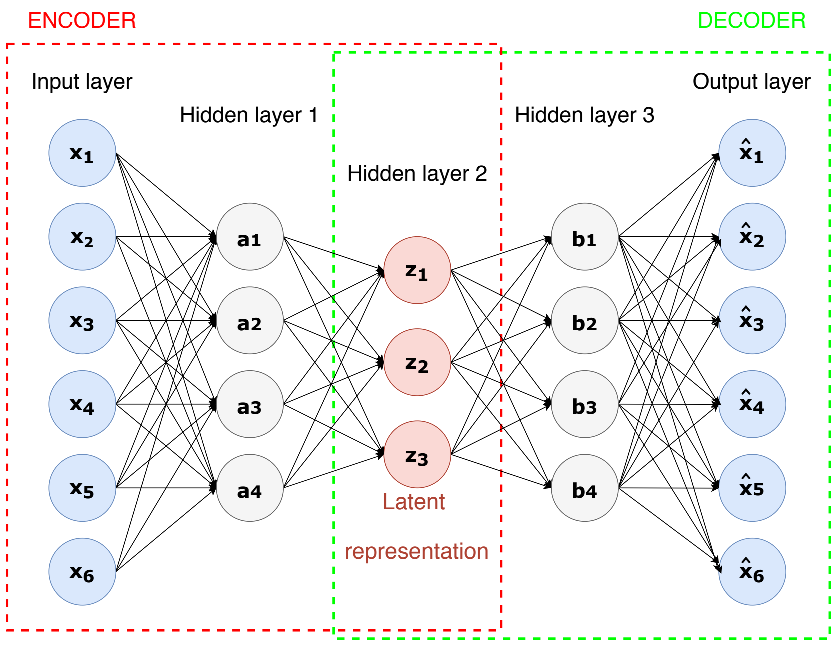
2.2.1 **Autoencoder**

Autoencoder is the most straightforward way to perform data compression using artificial neural networks. They are a type of neural network trained to approximate the identity function (BALLARD, 1987), developed to solve encoder problems (ACKLEY; HINTON; SEJNOWSKI, 1985) in which data compression is an example. Similarly to a compression system, autoencoders are composed of encoder and decoder modules.

The autoencoder has an input layer as the encoder and an output layer as the decoder, where both have the same number of neurons. The hidden layer yields the latent representation, also known as the bottleneck. This architecture can solve simple problems, but for challenging tasks, as image compression, a deep autoencoder is more suitable. Figure 3 shows a deep autoencoder architecture. It has the same base structure, but with more than one hidden layer. In this case, the latent representation corresponds to the layer with the smallest dimension.

The deep compressive autoencoder (THEIS et al., 2017) is a variant that has an additional module aiming to perform quantization. This structure mimics a lossy

Figure 3 – Deep autoencoder architecture. It is composed by encoder and decoder modules. The *latent representation* is often the layer with the smallest dimension.



Source: created by the author.

compression system in which its three main steps can be represented by the network. The encoder **E**, the decoder **D**, and the quantizer **Q** are constituted of layers of a neural network. The loss function $\pounds$, based on the rate-distortion trade-off, is given by:

$$\pounds = \mathbf{d}(\mathbf{x}, \mathbf{D}(\mathbf{Q}(\mathbf{E}(\mathbf{x})))) + \beta\mathbf{r}(\mathbf{Q}(\mathbf{E}(\mathbf{x}))), \tag{2.4}$$

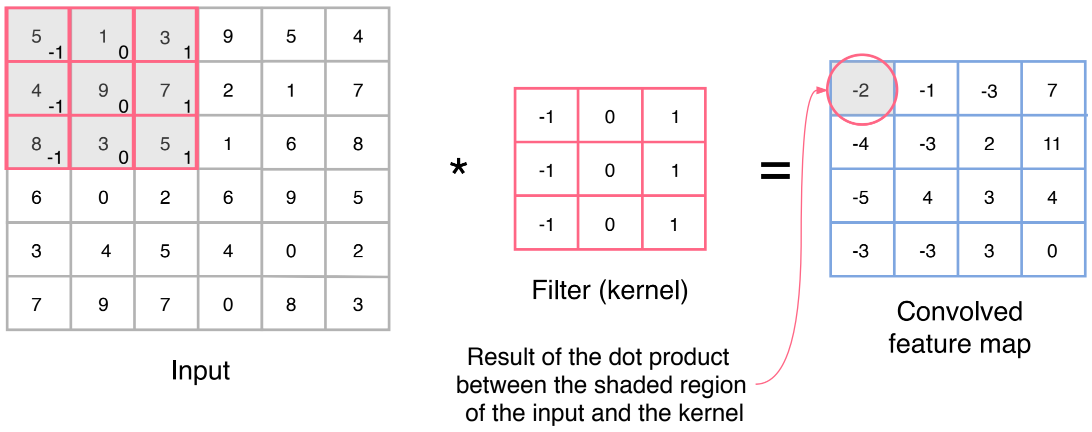
where $\mathbf{x}$ is the network input. The learning is based on the backpropagation algorithm and all modules must be differentiable. Since the quantization step is not differentiable and could not be represented by a neural network, an approximated soft quantization function is often used. For simplicity, in the remainder of this work, we will refer to a deep compressive autoencoder as a deep autoencoder.

2.2.2 Convolutional Neural Network

Convolutional networks (LECUN et al., 1990) are feed-forward neural networks in which their main layers perform the convolution operation. They have proven to be powerful tools for solving Computer Vision tasks such as image classification (KRIZHEVSKY; SUTSKEVER; HINTON, 2012) and image compression (TODERICI et al., 2017). Different from the fully-connected layer, convolutional layers can capture spatial dependencies in visual data.

Figure 4 illustrates a convolutional layer. Given an image, a filter (also called kernel) is applied over its pixels. The output is known as a feature map, is the result of the application of the filter on the image. The filter slides in the raster scan order with a given stride value until it reaches the last pixel of the image. Considering the central position of the filter as the reference, at each location, it is computed the dot product between the kernel and the input, and the result is attached to the output current position. Each layer can be composed of multiples filters, and this process is performed for all of them.

Figure 4 – Example of a convolution operation. An input is convolved with a kernel, yielding a feature map as output.



Input
Filter (kernel)
Convolved feature map
Result of the dot product between the shaded region of the input and the kernel

Source: created by the author.

The amount by which the filter shifts is the stride. The stride controls how the filter convolves around the input data. The stride can be useful for dimensionality reduction purposes since the output feature map dimensions are downscaled by a factor equals to the stride value. By default, applying a filter to an input yields a feature map with smaller spatial dimensions than the input. It occurs because the border values are always exposed to the filter edges. However, producing a feature map with the same dimension of the input is often desirable. To this end, padding strategies can be applied in which an extra border is added to the input. Although stride and padding definitions were made separately, it is important to emphasize that the feature map dimensions consider both at the same time. Thus, a combination of them can lead to different behavior from those previously described.

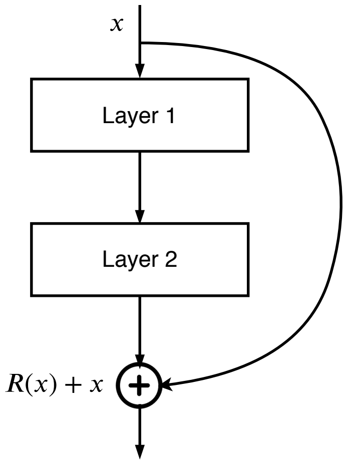
Transposed convolutions can be used to output feature maps larger than the input. The transposed convolution is a transformation that goes in the opposite direction of the conventional, equivalent to the gradient calculation for a regular convolution. It is useful for increasing the output feature map dimensions, where an upsampling function can be learned by the layer. A transposed convolution can be implemented as a regular convolution in which the padding is applied aiming to make the feature map dimensions greater than the input.

Our models are based on deep compressive autoencoders. As the seismic data is spatially correlated, convolutional layers are employed in the encoder part, by downsampling the input to generate a representation suitable for compression. The decoder performs transposed convolutions to reconstruct the input, learning the upsampling counterpart.

2.2.3 **Residual Block**

Training deeper networks is a hard task since they are exposed to the degradation problem, where as the network depth increases, the accuracy becomes saturated. To address this problem, He et al. (2016) proposed a residual learning framework. Instead of making the layers learn a function, they fit the residual between input and output. They verified the hypothesis that it is easier to optimize the residual mapping $R(x)$ than to optimize the original mapping. Besides that, as the number of layers increases, the gradient vanishing problem becomes critical. Residual blocks can be used to attenuate this problem, making possible to train deeper networks. Figure 5 depicts a residual block structure. The input x feeds the next layer and is also merged with the output of the layer about some hops away.

Figure 5 – Residual block structure. The input feeds the next layer and is also merged with the output of the layer about some hops away.



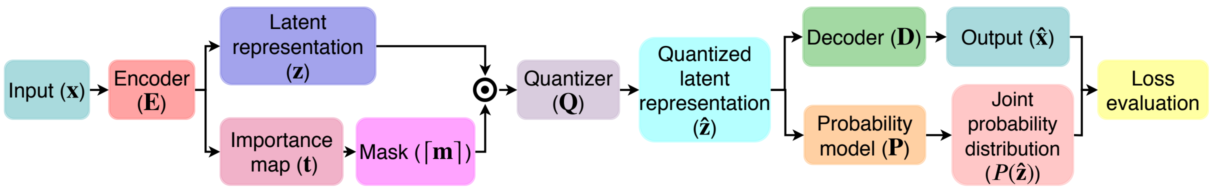
Source: created by the author.

2.3 CONDITIONAL PROBABILITY MODELS FOR DEEP IMAGE COMPRESSION

This section describes the method that is the basis of our work. With competitive performance to the state of the art results, Mentzer et al. (2018) proposed a general-purpose image compression approach based on two networks trained concurrently. A probabilistic model is used to learn the dependencies between symbols in the autoencoder latent representation, and an autoencoder uses it for entropy estimation to control the rate-distortion trade-off. Figure 6 depicts the steps executed in the image compression scheme proposed by Mentzer et al. (2018) that is also performed in our seismic compression approach. The input is encoded, it outputs a latent representation and an importance map.

The latent representation is masked with a mask generated through the importance map. The masked latent representation is quantized and used by the decoder to reconstruct the input. The probability model uses the quantized latent representation to estimate the bit rate used to compress the data. In the end, the metric is evaluated, yielding the quality and bit rate measures reached in the compression.

Figure 6 – Flowchart of the image compression scheme proposed by Mentzer et al. (2018) that is also performed in our seismic compression approach.

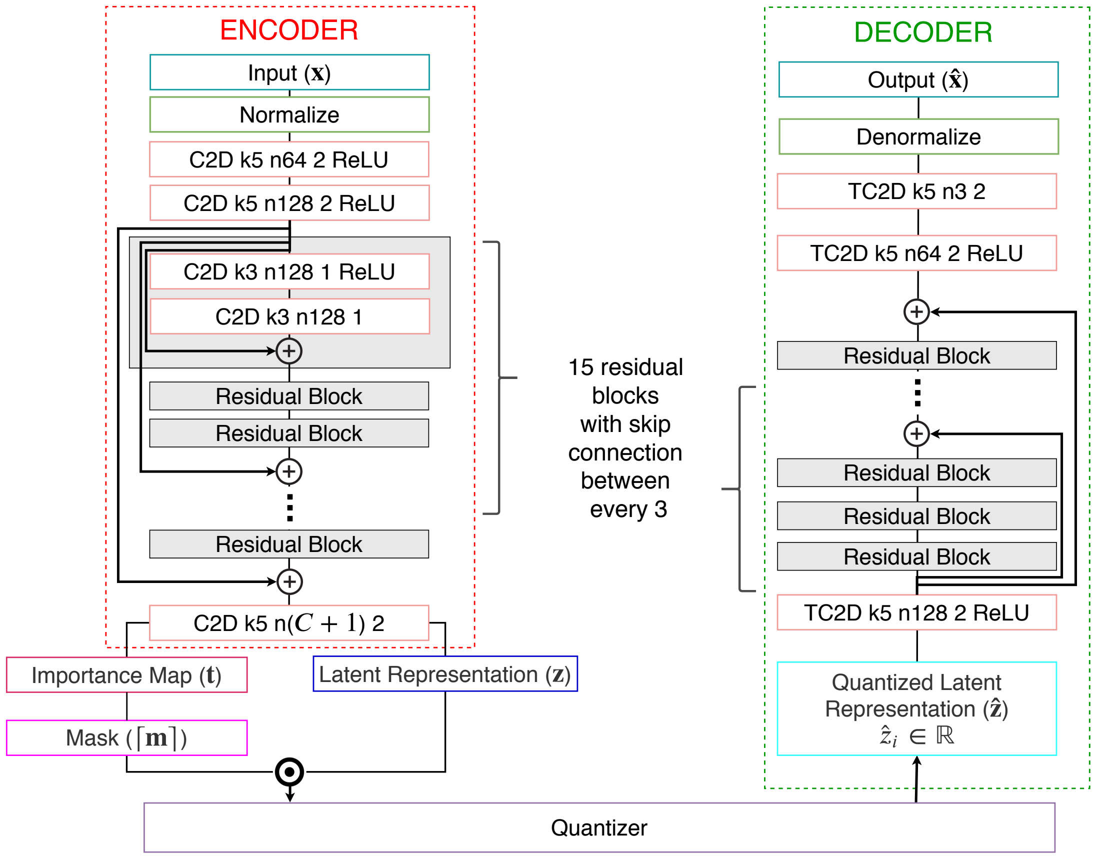

Input ($\mathbf{x}$)
Encoder ($\mathbf{E}$)
Latent representation ($\mathbf{z}$)
Importance map ($\mathbf{t}$)
Mask ($\lceil \mathbf{m} \rceil$)
$\odot$
Quantizer ($\mathbf{Q}$)
Quantized latent representation ($\hat{\mathbf{z}}$)
Decoder ($\mathbf{D}$)
Output ($\hat{\mathbf{x}}$)
Probability model ($\mathbf{P}$)
Joint probability distribution ($P(\hat{\mathbf{z}})$)
Loss evaluation

Source: created by the author.

The encoder $\mathbf{E} : \mathbb{N}^{h \times w \times 3} \rightarrow \mathbb{R}^{h/8 \times w/8 \times C}$ maps an input image $\mathbf{x}$ with spatial dimensions $h \times w$ and 3 color channels to a latent representation $\mathbf{z} = \mathbf{E}(\mathbf{x})$, where the value 8 corresponds to the network subsampling factor for spatial dimensions w and h and the parameter C indicates the number of feature maps of the latent representation. In this sense, the input volume of dimensions $h \times w \times 3$ is mapped into a latent representation volume of dimensions $h/8 \times w/8 \times C$. 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 set of centroids $\mathcal{C} = \{c_1, \ldots, c_L\}, c_l \in \mathbb{R}$, yielding $\hat{\mathbf{z}} = \mathbf{Q}(\mathbf{z})$. The decoder $\mathbf{D} : \mathcal{C}^{h/8 \times w/8 \times C} \rightarrow \mathbb{N}^{h \times w \times 3}$ then forms the reconstructed image $\hat{\mathbf{x}} = \mathbf{D}(\hat{\mathbf{z}})$.

Figure 7 shows the autoencoder architecture proposed by Mentzer et al. (2018) and that is the basis of our methods. 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" represents the convolution 2D with kernel size 5, 64 filters and stride 2 for both spatial dimensions w and h. To reduce the input spatial dimensions, the encoder performs two convolutions with stride 2 followed by 15 residual blocks with skip connection between every 3. 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 $\mathbf{t} \in \mathbb{R}^{h/8 \times w/8 \times 1}$. It is important to notice that the number of feature maps in the latent representation directly implies the bit rate obtained in practice. Small values of C lead to low bit rates, and vice versa. Then $\mathbf{z}$ is masked and quantized and its quantized representation $\hat{\mathbf{z}}$ is used to feed the decoder side. 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 activation function used is ReLU (NAIR; HINTON, 2010).

Figure 7 – Detailed encoder and decoder architectures proposed by Mentzer et al. (2018) and used in our methods.



Source: created by the author.

Considering that the visual information is spatially variant in an image, an importance map can be used to spatially reach different regions of the image with different bit rates. For example, a blue sky can require fewer bits to be reconstructed than a person, and the remaining bits can be allocated to more detailed regions. Instead of using a neural network to generate the importance map, Mentzer et al. (2018) adapted the idea from Li et al. (2018) by adding a single-channel output $\mathbf{t} \in \mathbb{R}^{h/8 \times w/8 \times 1}$ at the last layer of the encoder, as illustrated in the Figure 8. The importance map learns to weight the regions that provide the most relevant information for the reconstruction of the image. The importance map $\mathbf{t}$ is expanded into a mask $\mathbf{m} \in \mathbb{R}^{h/8 \times w/8 \times C}$ as follows:

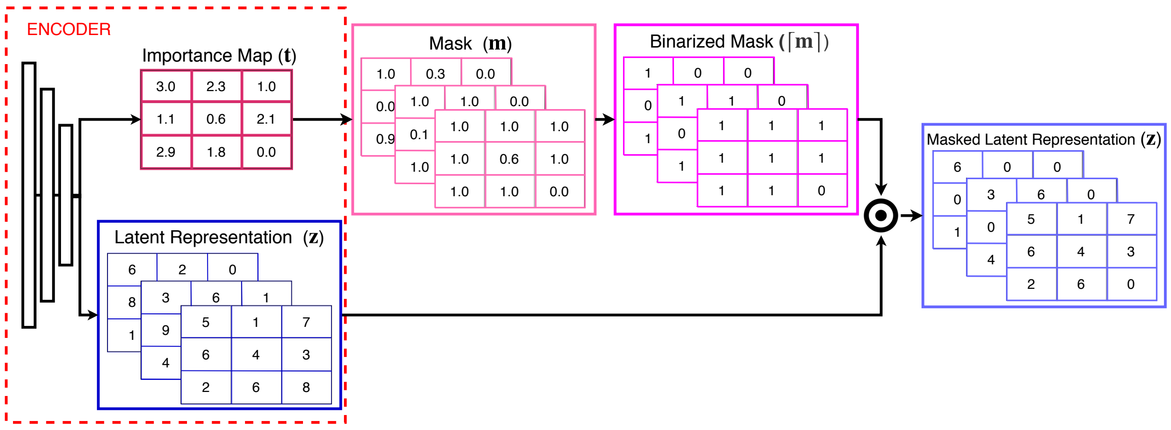
$$m_{w',h',c'} = \begin{cases} 1 & \text{if } c' < t_{w',h'} \\ (t_{w',h'} - c') & \text{if } c' \leq t_{w',h'} \leq c' + 1. \\ 0 & \text{if } c' + 1 > t_{w',h'} \end{cases} \tag{2.5}$$

At this point, $\mathbf{m}$ has values in the interval [0,1]. Since the goal is to produce zero entries in the latent representation, the mask is binarized and $\mathbf{z}$ is masked by a point-wise multiplication with $\lceil \mathbf{m} \rceil$:

$$\mathbf{z} \leftarrow \mathbf{z} \odot \lceil \mathbf{m} \rceil \, . \tag{2.6}$$

The ceiling operator $\lceil . \rceil$ is not differentiable, and the identity is used for the backward pass.

Figure 8 – Example of masking performed between the latent representation and the mask generated using the importance map.


ENCODER
Importance Map ($\mathbf{t}$)
Latent Representation ($\mathbf{z}$)
Mask ($\mathbf{m}$)
Binarized Mask ($\lceil \mathbf{m} \rceil$)
Masked Latent Representation ($\mathbf{z}$)
$\odot$

Source: created by the author.

Although some latent representation entries become zero, $\mathbf{z}$ still needs an infinite number of elements to be represented and the quantizer needs to discretize $\mathbf{z}$. Inspired by Agustsson et al. (2017), the quantization step considers that the quantizer is composed of a finite set $\mathcal{C} \subset \mathbb{R}$ of scalar centroids learned by the autoencoder. The latent representation entries $\hat{z}_i$ are assigned to the nearest centroid as follows:

$$\hat{z}_i = \arg\,\min_{c_l \in \mathcal{C}} \|z_i - c_l\| \,. \tag{2.7}$$

Since arg min is not differentiable, the soft quantization:

$$\tilde{z}_i = \sum_{l=1}^{L} \frac{\exp(-\sigma \|z_i - c_l\|)}{\sum_{j=1}^{L} \exp(-\sigma \|z_i - c_j\|)} c_l = \sum_{l=1}^{L} \mathrm{softmax}(-\sigma \|z_i - c_l\|) c_l, \tag{2.8}$$

is used in the backward pass of the backpropagation. The parameter σ controls the approximation level between $\tilde{z}_i$ and $\hat{z}_i$.

According to Equation (2.3), it is needed to know the distribution of the latent representation. A model that combines the proposals (OORD; KALCHBRENNER; KAVUKCUOGLU, 2016; OORD et al., 2016) was proposed to compute the bit rate used to compress the autoencoder latent representation. 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 latent representation can be stacked into a 3D volume in which the third dimension is related to the number of channels of the latent space. Let be $\hat{\mathbf{z}}$ indexed in the raster scan order (row by column by depth). The joint distribution $p(\hat{\mathbf{z}})$ can be represented as a product of conditional distributions:

$$p(\hat{\mathbf{z}}) = p(\hat{z}_1, \ldots, \hat{z}_n) = \prod_{i=1}^{n} p(\hat{z}_i | \hat{z}_{i-1}, \ldots, \hat{z}_1), \tag{2.9}$$

where $n = w/8 \cdot h/8 \cdot C$.

The problem of learning a conditional distribution can be formulated similarly 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 a match $\hat{z}_i$ 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 $\hat{z}_i$ to each one of the L centroids given the previous values.

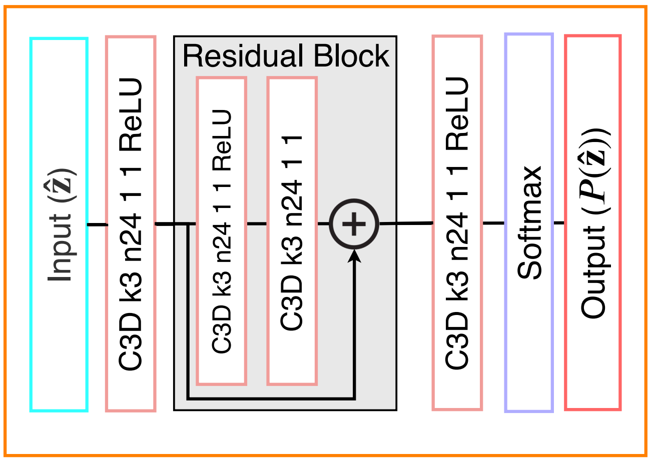
Although the true entropy $p(\hat{\mathbf{z}})$ is unknown, the labels $\hat{\mathbf{y}}$ can be calculated through $\hat{\mathbf{z}}$ and $\mathcal{C}$. The labels $\hat{\mathbf{y}}$ have a one-hot representation for each entry of $\hat{\mathbf{z}}$ in which the value 1 is in the index of its nearest centroid. It can be viewed as a distribution that has no associated uncertainty and zero entropy. However, the labels can be used to make the model $\mathbf{P}$ learn a distribution that approximates the desired one through a cross-entropy loss. The cross-entropy yields the average number of bits needed to encode information using an approximated distribution instead of the true.

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

$$P_{i,l}(\hat{\mathbf{z}}) \approx p(\hat{z}_i = c_l|\hat{z}_{i-1}, \ldots, \hat{z}_1), \tag{2.10}$$

where $P_{i,l}$ is the probability of each symbol $\hat{z}_i$ to be assigned to each centroid c_l of $\mathcal{C}$. Figure 9 shows the architecture of the probabilistic model. As in the autoencoder architecture, "C3D k3 n24 1" indicates the convolution 3D with kernel 3 and 24 filters and stride 1 for spatial and depth dimensions. At the last convolutional layer, the number of filters used is the size of the centroids set. The softmax outputs a distribution for each voxel of the input volume.

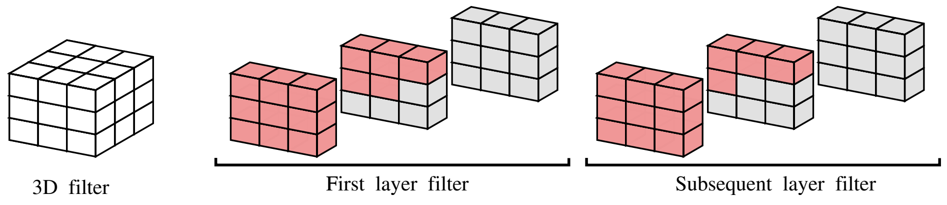
Figure 9 – Detailed Probabilistic Model architecture, also proposed by Mentzer et al. (2018) and used in our methods.



Source: created by the author.

Since each distribution $p(\hat{z}_i|\hat{z}_{i-1}, \ldots, \hat{z}_1)$ depends on the previous values, the network needs to deal with the conditioning. It can be guaranteed by using specific filters in the convolutional layers. Instead of considering all previous values, the conditioning takes into account only a region around the current voxel. The region is defined by the filter of the convolution. Mentzer et al. (2018) proposed to use a filter in the first layer of the network and another in the remaining ones. Figure 10 shows the 3D filter of dimensions $3 \times 3 \times 3$ for the first (left) and subsequent (right) layers.

Figure 10 – Example of 3D filter of dimensions $3 \times 3 \times 3$ for the first and subsequent layers. Red voxels have the value 1 and the gray ones have the value 0.



Source: created by the author.

The model $\mathbf{P}$ is trained with a loss:

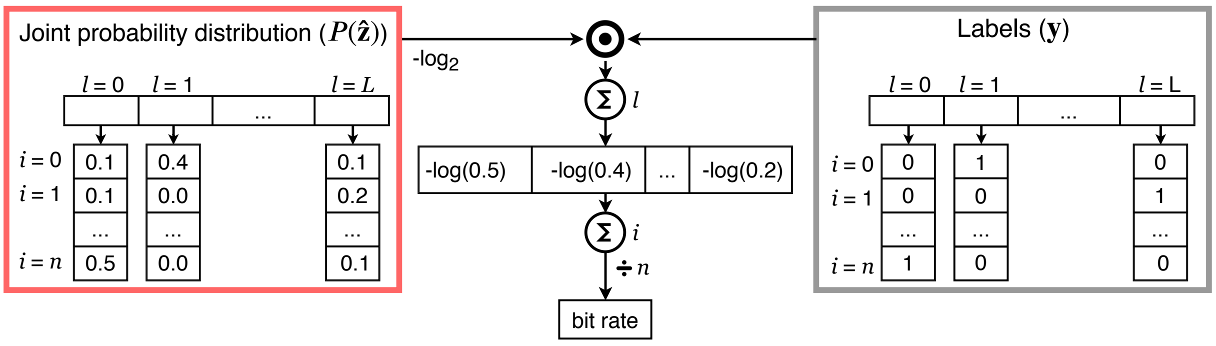
$$\mathcal{L}_P = \mathbf{d}(\mathbf{x}, \hat{\mathbf{x}}) + \beta \mathbf{C}(\hat{\mathbf{z}}), \tag{2.11}$$

where:

$$\mathbf{C}(\hat{\mathbf{z}}) := -\frac{1}{n} \sum_{i=1}^{n} \sum_{l=1}^{L} \mathbf{y}_{i,l} \log P_{i,l}(\hat{\mathbf{z}}), \tag{2.12}$$

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. Figure 11 illustrates the bit rate estimation. Given the labels set $\mathbf{y}$ and the estimated distribution $P(\hat{\mathbf{z}})$, the cross-entropy is calculated according to Equation 2.12. Notice that the cross-entropy does not depend on the values of $\hat{\mathbf{z}}$, but only on the probabilities.

Figure 11 – Bit rate estimation.



Source: created by the author.

The autoencoder uses the probability model loss to deal with the rate-distortion trade-off. To control the bit rate obtained, the autoencoder weights $P(\hat{\mathbf{z}})$ 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:

$$\mathcal{L}_{\mathbf{E},\mathbf{D},\mathbf{Q}} = \mathbf{d}(\mathbf{x}_j, \hat{\mathbf{x}}_j) + \beta \mathbf{MC}(\hat{\mathbf{z}}) + R_{AE} \cdot \ell_2(\mathbf{W}_{AE}) + R_{\mathcal{C}} \cdot \ell_2(\mathcal{C}), \tag{2.13}$$

where:

$$\mathbf{MC}(\hat{\mathbf{z}}) := -\frac{1}{n} \sum_{i=1}^{n} \sum_{l=1}^{L} \lceil m_i \rceil \, \mathbf{y}_{i,l} \log P_{i,l}(\hat{\mathbf{z}}), \tag{2.14}$$

ℓ_2 is the regularization loss, the parameter R_{AE} is the autoencoder regularization factor, $\mathbf{W}_{AE}$ is the autoencoder weights matrix and $R_{\mathcal{C}}$ is the centroids regularization factor. It is not added regularization for the probabilistic model. Regularization is used to prevent model overfitting. The MS-SSIM was used as a distortion function for both losses in Equations 2.11 and 2.13. The perceptual metric ensures that the output images are visually agreeable, reducing block artifacts and blur.

3 PROPOSED METHODS

The main objective of this work is to compress 3D post-stack seismic data using deep learning models. To this, we need to learn the functions expressed in Equations (1.1), (1.2) and (1.3) that allow the compression of the seismic data. Since neural networks are proven to be able to approximate functions, we intend to learn a deep autoencoder model that mimics a compression system. An encoder, quantizer and decoder are used to approximate Equations (1.1), (1.2) and (1.3), respectively, by minimizing the rate-distortion trade-off (Equation 2.1). By extending the work of Mentzer et al. (2018), we propose four methods to compress the seismic data:

1. **2D-based Seismic Data Compression (2DSC)**: described in the Subsection 3.1.1, it is considered as our baseline method, it is equivalent to the straightforward application of the method proposed by Mentzer et al. (2018) to compress the seismic data. The main difference is concerning to the data, in which instead of 3-channels of 8-bit unsigned integers, we compress slices represented as 1-channel 32-bit floating-points.

2. **2D-based Seismic Data Compression using Multi-resolution (2DSC-MR)**: described in Subsection 3.1.2, this method aims to compress seismic sections through an analysis of the data in multiple resolutions. The original input is downscaled by a factor of 2 and the result is downscaled again by a factor of 2 to produce 2 scales. They are individually compressed, and their reconstructions are combined in a residual way.

3. **3D-based Seismic Data Compression (3DSC)**: described in Subsection 3.2.1, instead of seismic sections, this method aims to directly compress seismic volumes. A 3D CNN is used to capture information from the three dimensions at the same time. Considering the similarity in the neighboring slices that constitute a volume, a set of them is represented using a latent representation corresponding to compress only one section.

4. **3D-based Seismic Data Compression using Vector Quantization (3DSC-VQ)**: described in Subsection 3.2.2, this method extends the 3DSC method. We propose to compress the whole volume without reductions to prevent subsampling inaccuracies. The vector quantization is used to reduce the computational cost of this approach.

Table 1 presents the main differences between our four approaches and the method proposed by Mentzer et al. (2018). The main objective of compressing the seismic data is reached by the 2DSC method. Improvements are added through successive approximations from the previous methods. From the 2DSC to 2DSC-MR, we incorporate a multi-resolution architecture in our encoder to extract features from different scales. The 3DSC method extends the 2DSC approach, by compressing the volume instead of its bi-dimensional sections. The 3DSC-VQ extends the previously mentioned 3D method, in which a vector quantization is performed and a set of indices $\mathcal{I}$ is used to reduce the computational cost of the probabilistic model.

Table 1 – Main differences between the approaches.

Method	Autoencoder input domain	Probabilistic model input domain	Quantization	Multiple scales	Distortion metric
Mentzer et al. (2018)	$\mathbb{N}^{h \times w \times 3}$	$\mathcal{C}^{h/8 \times w/8 \times C}$	Scalar	No	MS-SSIM
2DSC	$\mathbb{R}^{h \times w \times 1}$	$\mathcal{C}^{h/8 \times w/8 \times C}$	Scalar	No	PSNR
2DSC-MR	$\mathbb{R}^{h \times w \times 1}$	$\mathcal{C}^{h/8 \times w/8 \times C}$	Scalar	Yes	PSNR
3DSC	$\mathbb{R}^{h \times w \times d}$	$\mathcal{C}^{h/8 \times w/8 \times C}$	Scalar	No	PSNR
3DSC-VQ	$\mathbb{R}^{h \times w \times d}$	$\mathcal{I}^{h/8 \times w/8 \times C}$	Vectorial	No	PSNR

Source: created by the author.

To compress the seismic data, we can treat it as a set of 2D seismic sections or directly as a volume. The 2DSC and 2DSC-MR methods are described in Section 3.1. They are based on bi-dimensional compression, where the volume is detached into a set of slices. The 3DSC and 3DSC-VQ are presented in Section 3.2. They are related to the three-dimensional compression, where the data is seen in its volumetric form.

3.1 2D POST-STACK COMPRESSION

Our 2DSC method, described in Subsection 3.1.1, aims to verify the main hypothesis of this work. The main hypothesis is that deep neural networks can compress seismic data at low bit rates preserving most of its underlying structural information. This method is based on bi-dimensional compression, in which a volume is detached into 2D sections and each one is compressed separately. It is an extension from Mentzer et al. (2018), where image compression is performed.

The 2DSC-MR approach intends to provide improvements for the previous approach. From the premise that a scale can yield information that is no easily captured by others, this method aims to verify the second hypothesis, in which the information across scales improves the compression. This method extends the 2DSC by adding a multi-resolution architecture, as described in Subsection 3.1.2.

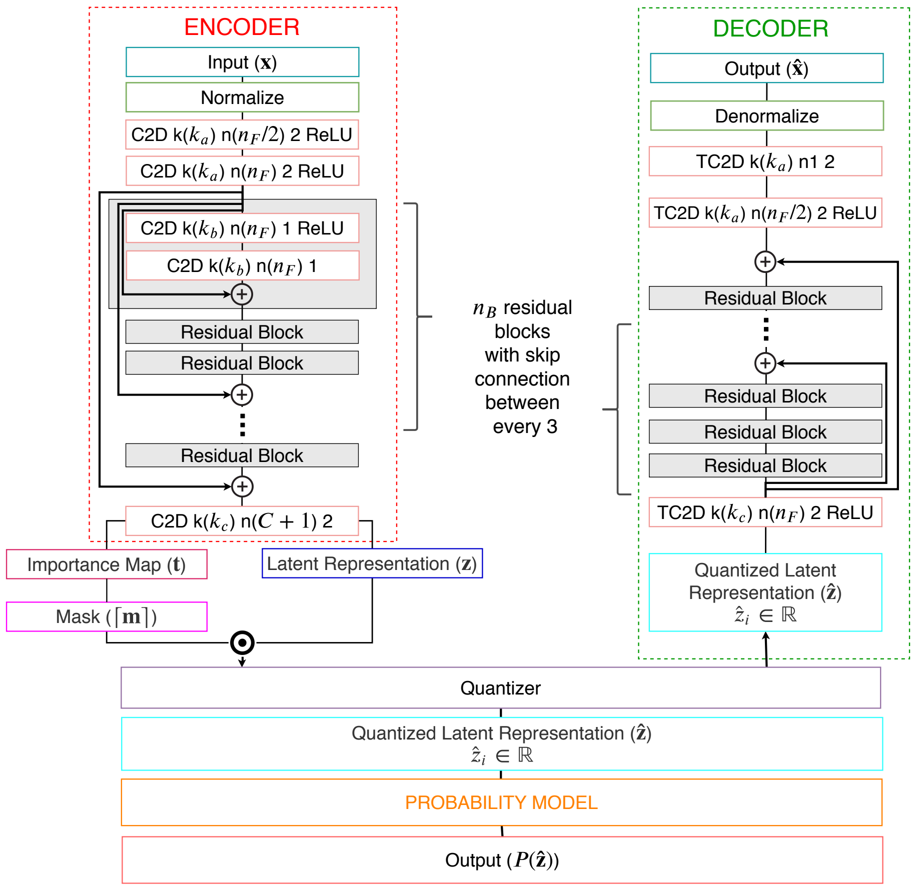
3.1.1 **2D-based Seismic Data Compression (2DSC)**

The 2DSC approach is the most straightforward adaptation from the Mentzer et al. (2018) proposal to the seismic domain. The main difference is that instead of compressing 3-channel 8-bit unsigned integers images, we compress seismic sections represented as

1-channel 32-bit floating-points. It is possible to adapt the slices to the network input, but the quantization from 32-bit floating-points to 8-bit unsigned integers introduces a loss of precision that cannot be recovered. Moreover, the model was originally trained using the perceptual loss MS-SSIM, and it does not preserve quantitative aspects that are important to the seismic domain. For this reason, we adapted the network and all metrics to work with 1-channel 32-bit floating-points seismic data and trained all models from scratch.

Figure 12 shows our autoencoder and probabilistic model architectures. From the encoder, the parameter n_B indicates the number of residual blocks. The parameter n_F is related to the number of filters of the convolutional layers. k_a, k_b and k_c describes the kernel size of the convolutions in the initial, middle and final parts of the encoder architecture, respectively. The parameter C is the number of filters of the latent space.

Figure 12 – Detailed architecture of our 2DSC method. It uses the probabilistic model from Figure 9.


ENCODER
Input (x)
Normalize
C2D k(k_a) n($n_F/2$) 2 ReLU
C2D k(k_a) n(n_F) 2 ReLU
C2D k(k_b) n(n_F) 1 ReLU
C2D k(k_b) n(n_F) 1
Residual Block
Residual Block
Residual Block
C2D k(k_c) n($C+1$) 2
Importance Map (t)
Mask (⌈m⌉)
Latent Representation (z)
n_B residual blocks with skip connection between every 3
DECODER
Output ($\hat{x}$)
Denormalize
TC2D k(k_a) n1 2
TC2D k(k_a) n($n_F/2$) 2 ReLU
Residual Block
Residual Block
Residual Block
Residual Block
TC2D k(k_c) n(n_F) 2 ReLU
Quantized Latent Representation ($\hat{z}$) $\hat{z}_i \in \mathbb{R}$
Quantizer
Quantized Latent Representation ($\hat{z}$) $\hat{z}_i \in \mathbb{R}$
PROBABILITY MODEL
Output ($P(\hat{z})$)

Source: created by the author.

The input $\mathbf{x} \in \mathbb{R}^{w \times h}$ is normalized using the mean and standard deviation from the training set. Several convolutional layers are applied aiming to generate a latent representation $\mathbf{z} \in \mathbb{R}^{w/8 \times h/8 \times C}$ and an importance map $\mathbf{t} \in \mathbb{R}^{w/8 \times h/8 \times 1}$ from the input. As in the original method, a mask $\lceil \mathbf{m} \rceil$ is constructed using $\mathbf{t}$ and applied to $\mathbf{z}$.

The same quantizer described in Section 2.3 is used to allow the seismic section compression. In this scheme, the entries of $\mathbf{z}$ are replaced by the nearest centroid value. It yields a volume $\hat{\mathbf{z}}$ that has at most L different values. This representation fed the decoder and the probabilistic model. The decoder structure mirrors the encoder aiming to reconstruct the input. Its last convolutional layer outputs a single channel slice that is denormalized to generate the output $\hat{\mathbf{x}}$. The probabilistic model is the same from Figure 9, and outputs the joint distribution of $\hat{\mathbf{z}}$ to bit rate estimation.

Different from Mentzer et al. (2018), we use the PSNR metric as distortion function. In this way, both aspects qualitative and quantitative tend to be preserved. However, PSNR is a metric that must be maximized, whereas the entropy must be minimized. To solve this problem, instead of minimizing the PSNR, we minimize δ - PSNR, where δ is a scalar greater than the PSNR maximum value. The loss functions for autoencoder and probabilistic model are the same described in Equations 2.13 and 2.11, respectively. We use MAX = 1.0 in Equation 2.2 for loss evaluation. The first version of this approach was submitted and accepted to the 81st EAGE Conference and Exhibition (NAVARRO et al., 2019) and to the 19th International Conference on Computational Science and its Applications (SCHIAVON et al., 2019).

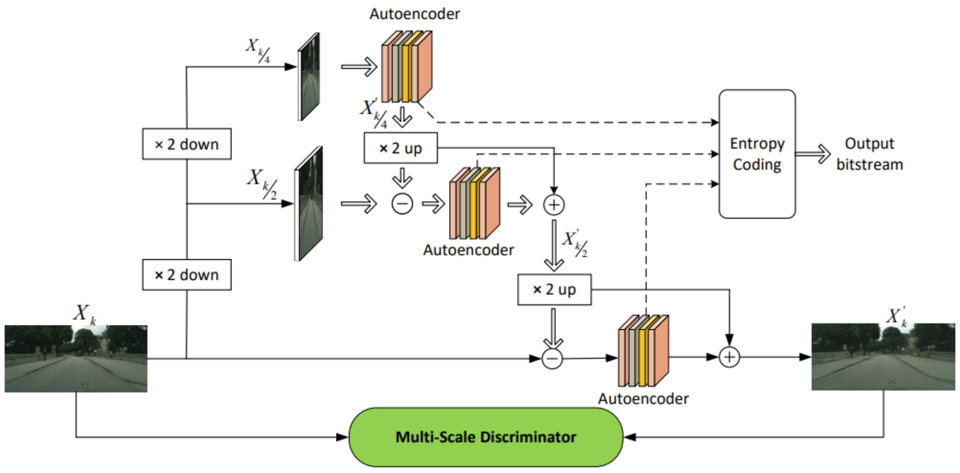
3.1.2 **2D-based Seismic Data Compression using Multi-resolution (2DSC-MR)**

Multi-resolution is a concept from the Signal Processing related to decomposing a signal into similar ones with different resolutions. The interpolation yields an upsampled signal whereas the decimation downscales the input. With signals in different resolutions, different information can be extracted from them. From this premise, we propose to extend the 2DSC method by introducing multiple resolutions information. Methods that take only one single scale as input fail to capture the scale-dependent information. It is expected that the independent analysis of multiple scales captures more relevant information than using only the input resolution.

Inspired by the architecture proposed by Huang et al. (2019), the 2DSC-MR method extends the 2DSC approach by decomposing the seismic section in three scales and combining them in a residual way to generate the latent representation. It allows the network to deal with different scales learning the most important features across them. Figure 13 shows the multi-resolution architecture proposed by Huang et al. (2019) that inspired us. The input is downscaled by a factor of 2 to generate two more inputs. The input scales are progressively encoded, from coarse ($X_{k/4}$) to fine (X_k), and their reconstructions are combined with the next scale input. The residual scheme benefits the information that was not observed in the previously encoded scale. The intermediary latent representations are combined to generate the output bitstream.

Different from Huang et al. (2019), we perform the scaling by using filtering. In

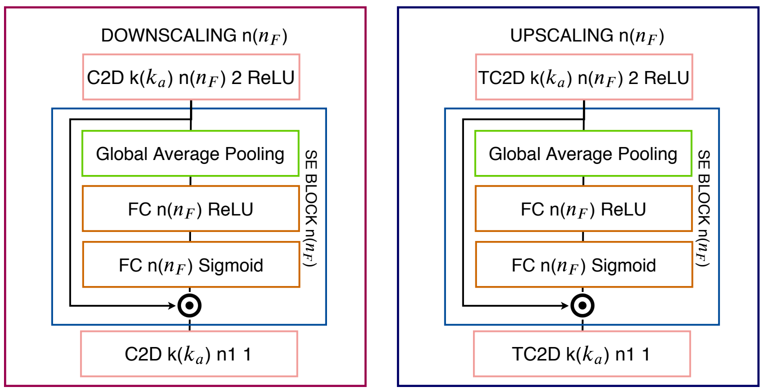
Figure 13 – Multi-resolution architecture proposed by Huang et al. (2019). The multi-resolution scheme captures the most important features across them.



Source: extracted from Huang et al. (2019).

this setting, the network learns the filters that are more suitable for this task. As shown in Figure 14, given an input, a convolutional layer is used to learn a filter bank. The filter bank is weighted by a Squeeze-Excitation (SE) block (HU; SHEN; SUN, 2018). The SE block is composed of a global average pooling layer, that computes the global average of each feature map from the input, followed by two fully connected layers. The idea is to benefit the filters that provide more relevant information before performing the dimensionality reduction. This operation is done by using a convolutional layer that outputs the downscaled (left) or upscaled (right) input. The downscaling uses regular convolutions, whereas the upscaling works with transposed ones. The term $n(n_F)$ indicates the number of filters learned by the convolutional layers and the size of the fully connected outputs.

Figure 14 – Downscaling (left) and upscaling (right) blocks. The main difference is the use of convolutional layer for downscaling and transposed convolutions for upscaling.

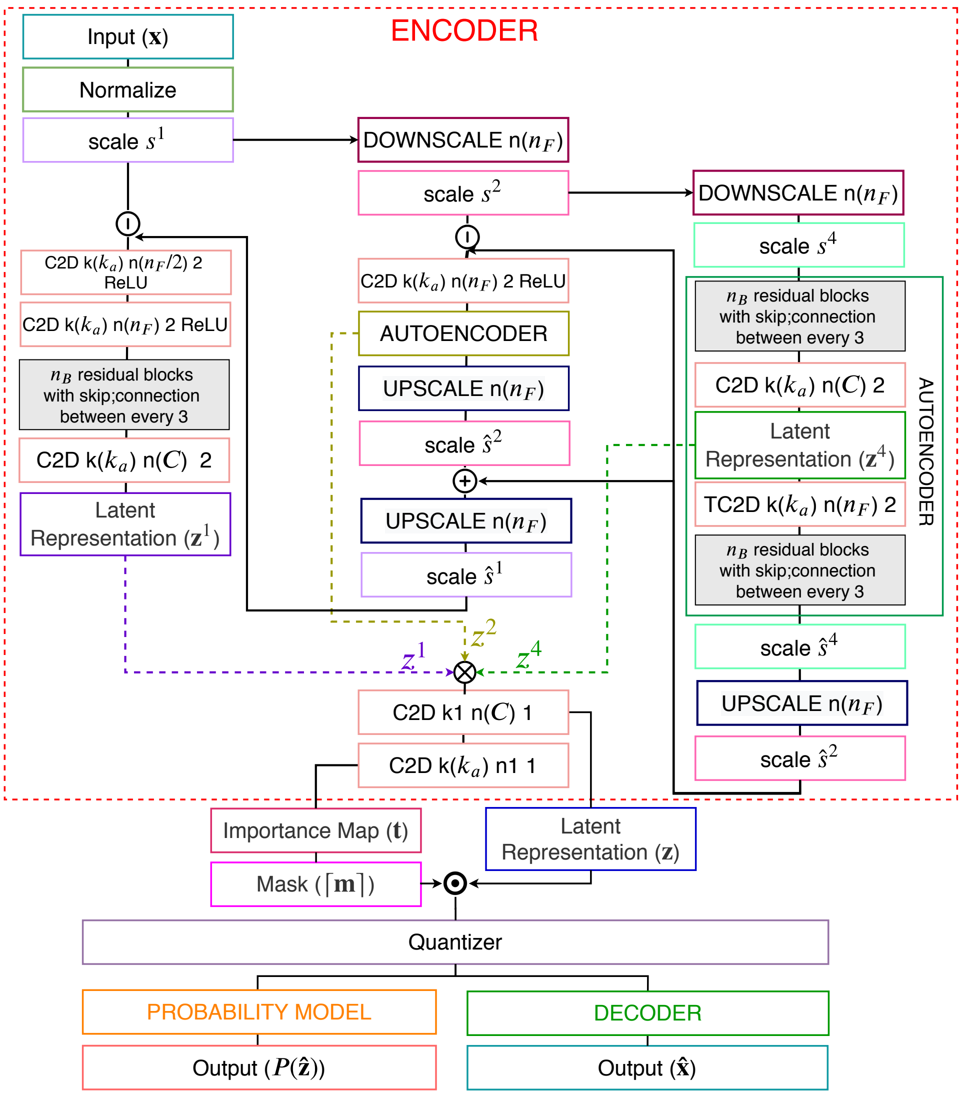


Source: created by the author.

Figure 15 illustrates our multi-resolution architecture. Similarly to Huang et al.

(2019), the original input $\mathbf{x} \in \mathbb{R}^{w \times h}$ is downscaled by a factor of 2 and the result is downscaled again by a factor of 2 to produce 3 final scales $s^1 \in \mathbb{R}^{w \times h}$, $s^2 \in \mathbb{R}^{w/2 \times h/2}$ and $s^4 \in \mathbb{R}^{w/4 \times h/4}$. The scales are progressively encoded, from coarse (s^4) to fine (s^1), and their reconstructions $\hat{s}^1$, $\hat{s}^2$ and $\hat{s}^4$ are combined in a residual scheme. The latent representations $\mathbf{z^1}$, $\mathbf{z^2}$ and $\mathbf{z^4} \in \mathbb{R}^{w/8 \times h/8 \times C}$ are concatenated instead of separately being encoded. A 1×1 convolutional layer is applied to select the C more representative feature maps from all scales. This avoids manually selecting the number of filters for each scale and yields a single final latent representation $\mathbf{z} \in \mathbb{R}^{w/8 \times h/8 \times C}$. An additional convolutional layer is used to generate the importance map from $\mathbf{z}$. With the importance map and the latent representation computed, the remaining steps are performed as in the 2DSC approach.

Figure 15 – Detailed architecture of our 2DSC-MR method. We use three different resolutions, the original input, one downscaled by a factor of 2 and one downscaled twice by a factor of 2. The smaller scales are encoded and combined in a residual manner. All intermediary latent representation are concatenated. The decoder and probabilistic model are the same from the 2DSC method.


ENCODER
Input ($\mathbf{x}$)
Normalize
scale s^1
DOWNSCALE n(n_F)
scale s^2
DOWNSCALE n(n_F)
scale s^4
C2D k(k_a) n(n_F/2) 2 ReLU
C2D k(k_a) n(n_F) 2 ReLU
n_B residual blocks with skip;connection between every 3
C2D k(k_a) n(C) 2
Latent Representation ($\mathbf{z}^1$)
C2D k(k_a) n(n_F) 2 ReLU
AUTOENCODER
UPSCALE n(n_F)
scale $\hat{s}^2$
UPSCALE n(n_F)
scale $\hat{s}^1$
n_B residual blocks with skip;connection between every 3
C2D k(k_a) n(C) 2
Latent Representation ($\mathbf{z}^4$)
TC2D k(k_a) n(n_F) 2
n_B residual blocks with skip;connection between every 3
AUTOENCODER
scale $\hat{s}^4$
UPSCALE n(n_F)
scale $\hat{s}^2$
z^1
z^2
z^4
C2D k1 n(C) 1
C2D k(k_a) n1 1
Importance Map ($\mathbf{t}$)
Mask ($\lceil \mathbf{m} \rceil$)
Latent Representation ($\mathbf{z}$)
Quantizer
PROBABILITY MODEL
DECODER
Output ($P(\hat{\mathbf{z}})$)
Output ($\hat{\mathbf{x}}$)

Source: created by the author.

3.2 3D POST-STACK COMPRESSION

Although it is possible to compress post-stack seismic data through its bi-dimensional slices, the most natural way is considering the data as a volume. The 3D compression considers the spatial correlation in all directions, and it is expected to achieve lower bit rates at least comparable to bi-dimensional approaches. We propose to extend our 2DSC approach to compress volumes instead of slices. By dealing with volumes, the autoencoder becomes three-dimensional and the probabilistic model should be four-dimensional. However, due to memory constraints, a 4D neural network is still not practical. We propose two schemes to address this problem without needing a 4D probabilistic model.

The 3DSC method, described in Subsection 3.2.1, aims to verify our third hypothesis. It states that three-dimensional models can generate representations more suitable to 3D post-stack data compression than bi-dimensional approaches. This method performs a reduction in the depth dimension through the encoder. The volume is encoded into a latent representation that corresponds to the compression of only one section.

The 3DSC-VQ approach extends the 3DSC method. The depth reduction in the 3DSC method can lead to high losses in latent representations. Our 3DSC-VQ method, described in Subsection 3.2.2, intends to verify our fourth hypothesis. The fourth hypothesis is that the compression of the whole volume without sample depth reductions can improve the quality reached. In this sense, the depth size of the latent representation has the same depth size of the input sample. To use the same probabilistic model from the 2DSC approach we propose to reduce the depth dimension by using vector quantization. The loss from vector quantization is expected to be lower than the encoding part.

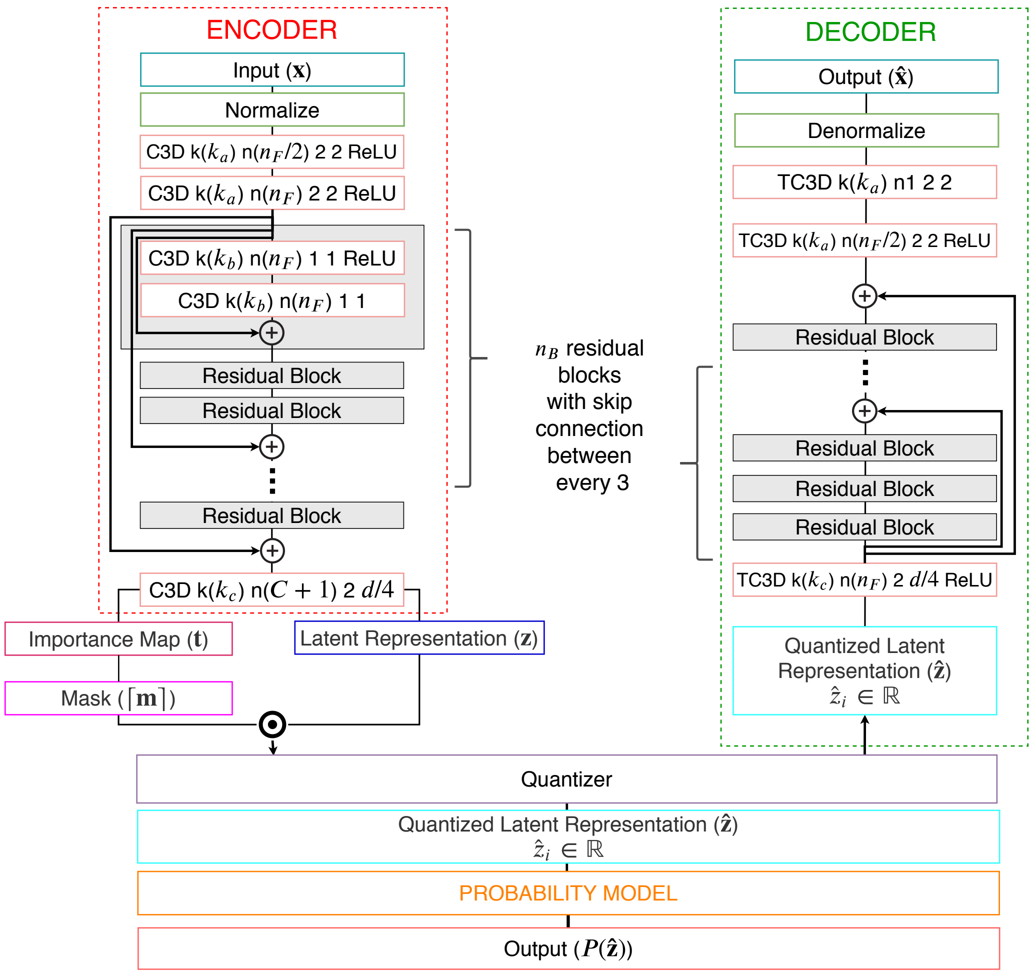
3.2.1 **3D-based Seismic Data Compression (3DSC)**

Seismic surveys are mappings from the subsurface that describe the geological structures present in the region where they were acquired. We can consider that there exists a similarity between neighboring slices since they describe spatially very close regions. In some cases, the difference between sections is subtle. In this context, it is reasonable to consider that a set of local slices could be represented by only one of them.

Our 3DSC approach extends the 2DSC by using 3D inputs instead of bi-dimensional sections. This method benefits from the similarity from slices in a region. The idea is to perform a depth reduction through convolutional layers so that a set of slices can be encoded with the cost of only one. The latent representation reached for the whole volume has the same dimensions from the latent representation for a single slice compression.

The 3DSC architecture is similar to the 2DSC. The main difference concerns the convolutional layers, in which 3D convolutions are performed. In this setting, the spatial correlation in all directions is considered. Figure 16 shows the details of our 3DSC architecture. Notice that the same structure from the 2DSC is maintained. The input volume $\mathbf{x} \in \mathbb{R}^{w \times h \times d}$ is normalized and encoded through several convolutional layers. The first two perform convolutions with stride 2 in all directions, reducing the dimensions by a factor of 4. If the volume is composed of more than 4 slices, the last layer of the encoder uses a depth stride of $d/4$. It ensures that the feature map depth becomes 1 at the latent representation $\mathbf{z} \in \mathbb{R}^{w \times h \times C}$. The decoder mirrors the encoder. The masking, quantization and bit rate estimation steps are performed as in the 2DSC approach.

Figure 16 – Detailed architecture of our 3DSC. It is the same base architecture of our 2DSC method, but with 3D convolutions.


ENCODER
Input ($\mathbf{x}$)
Normalize
C3D k(k_a) n(n_F/2) 2 2 ReLU
C3D k(k_a) n(n_F) 2 2 ReLU
C3D k(k_b) n(n_F) 1 1 ReLU
C3D k(k_b) n(n_F) 1 1
Residual Block
Residual Block
Residual Block
C3D k(k_c) n($C+1$) 2 d/4
Importance Map ($\mathbf{t}$)
Mask ($\lceil \mathbf{m} \rceil$)
Latent Representation ($\mathbf{z}$)
n_B residual blocks with skip connection between every 3
DECODER
Output ($\hat{\mathbf{x}}$)
Denormalize
TC3D k(k_a) n1 2 2
TC3D k(k_a) n(n_F/2) 2 2 ReLU
Residual Block
Residual Block
Residual Block
Residual Block
TC3D k(k_c) n(n_F) 2 d/4 ReLU
Quantized Latent Representation ($\hat{\mathbf{z}}$) $\hat{z}_i \in \mathbb{R}$
Quantizer
Quantized Latent Representation ($\hat{\mathbf{z}}$) $\hat{z}_i \in \mathbb{R}$
PROBABILITY MODEL
Output ($P(\hat{\mathbf{z}})$)

Source: created by the author.

3.2.2 3D-based Seismic Data Compression using Vector Quantization (3DSC-VQ)

The mapping from d slices to only one as in the previous method can lead to high losses. We propose the 3DSC-VQ to deal with this scenario. The idea is to compress a volume without depth reductions on the encoder. In comparison to the 3DSC, this method yields a higher dimensional latent space. We propose to use vector quantization to reduce the computational cost, avoiding the use of a 4D probabilistic model. By using vector quantization, we can better preserve information across slices once sufficient centroids are available.

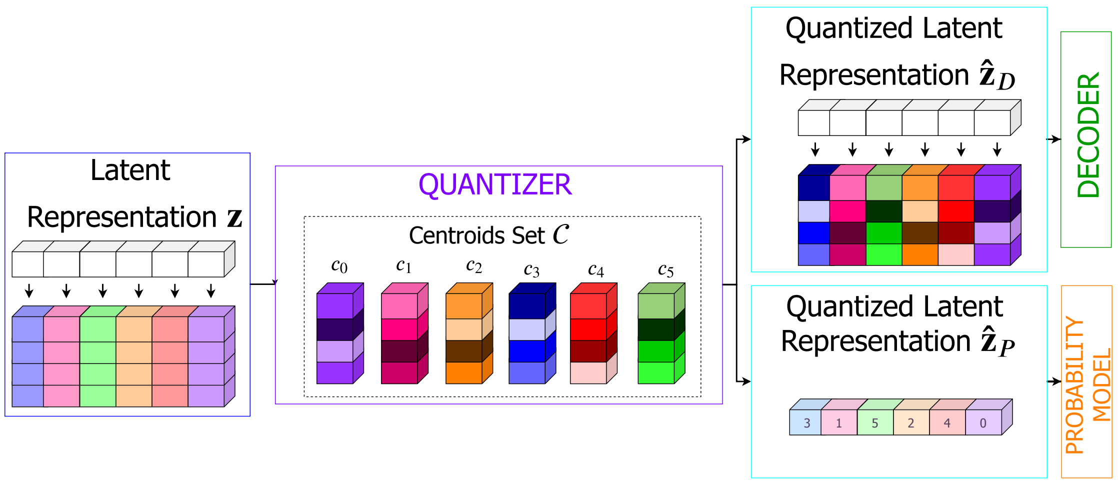
Considering $\mathbf{x} \in \mathbb{R}^{w \times h \times d}$ as an input volume, the encoder is similar to 3DSC method. The difference is in the stride, in which the vectorial approach does not perform reductions in the depth dimension of the input, yielding an encoded representation $\mathbf{z} \in \mathbb{R}^{w/8 \times h/8 \times d \times C}$. As in Mentzer et al. (2018), the importance map $\mathbf{t} \in \mathbb{R}^{w/8 \times h/8 \times d \times 1}$ is used to generate a mask. However, is not possible to use Equation 2.5, as $\mathbf{t}$ has an extra dimension. The mask $\mathbf{m} \in \mathbb{R}^{w/8 \times h/8 \times d \times C}$ is generated as follows:

$$m_{w',h',d',c'} = \begin{cases} 1 & \text{if } c' < t_{w',h',d'} \\ (t_{w',h',d'} - c') & \text{if } c' \leq t_{w',h',d'} \leq c' + 1. \\ 0 & \text{if } c' + 1 > t_{w',h',d'} \end{cases} \tag{3.1}$$

As in Equation 2.6, $\mathbf{m}$ is binarized and a point-wise multiplication is performed between $\lceil \mathbf{m} \rceil$ and $\mathbf{z}$. In this setting, the importance map is not used with the goal of weighting across regions, as in the previous methods. Instead, it is used to reduce the complexity of the latent space, by introducing zero values in some entries.

The vector quantization is performed as in Figure 17. We reshape $\mathbf{z}$ into a matrix $Z = [\vec{z}_1, \cdots, \vec{z}_n]$, so that $\vec{z}_i \in \mathbb{R}^d$ and $n = w/8 \cdot h/8 \cdot C$. The idea is to quantize each $\vec{z}_i$ according to a set of vectorial centroids $\mathcal{C} = \{\vec{c}_1, \cdots, \vec{c}_L\}$ of centroids, where $\vec{c}_l \in \mathbb{R}^d$. The quantizer $\mathbf{Q} : \mathbb{R}^{w/8 \times h/8 \times d \times C} \to \mathcal{C}^{w/8 \times h/8 \times C}$ maps $\mathbf{z}$ into a quantized representation $\hat{\mathbf{z}}_D$ by using Equation 2.7 and its soft quantization is performed through Equation 2.8. The quantized latent representation $\hat{\mathbf{z}}_D$ can be used by the decoder to reconstruct the output volume $\hat{\mathbf{x}} \in \mathbb{R}^{w \times h \times d}$. The decoder mirrors the encoder architecture.

Figure 17 – Vector quantization scheme. Each vector is replaced by its nearest centroid to yield $\hat{\mathbf{z}}_D$ and the index of the nearest centroid is used to yield a volume $\hat{\mathbf{z}}_P$ that is used to feed the probability model.


Latent Representation $\mathbf{z}$
QUANTIZER
Centroids Set C
c_0
c_1
c_2
c_3
c_4
c_5
Quantized Latent Representation $\hat{\mathbf{z}}_D$
DECODER
Quantized Latent Representation $\hat{\mathbf{z}}_P$
3
1
5
2
4
0
PROBABILITY MODEL

Source: created by the author.

To compute the bit rate estimation over $\hat{\mathbf{z}}_D$, a 4D probabilistic model is needed. However, due to memory constraints, it is still impracticable. To generate a volume that can feed the 3D probabilistic model we propose to replace each vector $\vec{z}_i$ by the index l of the nearest centroid $\vec{c}_l$. It is performed aiming to generate a volume $\hat{\mathbf{z}}_P$ that keeps the relationship between the feature maps that constitute the latent representation and that can feed the 3D probabilistic model. Similarly to Equations 2.7 and 2.8, each $\vec{z}_i$ can be replaced by the index of its nearest centroid as follows:

$$\hat{z}_{Pi} = \arg\min_{l \in \mathcal{C}} \|\vec{z}_i - \vec{c}_l\|, \tag{3.2}$$

and its soft quantization

$$\tilde{z}_{Pi} = \sum_{l=1}^{L} \frac{\exp(-\sigma \|\vec{z}_i - \vec{c}_l\|)}{\sum_{j=1}^{L} \exp(-\sigma \|\vec{z}_i - \vec{c}_j\|)} l = \sum_{l=1}^{L} \mathrm{softmax}(-\sigma \|\vec{z}_i - \vec{c}_l\|) l, \tag{3.3}$$

is performed in the backward pass of the backpropagation. Hence, $\hat{\mathbf{z}}_P \in \mathcal{I}^{w/8 \times h/8 \times C}$, where $\mathcal{I} = \{1, \ldots, L\}$ is the set of indices of $\mathcal{C}$. Since $\mathcal{I} \subset \mathbb{R}$, $\hat{\mathbf{z}}_P$ can feed the model $\mathbf{P}$ described in Section 2.3 without loss of generality.

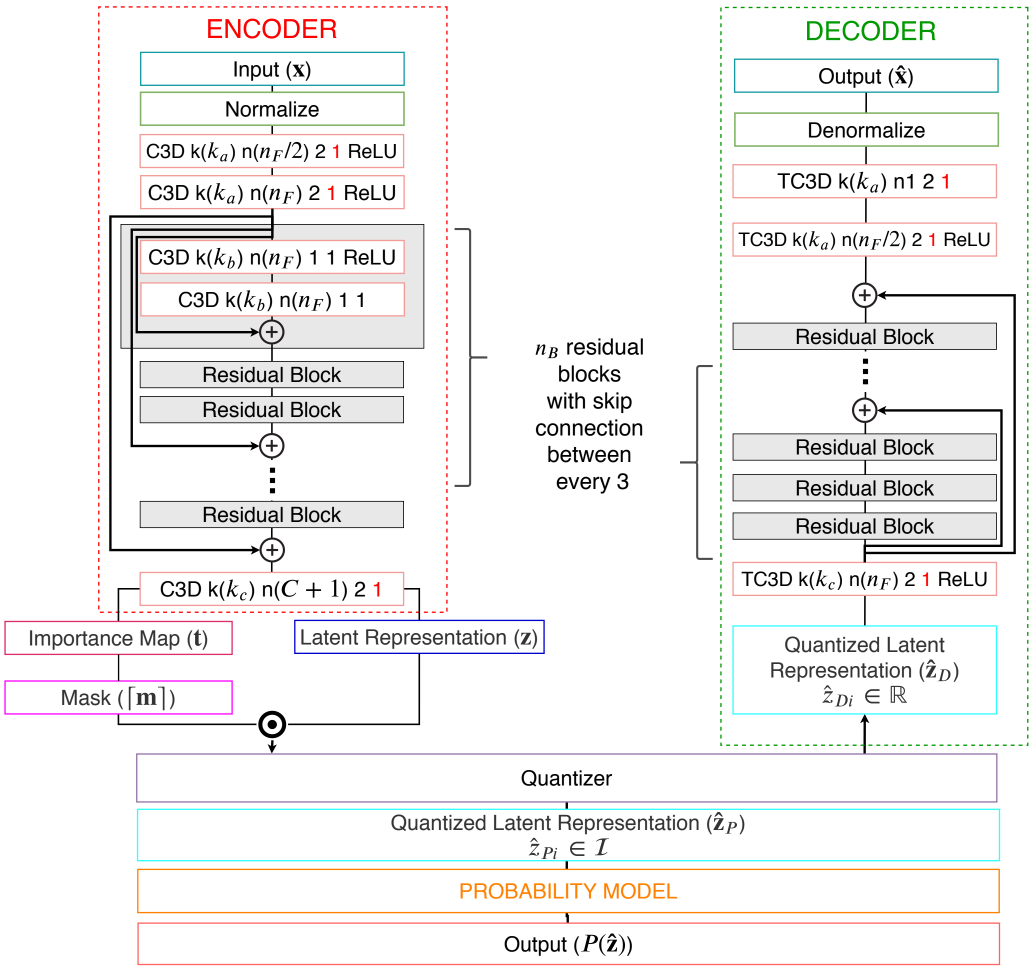
Since the mask $\lceil \mathbf{m} \rceil$ has d times more elements than the quantized latent representation $\hat{\mathbf{z}}_P$, the autoencoder loss function from Equation 2.13 cannot be used. Then, the loss function of the autoencoder and quantizer is given by:

$$\mathcal{L}_{\mathbf{E},\mathbf{D},\mathbf{Q}} = \mathbf{d}(\mathbf{x}_j, \hat{\mathbf{x}}_j) + \beta C(\hat{\mathbf{z}}) + R_{AE} \cdot \ell_2(\mathbf{W}_{AE}) + R_{\mathcal{C}} \cdot \ell_2(\mathcal{C}), \tag{3.4}$$

where C is the same cross entropy loss from Equation 2.12. ℓ_2 is the regularization loss, the parameter R_{AE} is the autoencoder regularization factor, $\mathbf{W}_{AE}$ is the autoencoder weights matrix and $R_{\mathcal{C}}$ is the centroids regularization factor.

Figure 18 shows the architecture used in this method. It is very similar to the 3DSC approach. The main difference is that the encoder and decoder layers have stride 1 for the depth dimension. The last layer of the encoder outputs an importance map that is used to weight the latent representation $\mathbf{z}$. The quantizer is responsible to assign each vector from Z to its nearest centroid. The decoder uses it to reconstruct the input. A volume that contains the index from the nearest centroid is used by the probability model to compute the bit rate estimation.

Figure 18 – Detailed architecture of our 3DSC-VQ. The main difference to the previous method is the stride from convolutional layers (in red) and the probabilistic model input.


ENCODER
Input ($\mathbf{x}$)
Normalize
C3D k(k_a) n($n_F/2$) 2 1 ReLU
C3D k(k_a) n(n_F) 2 1 ReLU
C3D k(k_b) n(n_F) 1 1 ReLU
C3D k(k_b) n(n_F) 1 1
Residual Block
Residual Block
Residual Block
C3D k(k_c) n($C+1$) 2 1
Importance Map ($\mathbf{t}$)
Mask ($\lceil \mathbf{m} \rceil$)
Latent Representation ($\mathbf{z}$)
n_B residual blocks with skip connection between every 3
DECODER
Output ($\hat{\mathbf{x}}$)
Denormalize
TC3D k(k_a) n1 2 1
TC3D k(k_a) n($n_F/2$) 2 1 ReLU
Residual Block
Residual Block
Residual Block
Residual Block
TC3D k(k_c) n(n_F) 2 1 ReLU
Quantized Latent Representation ($\hat{\mathbf{z}}_D$) $\hat{z}_{Di} \in \mathbb{R}$
Quantizer
Quantized Latent Representation ($\hat{\mathbf{z}}_P$) $\hat{z}_{Pi} \in \mathcal{I}$
PROBABILITY MODEL
Output ($P(\hat{\mathbf{z}})$)

Source: created by the author.

4 EXPERIMENTAL RESULTS SPECIFIC FOR EACH METHOD

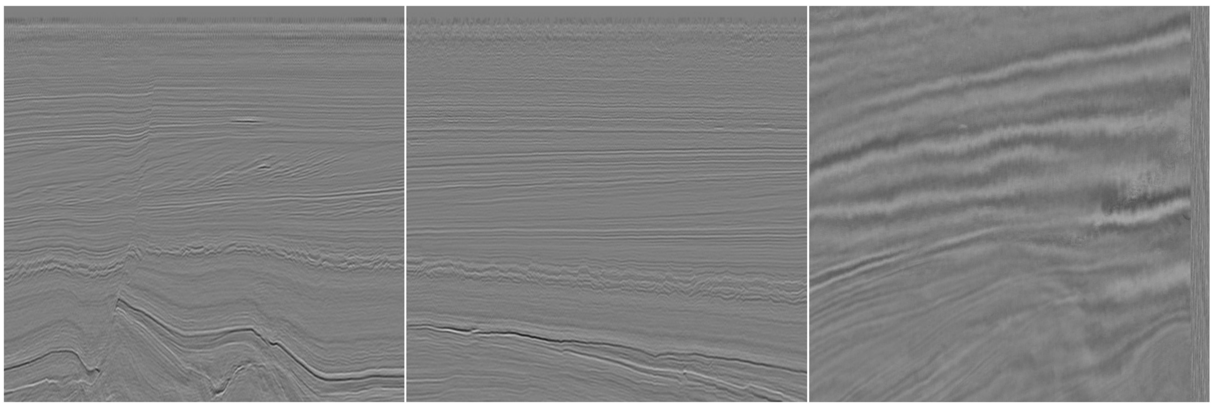
This chapter provides the details about the seismic data, protocols and experiments performed for each method separately. We proposed four methods: 2DSC, 2DSC-MR, 3DSC and 3DSC-VQ. Each one has intrinsic details that require us to evaluate its hyper-parameters individually. Section 4.2 describes the proposed training and inference protocols that allow seismic compression. Section 4.3 describes the common setup for all methods. Experiments based on the 2D autoencoders are described in Section 4.4 and based on 3D are described in Section 4.5.

4.1 3D POST-STACK SEISMIC DATA

The 3D post-stack seismic data is represented as a 32-bit floating point representation matrix $\mathbf{v} \in \mathbb{R}^{\mathfrak{I} \times \mathfrak{C} \times \mathfrak{T}}$, in which $\mathfrak{I}$ is related to the inline survey direction, constituted of planes (y, z), $\mathfrak{C}$ is the crossline direction, with (x, z) planes and $\mathfrak{T}$ is the time-depth direction, composed of (x, y) planes.

Figure 19 illustrates the three possible views of a seismic volume. The inline and crossline directions are similar, as they describe the front and left views of the volume. The time-depth direction is related to the top view of the volume and is often very different from the others.

Figure 19 – Inline (left), crossline (middle) and time-depth (right) views for the Netherlands F3-Block seismic data. Notice that the time-depth is very different from the others.

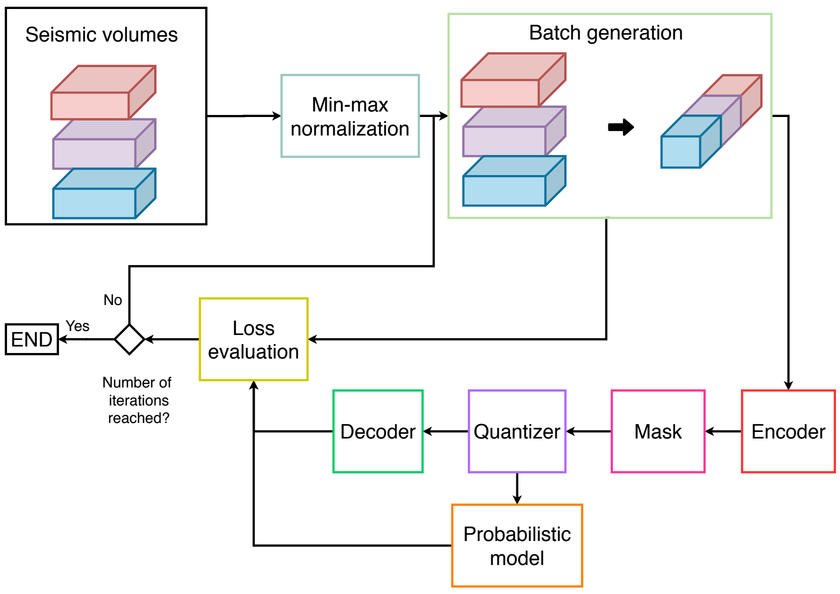


Source: created by the author.

4.2 TRAINING AND INFERENCE PROCEDURE

This section describes training and inference schemes that allow compressing seismic volumes using the previously described methods. Figure 20 illustrates the training procedure proposed by this work. As a pre-processing step, the volumes 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.

Figure 20 – Flowchart of our training procedure. We first pre-process all our datasets with a min-max normalization. Then the training proceeds producing each batch and sending it to the autoencoder. After evaluating the loss, the weights are adjusted with the backpropagation algorithm. These steps are repeated until a fixed number of iterations is reached.

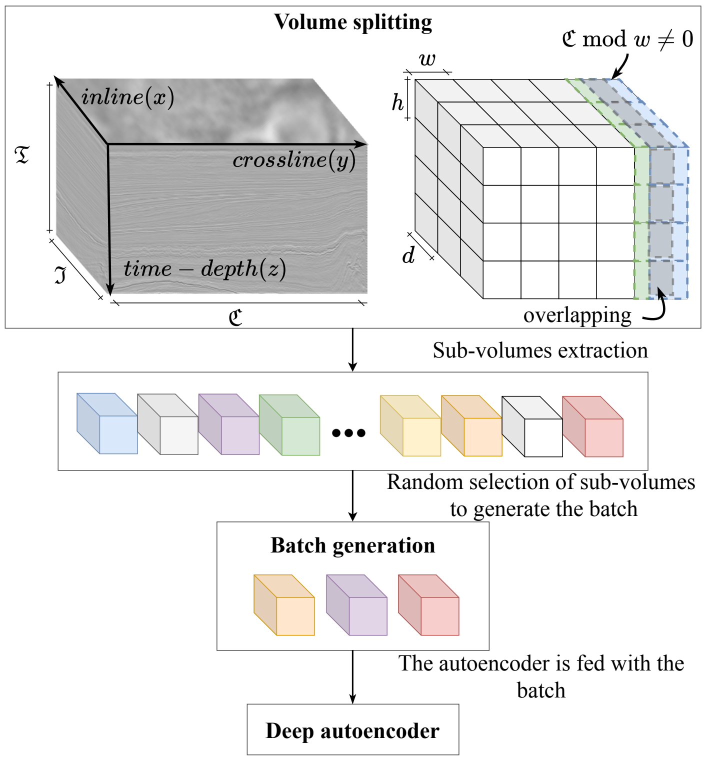


Source: created by the author.

The batch generation step is performed by extracting a set of samples from the seismic volume. For 2D approaches, we extract slices, and for 3D, sub-volumes. The samples can be extracted from the inline (x), crossline (y) or time-depth (z) directions. Samples from all directions can be used to train the network at the same time. 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.

The batch generation for 2D models is performed by extracting random crops of size $h \times w$ in an arbitrary direction. In contrast, three-dimensional approaches do not consider a random extraction of crops due to memory constraints. The batch generation step for 3D approaches is performed as depicted in Figure 21. Initially, we extract a set of sub-volumes from the whole seismic volume $\mathbf{v} \in \mathbb{R}^{\mathfrak{I} \times \mathfrak{C} \times \mathfrak{Z}}$. 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 bi-dimensional 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.

Figure 21 – Batch generation step for 3D models. The volume is detached in a set of sub-volumes that are randomly selected to generate the batch.



Source: created by the author.

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 $\mathfrak{T}, \mathfrak{C}$, or $\mathfrak{I}$ 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 21 shows an example in which the dimension $\mathfrak{C}$ is not divisible by w. In this sense, the last sub-volumes extracted (blue) have $w - (\mathfrak{C} \bmod w)$ columns overlapping the previous sub-volume extracted (green). An analogous process is used to extract crossline and time-depth samples.

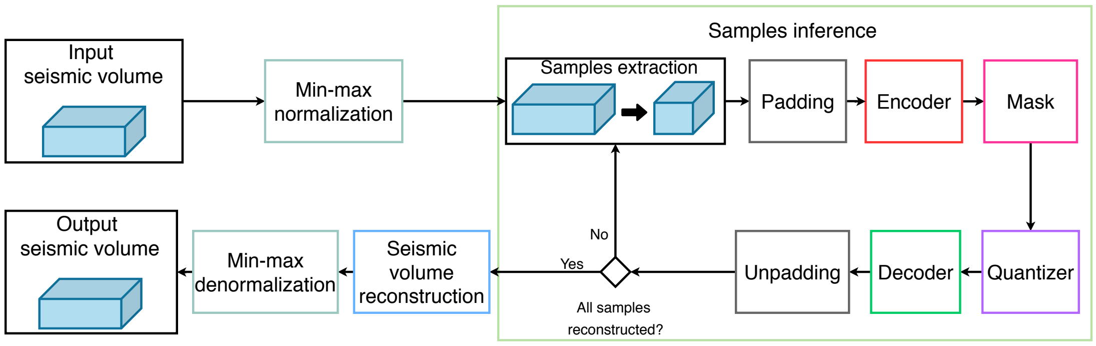
The batch is generated by randomly selecting sub-volumes extracted from the input volume. In the case of a training set composed of various datasets, we attempt to reduce the dataset bias by building the batch with samples from all of them at the same amount. The number of samples of a dataset can vary,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

The encoder is fed with the batch and a mask is applied to the encoded output to spatially attend different regions of the data with different bit rates. The masked representation is quantized, and it is used by the probabilistic model to estimate the joint probability of the quantized latent representation. The decoder reconstructs the input.

With the input reconstructed, the loss function is evaluated and it is used to adjust the network weights. We train our models using the PSNR metric as the distortion function. In this way, both aspects qualitative and quantitative tend to be preserved. The training step is repeated until the maximum number of iterations is reached.

The inference is performed according to the scheme shown in Figure 22. Considering a trained network, the seismic volume is normalized. Slices of size $\mathfrak{I} \times \mathfrak{C}$ are extracted for 2D approaches, and volumes of size $\mathfrak{I} \times \mathfrak{C} \times d$ for the 3D inference. If their spatial dimensions 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.

Figure 22 – Flowchart of our inference procedure. The testing volume is first min-max normalized. Then we extract samples across the inline or crossline directions. A padding operation is performed in case the volume size is not divisible by the subsampling factor. These samples are passed through the autoencoder and after all samples reconstructed, the volume is reconstructed and denormalized.



Source: created by the author.

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.

4.3 SETUP FOR ALL METHODS

To evaluate our proposed methods, we perform experiments in eight 3D post-stack seismic volumes, with distinct characteristics, selected from the SEG Open Data repository (SEG, 2019). We refer the reader to details such as size and grid dimension in Table 2. The grid dimensions are reported using the inline × crossline × time-depth order. The compression difficulty was chosen according to observations of the presence of high and low frequencies on these surveys. Surveys with the predominance of low frequencies are considered easy, and the difficulty level increases according to the presence of high frequencies. Our methods were implemented using the TensorFlow framework, and all experiments performed on NVIDIA Tesla V100 GPUs with 32GB and 16GB. 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 each amplitude value of the volume.

Table 2 – Uncompressed dataset properties.

Dataset	Size (GB)	Grid Dimension (voxels)	Compression Difficulty
Kahu3D	5.8	$(584 \times 1695 \times 1498)$	low
Kerry3D	0.8	$(226 \times 711 \times 1218)$	high
Netherlands F3-Block	1.2	$(631 \times 951 \times 463)$	medium
Opunake3D	2.6	$(501 \times 988 \times 1301)$	low
Parihaka3D	3.7	$(920 \times 1124 \times 874)$	low
Penobscot3D	0.6	$(401 \times 301 \times 1251)$	low
Poseidon3D	1.4	$(301 \times 967 \times 1176)$	high
Waihapa3D	0.3	$(201 \times 291 \times 1238)$	medium

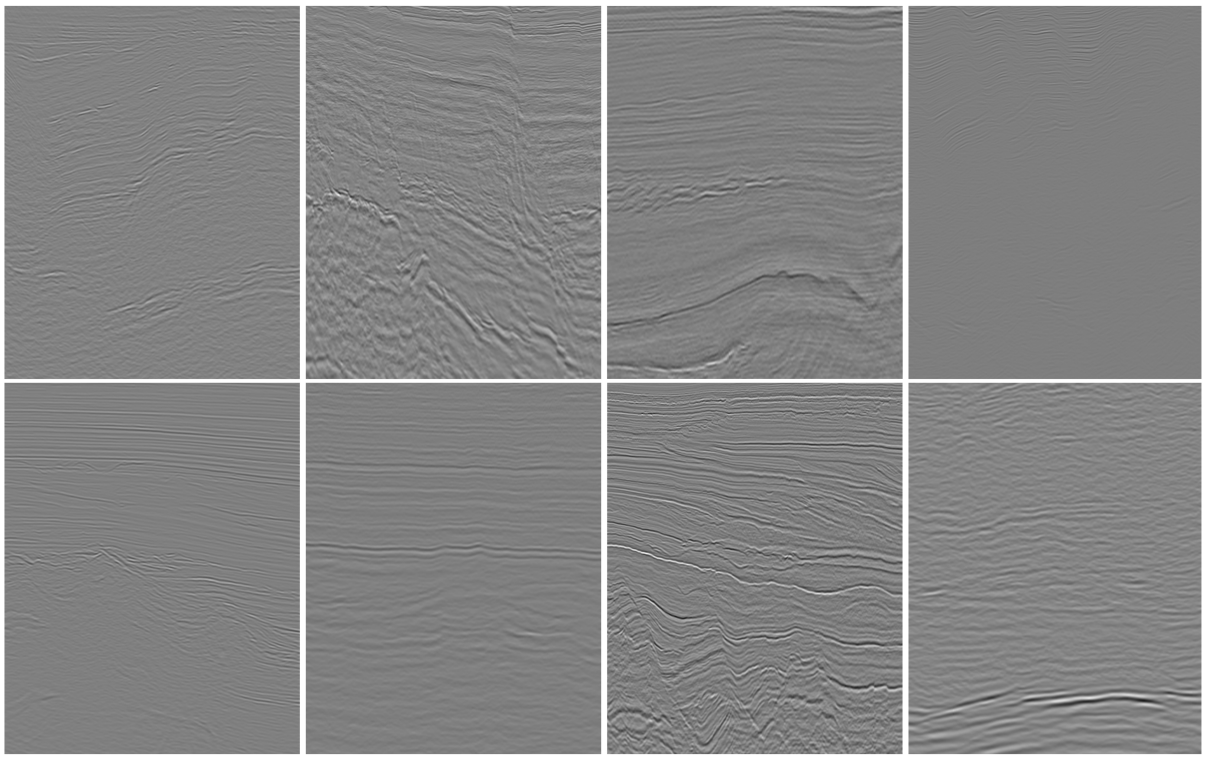
Source: created by the author.

Figure 23 depicts example slices for all surveys taken in the inline direction. The presence of many high frequencies regions in the surveys Kerry3D and Poseidon3D makes them difficult to compress. The Netherlands F3-Block and Waihapa3D surveys have a balance between low and high frequencies. The Kahu3D, Opunake3D, Parihaka3D and Penobscot3D are more homogeneous, with only a few regions with high frequencies.

To evaluate our methods, we empirically split our seismic volumes into three sets, according to the difficulty level. The first is the training set, composed of the Parihaka3D, Poseidon3D and Netherlands F3-Block surveys, in which each survey has a difficulty level. The second contains the Kerry3D and is used to find the best hyper-parameter setting and also as validation during the training. The remaining post-stack data are used for testing.

Due to their similarity, we use the inline and crossline directions alongside to train our models. Since the time-depth direction is too different from the previous ones, we

Figure 23 – Examples of inline sections extracted from the surveys of the Table 1. From left to right, from top to down: Kahu3D, Kerry3D, Netherlands F3-Block, Opunake3D, Parihaka3D, Penobscot3D, Poseidon3D, Waihapa3D.



Source: created by the author.

chose not to use it, as the top view is often too noisy compared with the other planes, causing the network to not converge. The batch is built with samples from all of them, at the same amount. The validation and testing steps consider only the inline samples.

For each method, we perform individual experiments, aiming to define its best parameter setting. Some of them were defined for all tests to allow comparison. As we need to compare the methods over the same bit rate, we define $\beta = 100$. This parameter controls the trade-off between distortion and bit rate. We also set $\delta = 100$ and $\sigma = 1$. We train our models for 30 epochs using the Adam optimizer with a batch composed of 30 samples. The model with the smallest validation loss during the training step is saved.

As proposed by Mentzer et al. (2018), to train our models to a target bit rate r_t, a clipping is used on the entropy term to force the bit rate target to be reached. The $\max(\beta\mathbf{r}(\hat{\mathbf{z}}), r_t)$ is used instead of $\beta\mathbf{r}(\hat{\mathbf{z}})$ in Equation 2.13 when optimizing the autoencoder. As we need to compare objectively, we set $r_t = 1.0$ bpv. We select the parameters that provide the best PSNR, excluding the results with bpv beyond 120% of the target value r_t. In fact, this margin is enough to deal with differences arising from network initialization. It is possible that a network obtains a high PSNR but cannot converge to the desired bit rate. Thus, the comparison between compression rates of different networks needs an acceptance criterion for the obtained bpv. We chose to exclude from comparisons the results having

$$bpv > 1.2 \cdot r_t, \tag{4.1}$$

since they become inadequate even with a high PSNR.

For the probability model $\mathbf{P}$, we fixed the architecture as the same proposed by Mentzer et al. (2018). We assume that the parameters determined by Mentzer et al. (2018) are suitable for seismic compression. In our experiments, we use the Adam optimizer with a learning rate of 8×10^{-5} and a step decay of 0.05 for every 10 epochs.

4.4 2D POST-STACK COMPRESSION

This section describes the experiments related to 2D seismic compression. To evaluate the 2DSC and 2DSC-MR methods, we train our models with a crop size of 160×160 pixels. The initial learning rate is 8×10^{-5} with a weight decay of 0.05 every 10 epochs.

4.4.1 **2D-based Seismic Data Compression (2DSC)**

In Section 4.4.1.1 we present the experiments performed to find the best parameters setting and in Section 4.4.1.2 we present the method performance evaluation on testing surveys.

4.4.1.1 *Parameters Setting*

A sequence of experiments was performed to find the best parameters for the 2DSC method according to Table 3. A grid-search over all of them would require more than 30000 models to train. The simplest model requires about 5 hours to be completed in a Tesla V100 GPU with 16GB of memory. It would be impractical, and so we evaluate them progressively according to their sensibility, starting from the least sensible. Results reported are from the inference over the validation set, composed of inline sections from the Kerry3D survey.

The first experiment intends to evaluate the regularization factor of the autoencoder R_{AE} and centroids $R_{\mathcal{C}}$. The regularization is used to reduce the model overfitting. We train our model to achieve the bit rate target of $r_t = 1.0$ bpv. We set $n_B = 15$, $n_F = 128$, $k_a = 5$, $k_b = 3$, $k_c = 5$, $C = 64$, $L = 50$ and $CIR = [-10, 10)$. The CIR parameter is the initial range for the centroids initialization. L centroids are randomly sampled from CIR. From the results presented in Table 4, we notice that smaller $R_{\mathcal{C}}$ values favor the compression rate, with a slightly better reconstruction quality. The R_{AE}, however does not have a meaningful impact. Using $R_{\mathcal{C}} = 0.01$ and $R_{AE} = 0.005$ the method achieves a PSNR of 38.27 dB using 1.09 bpv.

Table 3 – Parameters evaluated for the 2DSC method.

Parameters	Description	Values
R_{AE}	Autoencoder regularization factor	0.05, 0.005, 0.0005
$R_{\mathcal{C}}$	Centroids regularization factor	1.0, 0.1, 0.01
n_B	Number of residual blocks	15, 18
n_F	Number of filters in the convolutional layers	64, 128, 192
k_a	Kernel size	5, 7
k_b	Kernel size	3, 5
k_c	Kernel size	5, 7
CIR	Centroids initial range	[-1, 1), [-5, 5), [-10, 10)
C	Number of channels in the latent space	8, 16, 32, 64, 128
L	Centroids set size	6, 12, 24, 50, 100

Source: created by the author.

Table 4 – Regularization parameter evaluation for 2DSC method. Cells with darker colors meaning a better result.

$R_{\mathcal{C}}$	R_{AE}	PSNR	bpv
0.01	0.0005	37.65	1.01
0.01	**0.005**	**38.27**	**1.09**
0.01	0.05	38.22	1.07
0.1	0.0005	38.76	1.23
0.1	0.005	37.42	1.24
0.1	0.05	38.24	1.14
1.0	0.0005	37.47	1.42
1.0	0.005	34.88	1.32
1.0	0.05	37.23	1.62

Source: created by the author.

The second experiment aims to evaluate the number of residual blocks n_B, the number of filters in the convolutional layers n_F and the kernel sizes k_a, k_b and k_c that describes our 2DSC architecture. Table 5 shows the results for the architecture setting. Notice that better results are provided by small kernel sizes. It implies that the method benefits from highly local features. The number of residual blocks did not have a meaningful impact on the metrics, just like the width of the network. It indicates that the network is already above the capacity necessary for the problem. The best overall result is found with $n_B = 18$, $n_F = 64$, $k_a = 5$, $k_b = 3$ and $k_c = 5$, with a PSNR of 38.97 dB and 1.11 bpv.

With the network parameters defined, the next step is to exploit the impact of the centroids initialization. It is important since a bad initialization can hinder convergence and lead to an arbitrary centroid set, in which some of them can become equal. The CIR parameter is related to the range in which the centroids set is initialized. A set of L random values is chosen in the range of CIR. Table 6 shows the CIR parameter comparison. It is noticeable that as the interval increases, the distortion becomes small.

Table 5 – Architecture parameter evaluation for 2DSC method. The grayscale indicates the top 3 values that provided the best overall result, with darker colors meaning a better result.

n_B	n_F	k_a	k_b	k_c	PSNR	bpv	n_B	n_F	k_a	k_b	k_c	PSNR	bpv
15	64	5	3	5	38.02	1.04	**18**	**64**	**5**	**3**	**5**	**38.97**	**1.11**
15	64	5	3	7	37.26	1.03	18	64	5	3	7	38.37	1.05
15	64	5	5	5	37.22	1.15	18	64	5	5	5	37.48	1.17
15	64	5	5	7	37.70	1.09	18	64	5	5	7	37.75	1.20
15	64	7	3	5	37.80	1.08	18	64	7	3	5	37.70	1.09
15	64	7	3	7	37.18	1.02	18	64	7	3	7	37.40	1.07
15	64	7	5	5	37.65	1.10	18	64	7	5	5	36.80	1.09
15	64	7	5	7	36.39	1.04	18	64	7	5	7	37.58	1.15
15	128	5	3	5	38.27	1.09	18	128	5	3	5	37.65	1.07
15	128	5	3	7	38.00	1.13	18	128	5	3	7	37.15	1.14
15	128	5	5	5	38.35	1.12	18	128	5	5	5	37.17	1.08
15	128	5	5	7	37.93	1.10	18	128	5	5	7	37.49	1.09
15	128	7	3	5	37.88	1.09	18	128	7	3	5	37.34	1.12
15	128	7	3	7	38.06	1.11	18	128	7	3	7	37.86	1.09
15	128	7	5	5	37.49	1.15	18	128	7	5	5	37.16	1.10
15	128	7	5	7	37.37	1.10	18	128	7	5	7	36.33	1.07
15	192	5	3	5	38.47	1.12	18	192	5	3	5	38.73	1.12
15	192	5	3	7	38.02	1.10	18	192	5	3	7	37.63	1.11
15	192	5	5	5	37.17	1.10	18	192	5	5	5	37.41	1.07
15	192	5	5	7	37.92	1.06	18	192	5	5	7	36.54	1.09
15	192	7	3	5	38.16	1.09	18	192	7	3	5	37.55	1.06
15	192	7	3	7	37.01	1.10	18	192	7	3	7	37.82	1.10
15	192	7	5	5	37.31	1.08	18	192	7	5	5	36.63	0.98
15	192	7	5	7	36.24	1.11	18	192	7	5	7	37.34	1.07

Source: created by the author.

The initialization in a small interval yields closer centroids, and some of them tend to converge to the same point at the final training steps. In contrast, large intervals do not provide proportional improvement, and we set $CIR = [-10, 10)$ as the range of choice.

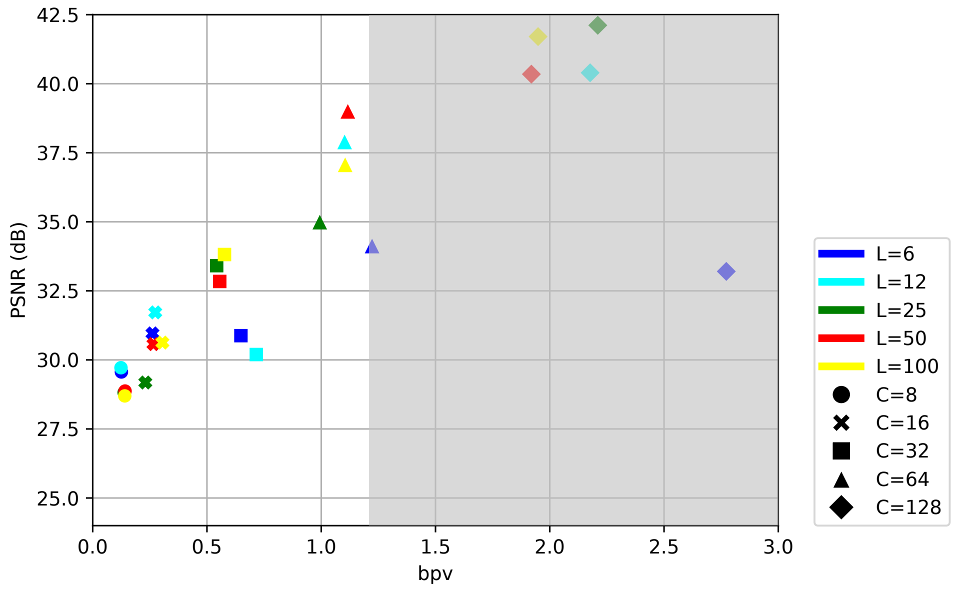
Table 6 – Centroids initialization range evaluation for 2DSC method. Small ranges lead to closer centroids.

CIR	PSNR	bpv
$[-1, 1)$	35.77	1.07
$[-5, 5)$	38.13	1.08
[-10,10)	**38.97**	**1.11**

Source: created by the author.

The next experiment aims to evaluate the impact of the number of centroids L and the number of channels C in the latent space that is related to the bit rate obtained. Figure 24 shows the curve PSNR × bpv for all combinations of L and C, represented by color and markers, respectively. We can see that the number of channels C controls the bit rate. We can notice that, as the number of channels increases, the bit rate also increases by the same factor, which is expected as they are directly related. The number of centroids L has no significant impact on the bit rate, but the PSNR benefits from a higher number of centroids. Then, the increment on the number of centroids only translates into an improvement of quality when paired with a high number of channels. For a bit rate target $r_t = 1.0$ bpv, the best performance is reached by using $C = 64$ and $L = 50$, with a bit rate of 1.11 bpv and a PSNR of 38.97 dB. We define $C = 64$ and $L = 50$ for next experiments since we are using $r_t = 1.0$ bpv. It is important to emphasize that for higher or lower bit rates a different parameter set can be more suitable. For example, we cannot reach a bit rate of 0.1 bpv using 64 channels in the latent space. In this case, $L = 12$ and $C = 8$ would provide better results.

Figure 24 – Evaluation of the number of centroids L and number of channels C for 2DSC method represented by color and markers, respectively. The gray area is the region above the margin threshold (1.2 bpv) of acceptable bpv (Eq. 4.1). Better results are closest to the top-left corner.

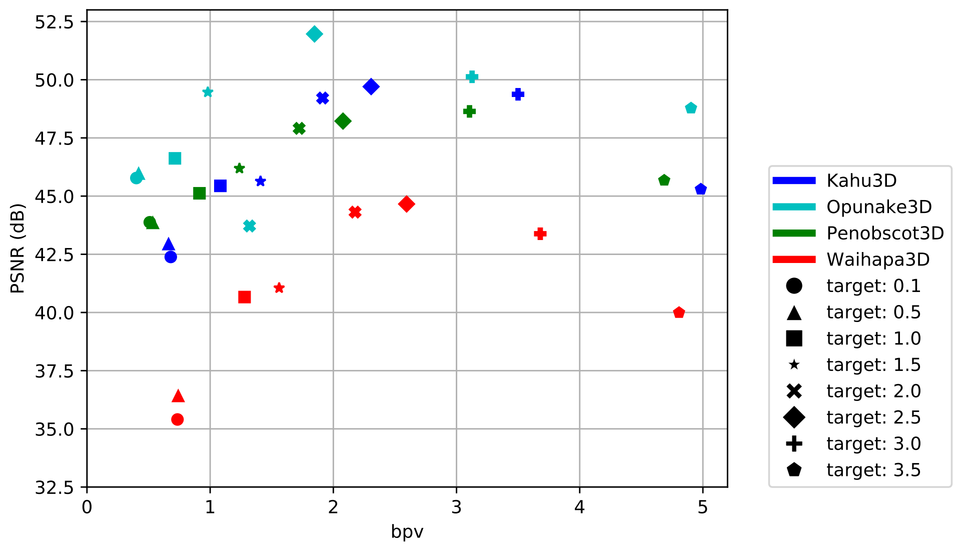


Source: created by the author.

4.4.1.2 *Model Evaluation on Testing Surveys*

With the best parameter setting determined, we can evaluate the testing surveys. We trained different models for 8 different bit rates target $r_t = 0.1$, 0.5, 1.0, 1.5, 2.0, 2.5, 3.0 and 3.5 bpv, aiming to evaluate the method performance. Figure 25 shows the results of the 2DSC method for testing sets Kahu3D, Opunake3D, Penobscot3D and Waihapa3D. We can see that using the parameters found in the previous subsection the method reached bit rates higher than 0.4 bpv. This is a limitation from this parameter set, as they were adjusted to fit the bit rate target $r_t = 1.0$ bpv. By using fewer feature maps in the latent space it is possible to achieve smaller bit rates. We can notice that in general, the method has difficulty adjusting the bit rate target. As r_t increases, the difference between the bit rate targets and obtained is enhanced. It probably occurs due to the rate-distortion trade-off. For a bit rate target higher than 2.5 bpv, the method attempts to adjust the target even if it reflects in a distortion loss. It can be attenuated by using a small value for β. In this sense, the method benefits the distortion instead of the bit rate, and the curve becomes stable. However, for a very small β, it is not possible to control the bit rate and the method will reach the highest possible PSNR value.

Figure 25 – PSNR × bpv curve for 2DSC method for testing datasets over different bit rates.



Source: created by the author.

The best performance is reached for surveys with fewer details, i.e. with small high frequency presence. The Kahu3D, Opunake3D and Penobscot3D achieved higher PSNR values even using a bit rate next to 0.5 bpv. It is due to the presence of homogeneous regions in these data. In particular, for Opunake3D dataset, there exist a few regions of high frequencies, reaching a PSNR of 46.05 dB using 0.45 bpv. In contrast, the Waihapa3D has noisy regions that difficult the compression. Even under these conditions, the method achieved a PSNR higher than 40 dB using 1.12 bpv for this survey.

The second experiment intends to evaluate the compression capabilities of the method under different conditions. With this goal, we propose to use the leave-one-in protocol (TRIPPA et al., 2015). The leave-one-in is the opposite of the leave-one-out protocol, in which one dataset is used for training and the remaining sets are used for testing. To this end, we select the first 10% slices from the training data for validation.

From Table 7, we can notice that the generalization capability of our 2DSC method is directly conditioned to the training set choice. Homogeneous surveys (Opunake3D and Penobscot3D), with a predominance of low frequencies, do not provide enough information to reach good performance, leading to network overfitting. Kahu3D and Parihaka3D, Netherlands F3-Block and Waihapa3D surveys have both high and low frequencies, achieving reasonable results for all testing sets. The Waihapa3D achieves good PSNR values, but the bit rate is considerably greater than using the other surveys. The highest loss often occurs for noisy data. Very difficult surveys (Kerry3D and Poseidon3D) are good training data, as they can achieve good performance even for very simple volumes. However, they are difficult to be compressed by using other surveys as the training set. In general, the performance is related to the presence of high and low frequencies in the training set. Balanced training sets can generalize in different situations. As the experiments show, higher PSNR values are reached when trained and tested on the same survey. It is expected, as specialized networks will adjust the weights to the training data. An exception is found on the Opunake3D. For this data, the best result uses Parihaka3D as the training set. We believe that it occurs because there exist only a few regions of high-frequency in the Opunake3D data. The number of samples with homogeneous regions is greater than with noisy regions and the network adapts to learn the predominant information. In general, these results indicate that our 2DSC method achieves good performance even under different conditions. In a generalist situation, the method overall performance is good.

Table 7 – Leave-one-in protocol results for the 2DSC. In bold the best result for each test survey. Painted cells indicate the top 3 best results for each testing survey, with darker colors meaning a better result. We consider only results below the margin threshold (1.2 bpv) of acceptable bpv (Eq. 4.1).

Test / Train	Opunake3D	Penobscot3D	Kahu3D	Parihaka3D	Waihapa3D	N. F3-Block	Kerry3D	Poseidon3D
Opunake3D	50.39/0.99	28.63/1.14	20.42/1.29	25.04/1.08	11.24/1.72	17.34/1.15	9.45/1.87	9.20/2.00
Penobscot3D	22.95/1.04	48.07/1.22	31.66/1.26	15.80/1.13	37.51/1.43	13.97/1.15	12.50/1.54	8.37/1.59
Kahu3D	47.27/1.08	45.78/1.10	**47.68/1.00**	43.72/1.06	**46.05/1.14**	36.21/1.17	32.43/1.29	29.64/1.50
Parihaka3D	**50.44/0.77**	**46.00/1.00**	39.21/1.02	**50.02/1.00**	40.26/1.34	39.07/1.07	35.83/1.48	32.01/1.33
Waihapa3D	42.56/1.88	43.29/1.77	46.52/1.39	38.80/1.68	46.59/1.40	34.00/1.69	33.34/1.5	28.77/1.85
N. F3-Block	46.90/0.78	44.90/0.86	35.90/0.91	45.75/0.87	30.15/1.02	**40.30/1.01**	33.22/1.19	33.06/1.27
Kerry3D	40.43/1.12	42.17/1.16	46.34/1.03	38.95/1.03	44.84/1.15	35.00/1.06	39.98/1.23	33.84/1.32
Poseidon3D	45.12/0.86	43.19/0.78	40.81/0.69	44.26/0.71	40.74/0.88	**38.98/0.96**	**37.00/0.96**	38.40/0.76

Source: created by the author.

4.4.2 2D-based Seismic Data Compression using Multi-resolution (2DSC-MR)

In Section 4.4.2.1 we present the experiments performed to find the best parameters setting and in Section 4.4.2.2 we present the method performance evaluation on testing surveys.

4.4.2.1 *Parameters Setting*

To evaluate the 2DSC-MR method we perform a sequence of experiments to find the best parameters setting. Due to the similarity to 2DSC method, we fixed the parameters that are less sensible to architecture changes. We set the centroids regularization factor $R_{\mathfrak{C}} = 0.01$, autoencoder regularization factor $R_{AE} = 0.005$, kernel sizes $k_a = 5$, $k_b = 3$, $k_c = 5$ and the range in which the centroids set is initialized $CIR = [-10, 10)$ as the same discovered in the previous experiments for all tests. We evaluate the parameters from Table 8 for this method.

Table 8 – Parameters evaluated for the 2DSC-MR method.

Parameters	Description	Values
n_B	Number of residual blocks	3, 6, 9, 12, 15, 18
n_F	Number of filter in the convolutional layers	64, 128, 192, 256
C	Number of channels in the latent space	8, 16, 32, 64, 128
L	Centroids set size	6, 12, 24, 50, 100

Source: created by the author.

The 2DSC method is composed of a reduced number of layers if compared to the 2DSC-MR method. From the results shown in Table 3, it is reasonable to suppose that the multi-resolution model also does not need very deeper architectures to extract the seismic information. Our first experiment aims to evaluate the optimal number for the network depth, by varying the number of residual blocks n_B. We fixed the number of filters in the convolutional layers $n_F = 64$, the number of channels in the latent space $C = 64$ and the number of centroids $L = 50$ for this experiment. We evaluate n_B for 3, 6, 9, 12, 15 and 18 residual blocks. The results are shown in Table 9. A higher number of residual blocks do not improve the performance of the method, and the best result is reached for $n_B = 3$. Note that with $n_B = 18$ we have a good result as well, with the best bpv and reasonably good PSNR.

Table 9 – Number of residual blocks evaluation for 2DSC-MR method.

n_B	PSNR	bpv
3	**38.26**	**1.07**
6	37.44	1.03
9	38.14	1.12
12	37.71	1.06
15	37.45	1.09
18	37.83	1.02

Source: created by the author.

A second experiment is performed aiming to evaluate the parameter n_F related to the width of the network. In comparison to the 2DSC, since the number of residual blocks has been reduced, it is expected that the optimal number of filters n_F changes as well. Table 10 shows the results for 64, 128, 192 and 256 filters. The best width is $n_F = 256$, yielding a slightly better bpv and PSNR than $n_F = 64$, but the computational cost of 64 filters is much lower than 256. A model with 256 filters requires approximately 17 hours to train, whereas with 64 filters it requires 12 hours. The improvement is very slight compared to the increase in computational cost. For the next experiments, we set $n_F = 64$, as it provides good results at a lower cost.

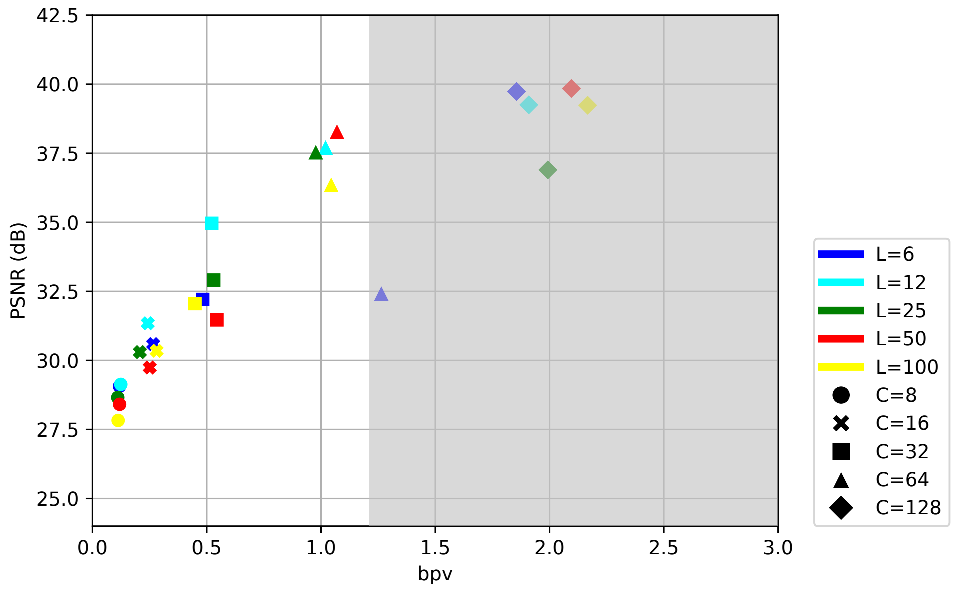
Table 10 – Results for the evaluation of the number of filters in the convolutional layers for 2DSC-MR method.

n_F	PSNR	bpv
64	38.26	1.07
128	37.06	1.18
192	37.69	1.04
256	**38.33**	**1.05**

Source: created by the author.

The next experiment intends to investigate how the method behaves for a different number of centroids L and channels C in the latent space. Figure 26 exhibits the PSNR and bpv results for all combinations of L and C for a bit rate target $r_t = 1.0$ bpv. As in the 2DSC method, the number of channels controls the overall result. As it increases, the bit rate also increases by the same factor. For small values of C, better results are reached using 12 centroids. For values higher than 1.0 bpv, it is better to use 50 centroids. The increment on the number of centroids only translates into an improvement of quality when paired with a high number of channels. The best overall performance is reached using $C = 64$ and $L = 50$, with a PSNR of 38.26 dB using 1.07 bpv.

Figure 26 – Evaluation of the number of centroids L and number of channels C for 2DSC-MR method represented by color and markers, respectively. The gray area is the region above the margin threshold (1.2 bpv) of acceptable bpv (Eq. 4.1). Better results are closest to the top-left corner.

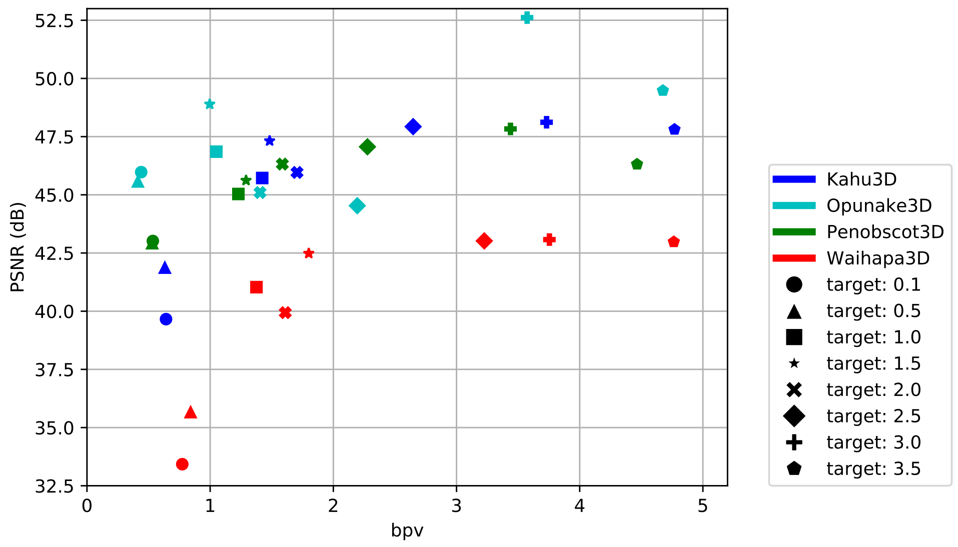


Source: created by the author.

4.4.2.2 *Model Evaluation on Testing Surveys*

Initially, we intend to evaluate the testing surveys over different bit rates with the same parameters found in the previous section. To this end, we trained 8 different models with variable bit rates target $r_t = 0.1, 0.5, 1.0, 1.5, 2.0, 2.5, 3.0$ and 3.5 bpv and inferred using the testing surveys. Results can be viewed in Figure 27. As in the 2DSC approach, the multi-scale also cannot achieve very low bit rates, since the parameters were adjusted to fit the bit rate target of 1.0 bpv. The performance for all testing sets is similar, with the PSNR increasing until a bit rate of 2.5 bpv, and remaining considerably stable from this point. The exception is Opunake3D. The increase in the bit rate target does not necessarily imply an increase in PSNR. For this data, the method achieves good quality using fewer bits.

Figure 27 – PSNR × bpv curve for 2DSC-MR method for testing datasets over different bit rates.



Source: created by the author.

To evaluate our model on different conditions, we use the leave-one-in protocol. As in the 2DSC method, we train models using only one survey and perform inference using the others. We select the first 10% slices from the training data for validation. Table 11 shows the results of the leave-one-in protocol. We can notice that the method performance is highly dependent on the training set. The 2DSC-MR approach is beneficial for lower frequency training sets, mainly for Opunake3D and Penobscot3D. In contrast to Table 7, the 2DSC-MR method reached higher PSNR values by using these datasets as training. We believe that it occurs due to the downscaling and upscaling blocks. The scaling highlights the differences of frequencies in these data, reducing the model overfitting. However, for noisy testing surveys, such as Kerry3D and Poseidon3D, the PSNR achieved in both approaches is not enough to consider that the relevant original information was preserved.

In general, better results are achieved by training on surveys with the same difficulty level of the testing set.

Table 11 – Leave-one-in protocol results for the 2DSC-MR. Painted cells indicate the top 3 best results for each testing survey, with darker colors meaning a better result. We consider only results below the margin threshold (1.2 bpv) of acceptable bpv (Eq. 4.1).

Train \ Test	Opunake3D	Penobscot3D	Kahu3D	Parihaka3D	Waihapa3D	N. F3-Block	Kerry3D	Poseidon3D
Opunake3D	51.50/1.01	42.02/1.18	36.47/1.41	39.31/1.00	27.85/1.77	31.24/1.16	25.60/2.42	23.25/2.48
Penobscot3D	45.94/1.23	48.79/1.23	45.12/1.36	21.27/1.06	36.54/1.26	18.34/1.03	16.71/1.04	18.57/0.54
Kahu3D	46.55/1.18	45.47/1.13	46.01/1.00	44.49/1.04	**48.84/1.10**	24.80/0.89	16.64/0.97	14.92/0.97
Parihaka3D	**51.90/0.95**	**45.47/1.07**	**46.80/1.08**	**49.38/0.99**	39.61/1.23	38.82/1.14	23.60/1.05	16.34/1.12
Waihapa3D	38.47/1.53	39.53/1.40	44.84/1.46	37.64/1.27	45.93/1.45	33.33/1.39	33.00/1.33	28.16/1.55
N. F3-Block	47.04/0.83	44.90/0.92	37.75/1.04	46.06/0.93	30.38/1.21	**39.76/1.02**	31.95/1.29	32.28/1.33
Kerry3D	42.49/1.03	43.03/1.06	45.83/1.04	40.91/1.20	45.02/1.22	36.45/1.24	38.04/1.21	31.76/1.27
Poseidon3D	41.83/0.59	38.67/0.56	39.76/0.68	40.63/0.81	40.64/1.14	37.26/1.07	**36.23/0.98**	**37.93/0.95**

Source: created by the author.

4.5 3D POST-STACK COMPRESSION

The 3D post-stack compression is different from the 2D approaches and requires an evaluation of its training parameters. We train our models with a crop size of $160 \times 160 \times d$. For all tests we fixed the regularization parameters for the centroids $R_{\mathfrak{C}} = 0.01$, and for the autoencoder $R_{AE} = 0.005$, and the kernel sizes $k_a = 5$, $k_b = 3$, $k_c = 5$ as the same from the 2DSC. They were fixed as they did not provide a meaningful impact on the 2DSC model. Moreover, slight changes were made in the 2DSC model to allow 3D compressions and we assume that these parameters remain suitable for the next models.

4.5.1 **3D-based Seismic Data Compression (3DSC)**

In Section 4.5.1.1 we present the experiments performed to find the best parameters setting and in Section 4.5.1.2 we present the method performance evaluation on testing surveys.

4.5.1.1 *Parameters Setting*

To find the best parameters for the 3DSC method, a set of experiments was performed. As in the 2DSC approach, a grid search over the same parameters from Table 3 would require training more than 30000 models. The simplest model requires about 8 hours to be trained. Consequently, we reduce the parameter space by fixing some of them and evaluating the most important, according to Table 12. In this sense, we define the best value for the parameters set throughout a sequence of experiments. For all tests we fixed the parameter $CIR = [-10, 10)$ as the same from the 2DSC since this method does not provide any change in the quantization step. We train our models with an initial learning rate of 8×10^{-5}.

Table 12 – Parameters evaluated for 3DSC method.

Parameters	Description	Values
n_B	Number of residual blocks	3, 6, 9, 12, 15, 18
n_F	Number of filter in the convolutional layers	64, 128, 192
C	Number of channels in the latent space	8, 16, 32, 64, 128
L	Centroids set size	6, 12, 24, 50, 100
d	Input depth size	4, 8, 16

Source: created by the author.

Initially, we want to evaluate the parameters related to the architecture of the 3DSC approach. Our first experiment intends to determine the deep and width of the encoder and decoder parts of the network. In this setting, we perform experiments with 3, 6, 9, 12, 15 and 18 residual blocks and 64, 128, and 192 filters in the convolutional layers. We fixed $L = 50$ centroids, $C = 64$ channels in the latent space and the input depth size as $d = 4$. From Table 13, we can notice that the main difference comes from the use of more than 64 filters in the convolutional layers. From 64 to 128 filters, the improvement is considerable in most of the cases. However, from 128 to 192, the same not occur. The number of residual blocks has no meaningful impact on the results. It implies that the 3DSC method also does not need a very deep network to compress the seismic volume. Training deeper networks demand bigger training sets. Even with a training set composed of a small number of samples, with only 3 residual blocks, the 3DSC method reaches good results. In this sense, we set $n_F = 128$ an $n_B = 3$ in next experiments.

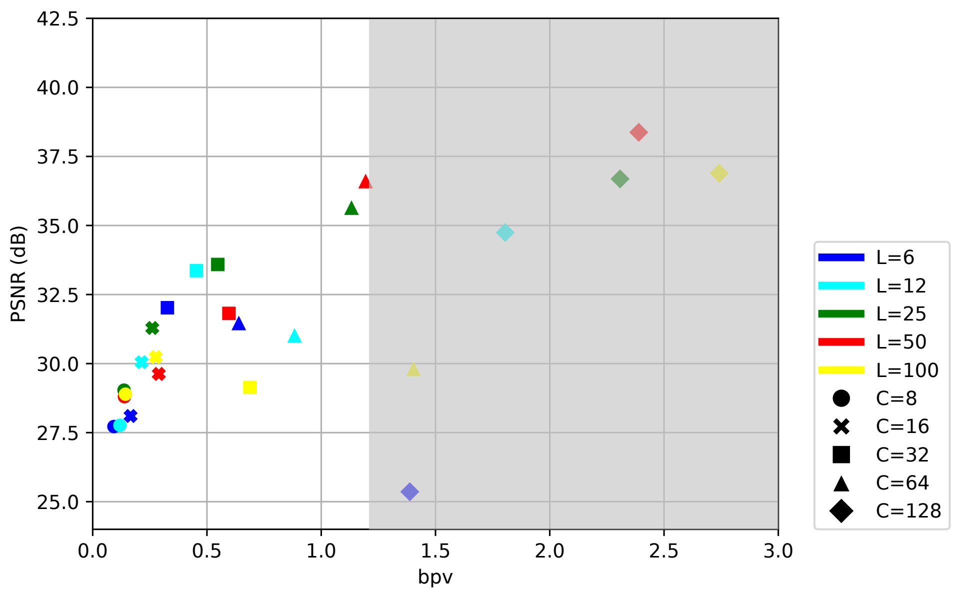
Table 13 – Architecture parameter evaluation for 3DSC method. The grayscale indicates the values that achieved good results.

n_B	n_F	PSNR	bvp	n_B	n_F	PSNR	bvp
3	64	32.93	1.20	12	64	34.52	1.18
3	**128**	**36.59**	**1.19**	12	128	34.59	1.25
3	192	35.00	1.18	12	192	33.70	1.25
6	64	33.04	1.27	15	64	33.20	1.32
6	128	36.35	1.21	15	128	36.41	1.20
6	192	36.11	1.18	15	192	35.20	1.22
9	64	32.47	1.28	18	64	34.66	1.24
9	128	36.25	1.18	18	128	35.43	1.23
9	192	35.38	1.21	18	192	36.15	1.20

Source: created by the author.

The next experiment aims to find the best number of centroids L and the number of channels C of the latent space. We perform experiments using $L = 6$, 12, 25, 50 and 100, and using $C = 8$, 16, 32, 64, 128. Figure 28 shows the results for all combinations of L and C, represented using colors and markers, respectively. The bit rate is highly dependent on the number of channels so that both increases by the same factor. However, the PSNR has no significant increase by using a high number of channels. This suggests that our 3DSC model performs better for lower bit rates. In general, as the number of centroids increases, it also increases the PSNR and bit rate until they become stable. The improvement in increasing the number of centroids is highlighted using a higher number of channels in the latent space. The better overall result is found with $L = 50$ and $C = 64$, with a PSNR of 36.59 dB and 1.19 bpv.

Figure 28 – Evaluation of the number of centroids L and number of channels C for 3DSC method represented by color and markers, respectively. The gray area is the region above the margin threshold (1.2 bpv) of acceptable bpv (Eq. 4.1). Better results are closest to the top-left corner.



Source: created by the author.

To evaluate how the depth of the input volume affects the method performance we perform experiments by using $d = 4$, 8 and 16 slices. Table 14 shows that as the depth increases, the bit rate decreases considerably. It is expected, as we are mapping more slices to a latent representation equivalent to the encoding of only one. However, the PSNR does not decrease at the same factor, and reasonable results are achieved by using more slices. The main loss occurs in the Waihapa3D survey, due to the presence of high frequencies, but even on these conditions, the difference from 4 and 16 slices is about only 4.22 dB. We set $d = 4$ for the next experiments since it provides the best overall result.

Table 14 – Input depth size evaluation for 3DSC method.

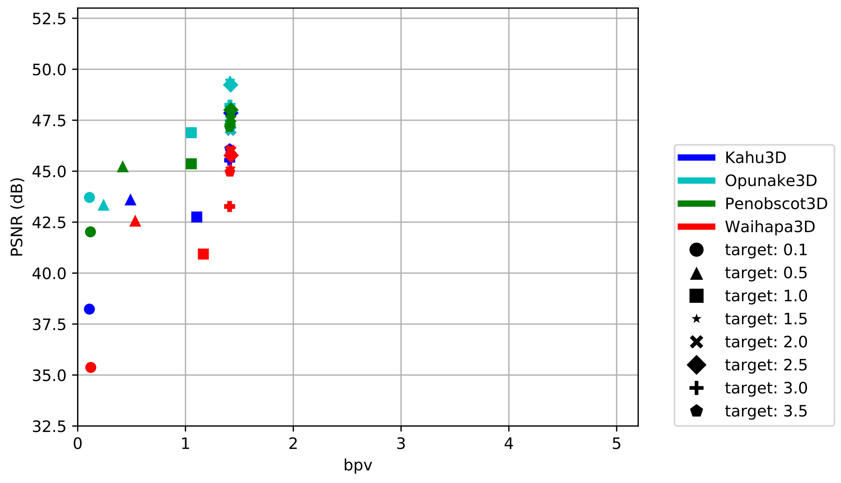
PSNR/bpv	Kahu3D	Opunake3D	Penobscot3D	Waihapa3D
$d = 4$	**42.75/1.10**	**46.89/1.05**	**45.36/1.05**	**40.93/1.16**
$d = 8$	41.77/0.70	43.59/0.71	44.78/0.71	40.00/0.70
$d = 16$	38.43/0.35	46.43/0.35	42.54/0.35	36.71/0.35

Source: created by the author.

4.5.1.2 *Model Evaluation on Testing Surveys*

With the parameters from the previous section, we intend to evaluate our model over different bit rates. We train 8 models with bit rate target $r_t = 0.1$, 0.5, 1.0, 1.5, 2.0, 2.5, 3.0 and 3.5 bpv. According to the results from Figure 29, the 3DSC method can achieve very low bit rates for all testing surveys. As the target increases, the PSNR also increases until becomes stable with a bit rate of 1.4 bpv. It shows that 3DSC deals better with smaller bit rates, but the performance does not worsen as they increase. In general, for all surveys, the method achieves reasonable results, even with a bit rate of 0.1 bpv.

Figure 29 – PSNR × bpv curve for 3DSC method for testing datasets over different bit rates.



Source: created by the author.

The next experiment intends to evaluate the 3DSC model over different surveys. We use the same parameters found in the previous section. The leave-one-in protocol is employed, such that all surveys are used individually for training, and the remaining ones are used for testing. We select the first 10% samples from the training set for validation. From the analysis of Table 15 it is noticeable that, in most of the cases, the method reached bit rates higher than the considered threshold from Equation 4.1. To achieve a bit rate of 1.0 bpv in practice, the model needs to set the bit rate target smaller than 1.0 bpv. In this sense, we evaluate these results taking into account only the PSNR reached. We can notice that the 3DSC method can achieve PSNR values higher than 40 dB for most of the surveys. The exception is the Kerry3D, Netherlands F3-Block and the Poseidon3D. The performance of the 3DSC approach for higher frequency data is not enough to consider, in a generalist approach, that the underlying structures from these data were preserved.

4.5.2 **3D-based Seismic Data Compression using Vector Quantization (3DSC-VQ)**

In Section 4.5.2.1 we present the experiments performed to find the best parameters setting and in Section 4.5.2.2 we present the method performance evaluation on testing

Table 15 – Leave-one-in protocol results for the 3DSC method. Painted cells indicate the top 3 best results, with darker colors meaning a better result.

Test / Train	Opunake3D	Penobscot3D	Kahu3D	Parihaka3D	Waihapa3D	N. F3-Block	Kerry3D	Poseidon3D
Opunake3D	41.59/1.20	41.11/1.25	38.22/1.40	38.87/1.27	33.30/1.65	33.84/1.30	28.70/1.75	26.67/1.83
Penobscot3D	36.40/1.37	35.69/1.30	37.59/1.37	28.77/1.40	39.50/1.36	27.98/1.39	25.72/1.43	23.05/1.50
Kahu3D	46.13/1.28	40.07/1.24	41.19/1.16	40.84/1.26	43.79/1.24	34.34/1.25	30.46/1.34	27.87/1.39
Parihaka3D	**51.69/1.21**	**47.15/1.20**	**46.86/1.25**	**49.04/1.10**	44.11/1.29	**37.89/1.22**	34.29/1.32	30.76/1.33
Waihapa3D	43.33/1.37	43.40/1.36	40.00/1.36	38.88/1.37	39.78/1.29	33.40/1.36	30.55/1.36	27.88/1.39
N. F3-Block	49.56/1.36	45.66/1.33	43.71/1.36	46.28/1.35	**44.20/1.33**	36.31/1.24	33.90/1.35	31.29/1.34
Kerry3D	43.01/1.35	42.94/1.31	43.58/1.34	39.89/1.33	42.14/1.30	34.73/1.30	33.00/1.17	30.50/1.34
Poseidon3D	46.41/1.31	43.18/1.30	43.27/1.31	43.28/1.31	42.89/1.30	36.13/1.29	**36.13/1.29**	**42.24/1.18**

Source: created by the author.

surveys.

4.5.2.1 *Parameters Setting*

Although the 3DSC-VQ method is similar to the 3DSC approach, the former is not a straightforward generalization of the latter. Analogously to the 3DSC, the vectorial approach requires to modify the depth stride from their convolutional layers in the autoencoder. In this sense, some parameters from the previous method may not fit in this case, and we perform experiments varying the parameters according to Table 16.

Table 16 – Parameters evaluated for 3DSC-VQ method.

Parameters	Description	Values
CIR	Centroids initial range	$[-1, 1), [-5, 5), [-10, 10)$
n_B	Number of residual blocks	3, 6, 9, 12
n_F	Number of filter in the convolutional layers	64, 128
C	Number of channels in the latent space	8, 16, 32, 64, 128
L	Centroids set size	6, 12, 24, 50, 100
d	Input depth size	4, 8, 16

Source: created by the author.

The main difference is related to the vectorial centroids, in which their initialization must ensure that they become different during the training. The method does not converge with a simple random sampling from the CIR for each centroid entry. We propose to initialize by sampling u evenly spaced values from CIR. We perform a combination with replacement over the u values to generate a set of centroids of length d. From this set, we randomly sample the L initial centroids.

The training set also needed some changes. Different from previous methods, the 3DSC-VQ does not converge with the Poseidon3D as part of the training set. To evidentiate this, we perform tests combining the training surveys. We fixed the number of residual blocks as $n_B = 1$, number of channels in the convolutional layers $n_F = 64$, number of centroids $L = 50$, number of channels in latent space $C = 64$, input depth dimension $d = 4$ and $u = 6$, and $CIR = [-1, 1)$. We train our models with an initial learning rate of 8×10^{-4}. Table 17 shows that the combinations of surveys and the PSNR and bit rates obtained for each one. We can see that for any combination in which the Poseidon3D is part of the training set, the method does not converge. In this sense, the next experiments were performed by using only the Parihaka3D and Netherlands F3-Block as training surveys.

Table 17 – Training survey combination for 3DSC-VQ method.

Dataset	PSNR	bpv
Parihaka3D + N. F3-Block	34.79	0.84
Parihaka3D + Poseidon3D	25.33	0.97
N. F3-Block + Poseidon3D	25.33	0.86
Parihaka3D + N. F3-Block + Poseidon3D	25.29	0.68

Source: created by the author.

In our first experiment, we intend to evaluate the centroids initialization. We evaluate the ranges $[-1, 1)$, $[-5, 5)$ and $[-10, 10)$ for CIR. We can notice from the results of Table 18 that different from previous approaches, smaller ranges are beneficial for the 3DSC-VQ method. It occurs because the initialization ensures that the centroids will be different, but for larger ranges they can become too different, making convergence difficult.

Table 18 – Results from varying ranges of CIR for 3DSC-VQ method.

CIR	PSNR	bpv
[-1,1)	**34.79**	**0.84**
$[-5, 5)$	34.13	0.93
$[-10, 10)$	33.74	0.99

Source: created by the author.

To adjust the architecture parameters, we perform the same experiments from the 3DSC approach. Initially, the parameters from the previous experiments were fixed. We evaluate the number of residual blocks n_B for the values 3, 6, 9 and 12 and the number of filters of the convolutional layers n_F using 64 and 128. Results are shown in Table 19. Considering the number of filters n_F, there is no considerable improvement from 64 to 128 filters. The number of residual blocks n_B yields a meaningful increase in the bit rate that is not reflected in the PSNR. The best overall performance is reached by using 3 blocks and 64 filters. In this sense, we set $n_B = 3$ and $n_F = 64$ for next experiments.

The next experiment intends to evaluate how the number of centroids L and the number of channels in the latent space C impact the results. Figure 30 shows the curve PSNR $\times$ bpv for all combinations of L and C, represented by color and markers, respectively. We can notice that for small values of C, as the number of centroids increases, the bit rate also increases, while the PSNR is slightly affected. This shows that the 3DSC-VQ method performs better for lower bit rates. As in the previous methods, the

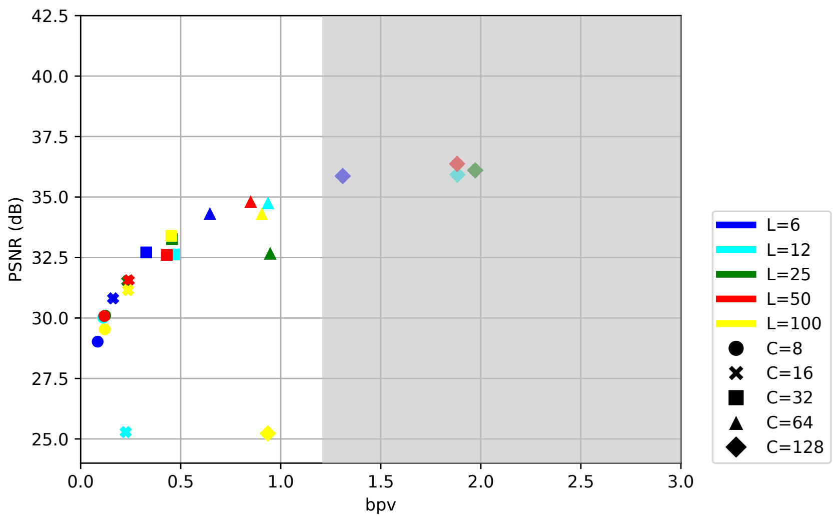
Table 19 – Architecture parameter evaluation for 3DSC-VQ method. The grayscale indicates the values that achieved good results.

n_B	n_F	PSNR	bpv
3	**64**	**34.79**	**0.84**
3	128	34.23	0.90
6	64	34.78	0.92
6	128	34.42	0.97
9	64	32.84	0.92
9	128	34.35	0.97
12	64	34.71	0.96
12	128	34.67	0.98

Source: created by the author.

number of channels also implies in the bit rate obtained, proportionally, by a factor of 2. The highest PSNR until a bit rate of 1.2 bpv is reached using $L = 50$ and $C = 64$, with 34.79 dB and 0.84 bpv.

Figure 30 – Evaluation of the number of centroids L and number of channels C for 3DSC-VQ method represented by color and markers, respectively. The gray area is the region above the margin threshold (1.2 bpv) of acceptable bpv (Eq. 4.1). Better results are closest to the top-left corner.



Source: created by the author.

Aiming to evaluate the performance of the method over different input depth sizes, we perform experiments by using $d = 4$, 8 and 16 slices. From Table 20 we can notice that the increase in the number of slices affects negatively the quality obtained. We believe that as the depth increases, more centroids are needed to deal with the number of combinations of possible vectors. The bit rate is slightly affected, and there is no benefit in using a depth size with more slices than 4. Thus, we set $d = 4$ for the next experiments.

Table 20 – Input depth size evaluation for 3DSC-VQ method.

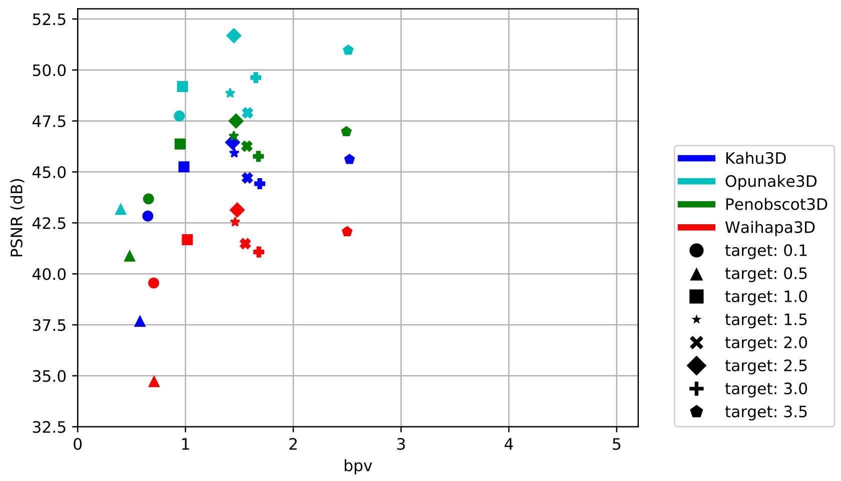
PSNR/bpv	Kahu3D	Opunake3D	Penobscot3D	Waihapa3D
$d = 4$	**45.25/0.98**	**49.19/0.97**	**46.37/0.95**	**41.68/1.02**
$d = 8$	42.65/0.92	48.74/0.94	44.90/0.95	39.36/0.94
$d = 16$	38.37/0.95	44.14/0.78	41.77/0.79	35.77/0.88

Source: created by the author.

4.5.2.2 *Model Evaluation on Testing Surveys*

We trained 8 different models for bit rates target $r_t = 0.1, 0.5, 1.0, 1.5, 2.0, 2.5, 3.0$ and 3.5 bpv, aiming to evaluate the performance of the method over different conditions. Figure 31 shows the 3DSC-VQ method performance for all testing sets. We can notice that this method is better for bit rates smaller than 1.5 bpv. For higher bit rates, the PSNR does not increase, becoming considerably stable from 1.5 bpv. The method benefits from the spatial correlation to achieve higher PSNR values at a small cost. The performance is better for low-frequency surveys, as Opunake3D and Penobscot3D. However, even for the Waihapa3D, it achieves PSNR values higher than 40 dB using 1.0 bpv. In general, this method does not fit the bit rate target, but the quality does not deteriorate as the target increases.

Figure 31 – PSNR × bpv curve for 3DSC-VQ method for testing datasets over different bit rates.



Source: created by the author.

As in the previous approaches, we evaluate our 3DSC-VQ method on different surveys by using the leave-one-in protocol. Similarly to the previous methods, we train one model for each survey and evaluate the remaining ones. We select the first 10% samples from the training set for validation. We use the same parameters from the previous section. Table 21 shows the results for all datasets. Different from 2D approaches, the dependency of the method over the training set is attenuated by using the 3DSC-VQ model. All surveys have good performance as a training set, but better results are still reached by training on similar surveys. In general, the method achieves PSNR values higher than 40 dB for most of them. However, for Kerry3D, Netherlands F3-Block and Poseidon3D the PSNR achieved is not enough to consider that the relevant original information was preserved. We believe that the third dimension introduces noises from these surveys that lead to an increase in the reconstruction error.

Table 21 – Leave-one-in protocol results for the 3DSC-VQ method. Painted cells indicate the top 3 best results, with darker colors meaning a better result. We consider only results below the margin threshold (1.2 bpv) of acceptable bpv (Eq. 4.1).

Test / Train	Opunake3D	Penobscot3D	Kahu3D	Parihaka3D	Waihapa3D	N. F3-Block	Kerry3D	Poseidon3D
Opunake3D	43.20/0.97	41.67/0.94	36.23/0.92	39.46/0.96	30.98/0.55	34.67/0.96	27.89/0.92	26.57/0.90
Penobscot3D	41.76/0.93	34.84/0.99	39.00/0.89	35.93/0.94	35.82/0.94	31.64/0.97	28.72/1.03	25.68/1.08
Kahu3D	45.52/0.86	44.96/0.90	41.82/0.97	40.67/0.96	42.87/0.97	34.82/0.97	31.88/1.22	27.75/1.22
Parihaka3D	**49.39/0.94**	**45.53/0.95**	44.39/0.97	**47.86/0.94**	40.81/0.98	**37.10/1.00**	32.76/1.03	29.72/1.03
Waihapa3D	40.54/0.97	40.71/0.94	40.80/0.96	36.55/1.00	39.17/0.98	32.52/1.02	30.15/1.10	27.23/1.10
N. F3-Block	45.45/1.07	43.23/0.98	43.43/0.98	44.38/0.96	39.77/0.98	36.00/0.99	33.13/0.98	29.80/1.01
Kerry3D	43.13/0.83	41.98/0.94	**45.36/1.00**	41.12/0.92	**43.34/0.96**	36.64/0.69	**35.81/0.96**	30.58/0.95
Poseidon3D	42.78/0.88	41.54/0.89	40.19/0.90	40.43/0.90	38.76/0.93	35.75/0.90	32.52/0.92	**36.51/0.88**

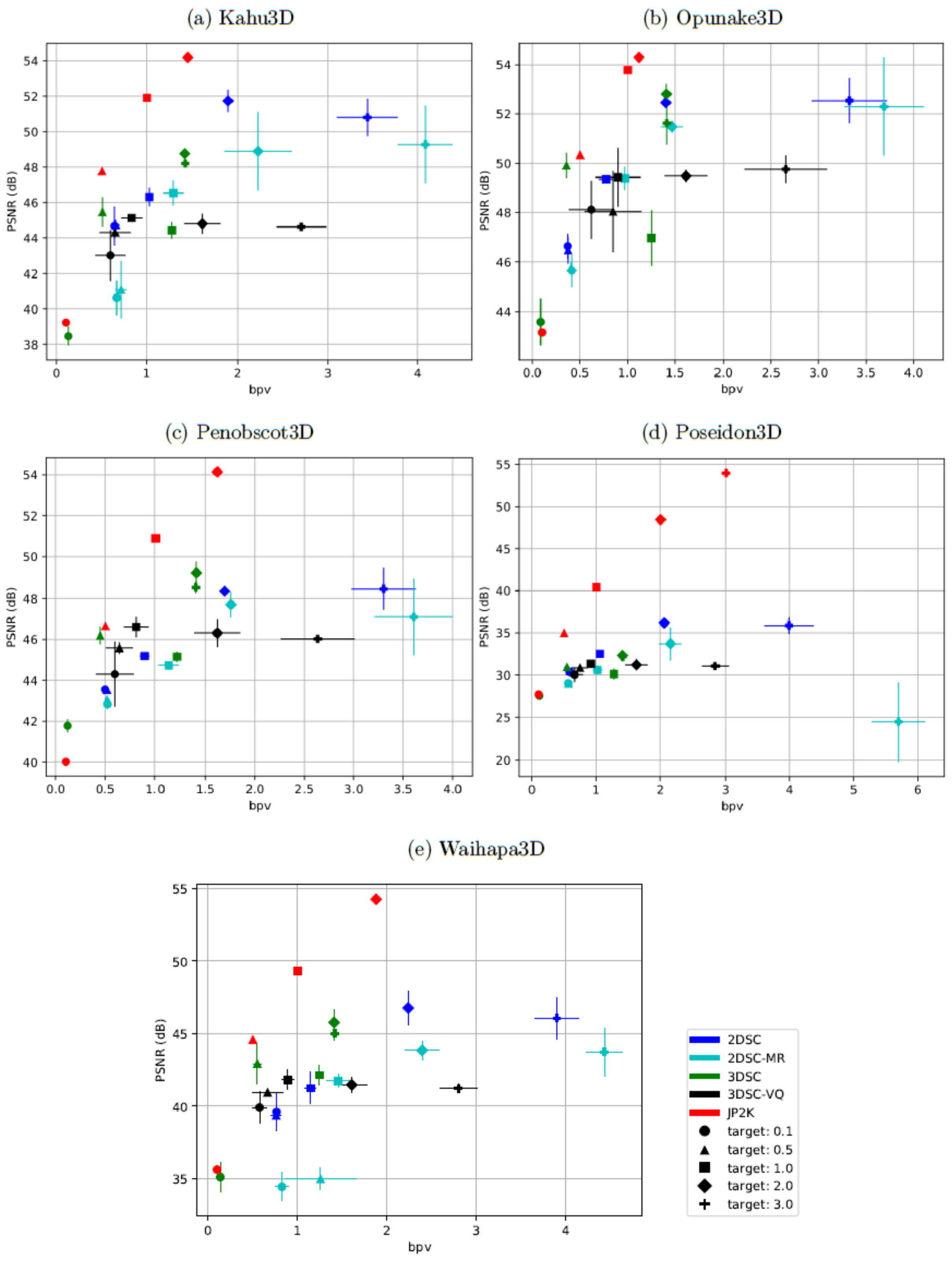
Source: created by the author.

4.6 COMPARISON RESULTS

This section provides a comparison between our four proposed methods 2DSC, 2DSC-MR, 3DSC, 3DSC-VQ and the JPEG2000 codec. In the previous sections, we discovered the best hyper-parameters setting for each method by performing individual experiments for a bit rate target $r_t = 1.0$ bpv. Besides, we noticed that the training set did not allow a comparison of all methods by using the same results from Chapter 4. As the Poseidon3D did not allow the convergence of the 3DSC-VQ method, for the next experiments we remove it from the training set and use only the Netherlands F3-Block and Parihaka3D surveys to train our models. We set the Kerry3D for validation and the remaining surveys Kahu3D, Opunake3D, Penobscot3D, Poseidon3D and Waihapa3D are used for testing.

Aiming to compare all methods, we train different models for 5 different bit rate targets $r_t = 0.1$, 0.5, 1.0, 2.0 and 3.0 bpv using the same training and inference procedures described in Section 4.2. Figure 32 shows the curve PSNR $\times$ bpv for all methods, for the testing surveys Kahu3D (Fig. 32a), Opunake3D (Fig. 32b), Penobscot3D (Fig. 32c), Poseidon3D (Fig. 32d) and Waihapa3D (Fig. 32e), respectively. Each subfigure shows the performance of each method in each survey for each bit rate target at an average of 3 rounds. Lines indicate the standard deviation for PSNR (vertical) and bpv (horizontal).

Figure 32 – Method comparison for all testing surveys. Reported results correspond to an average of 3 rounds. Lines in the vertical and horizontal directions indicate the standard deviation for PSNR and bpv, respectively.



Source: created by the author.

4.6.1 2DSC and 2DSC-MR Methods Comparison

Results from Figure 32 show that our methods can compress the volume using seismic sections at low bit rates with good PSNR. This result confirms our main hypothesis. In this sense, the main objective of this work is achieved even by using the simplest method. We can see that the 2DSC can reach bit rates around to 0.5 bpv with PSNR values close to 40 dB or more. For high bit rates, the method becomes less stable, with considerable variations in terms of PSNR and bit rate. We believe that it is due to the balance of the trade-off, in which the method tries to adjust to the bit rate target even it has already exceeded its compression limit. In general, the performance of the 2DSC method is reasonable for all seismic surveys.

Although the 2DSC is capable of compressing seismic sections, we conjectured that the information across scales from a multi-resolution approach improves the compression. Intending to verify it, we proposed the 2DSC-MR method, by decomposing the seismic section into two more scales and compressing them in a residual scheme. From Figure 32, we can see that, in general, the 2DSC method outperforms the 2DSC-MR. It evidences that in this case, most of the information can be captured using a simpler approach. For higher bit rates, next to 3.5 bpv, the 2DSC-MR can surpass the 2DSC for Opunake3D. This survey is more homogeneous, with only a few regions with high-frequency. It indicates that the multi-resolution can be beneficial for scenarios in which it is desirable to achieve high PSNR for low-frequency surveys. In this case, the 2DSC method might have reached its upper bound for PSNR, whereas the 2DSC-MR approach can still extract information from the data. More tests would be needed to explore this scenario, as well as a new parameter evaluation, but this is not our focus on this work.

4.6.2 3DSC and 3DSC-VQ Methods Comparison

From the analysis of Figure 32, we can see that for most of the surveys the 3DSC surpasses the 3DSC-VQ method. It seems to be a paradox since the representative power of the 3DSC-VQ is higher than the 3DSC. The encodings provided by 3DSC-VQ were expected to be more representative than those of 3DSC. We believe that it is due to the number of channels C of the latent representation. With more channels, the 3DSC-VQ has more vectors to encode, and the accuracy of the representation can be deteriorated. On the other hand, the 3DSC benefits from these channels to provide more accurate representations. We conjecture that in situations with a restricted number of channels in latent representation the 3DSC-VQ surpasses the 3DSC method. In this sense, we perform an experiment by using $r_t = 1.0$ bpv and $C = 32$ channels in the latent space. Table 22 shows the comparison between the 3DSC and 3DSC-VQ for all testing datasets. The mean and standard deviation reported for each survey refers to an average of 3 runs. We can notice that, on average, the 3DSC-VQ surpasses the 3DSC for all surveys in terms of PSNR and bit rate. It implies that for scenarios in which the bit rate is controlled by the number of centroids and it is desirable a smaller bit rate, the 3DSC-VQ is preferable than the 3DSC for seismic compression. The 3DSC-VQ benefits from the small number of vectors to learn more accurate centroids. Although a more detailed evaluation from this scenario is needed, this result validates our fourth hypothesis, which states that the compression of the whole volume without depth reductions can improve the quality reached.

Table 22 – 3DSC and 3DSC-VQ methods comparison for all testing surveys, regarding PSNR and bpv, respectively. Reported results are the mean and standard deviation from 3 rounds for a bit rate target $r_t = 1.0$ bpv.

Method	Kahu3D	Opunake3D	Penobscot3D	Poseidon3D	Waihapa3D
3DSC	41.29 ± 1.11	44.08 ± 0.41	43.19 ± 0.24	29.01 ± 0.58	39.44 ± 1.04
	0.62 ± 0.05	0.62 ± 0.04	0.60 ± 0.05	0.62 ± 0.05	0.62 ± 0.04
3DSC-VQ	**43.00 ± 0.45**	**48.11 ± 0.65**	**44.39 ± 0.34**	**29.74 ± 0.34**	**39.62 ± 0.38**
	0.46 ± 0.02	**0.46 ± 0.03**	**0.46 ± 0.03**	**0.47 ± 0.02**	**0.47 ± 0.01**

Source: created by the author.

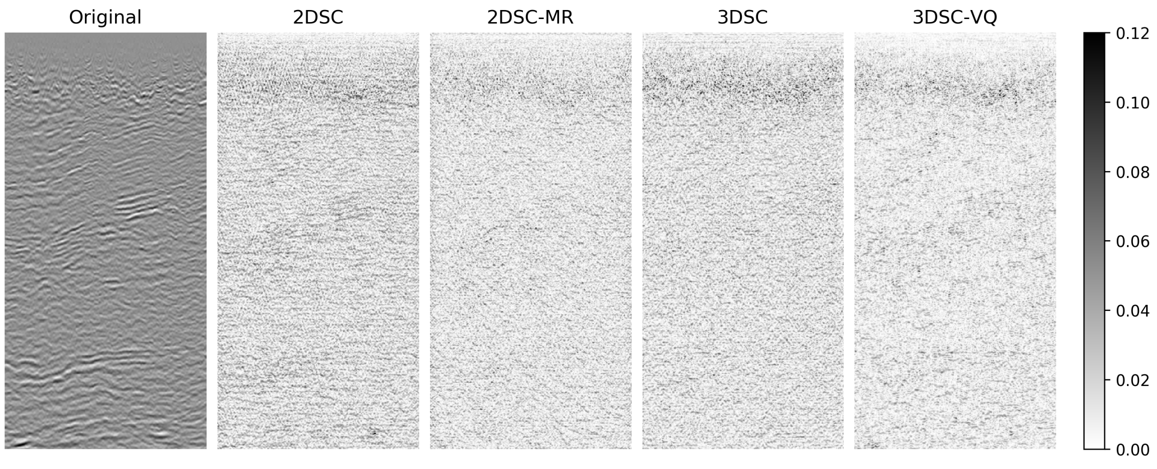
4.6.3 **Discussion about 2D and 3D Approaches**

Although bi-dimensional approaches have shown that they can compress the seismic data with good performance, smaller bit rates can be achieved by using three-dimensional models. From the analysis of Figure 32 we can see that in a comparison of our methods, for most of the surveys, the 3DSC method outperforms the 2DSC, 2DSC-MR and 3DSC-VQ for all surveys except for Poseidon3D. This verifies our third hypothesis, such that the 3D approach can generate representations more suitable for compression than 2D ones. In particular, for Poseidon3D, the 3DSC cannot reach the same PSNR from 2D approaches. The 3D convolution can extract spatial correlation in all directions, and for noisy surveys the interferences from all directions are considered, increasing the reconstruction error. The method achieves its upper bound by using fewer bits compared to previous approaches, and after 1.5 bpv there is no meaningful improvement in the PSNR.

Through the analysis from Figure 32 it is noticeable that no method always provides the best result in all situations. According to the objective, an approach can be more advantageous than another. In a generalist approach, the 3DSC is the better method for seismic compression since it achieves low bit rates with high PSNR values. Compared to other approaches, this method is more stable, yielding outputs with a smaller variance in terms of PSNR and bpv for all bit rate targets. However, the method performance is highly related to the seismic survey to be compressed. We notice that for Kahu3D, Opunake3D, Penobscot3D and Waihapa3D, until a bit rate of 1.5 bpv, the 3DSC provides better results. From this point the 3DSC is limited and bi-dimensional approaches can be used to achieve higher PSNR values. There is a slight improvement in PSNR and it may not be advantageous considering the number of bits spent on this task. For Poseidon3D the results show that the predominance of high frequencies leads to reconstructions with considerable loss of information. Bi-dimensional approaches are more suitable to deal with high frequencies. Noisy surveys need a specific model to deal with its interferences since in a generalist approach it is difficult to achieve PSNR higher than 40 dB.

Figure 33 depicts a visual comparison of the relative error for all methods for a slice of the Waihapa3D survey. We can notice that for all methods, most of the information is recovered with a PSNR about 41 dB, with the main error peaks occurring in high-frequency regions. The 2DSC provides a more structured error than other methods. Visually, we can notice that some frequencies are ignored. The 2DSC-MR provides a better relative error, with a balance between high and low frequencies. This error is closest to the expected. Better methods provide errors similar to the white noise. The 3DSC and 3DSC-VQ are intermediary, and the 3DSC-VQ surpasses the 3DSC in terms of visual quality.

Figure 33 – Comparative result of a reconstruction section from the Waihapa3D survey. The high quality represented by a high PSNR value reflects visually in the reconstruction. From left to right: original slice, 2DSC (41.22 dB /1.10 bpv), 2DSC-MR (42.07 dB /1.60 bpv), 3DSC (41.72 dB /1.27 bpv), 3DSC-VQ (41.34 dB /0.81 bpv).



Source: created by the author.

4.6.4 **JPEG2000 Comparison**

We cannot perform a comparison to methods of literature for seismic compression since the data used by them is not available. In general, most of the methods focus on compressing 2D seismic data, that differs from our approach and cannot be directly compared. The method proposed by Radosavljević et al. (2017) is the closest to ours, in which 3D seismic data is compressed. However, the data used in their experiments is private, and the source code is not available. Thus, we perform a comparison with the JPEG2000 codec. Although the JPEG2000 was not developed to deal with the seismic domain, it can be used to compress seismic images since it is available and does not need additional efforts to be used. To this end, we convert the volume to 8-bit unsigned integers and extract each section to be compressed individually. The reconstructed image is converted to the original range and the PSNR is calculated over the 32-bit floating-point representation. From Figure 32, we can notice that most our methods do not outperform the JPEG2000. Moreover, compared to our methods, the JPEG has a more stable and predictable behavior in terms of PSNR and bpv. We believe that it occurs because the seismic surveys used in this work were quantized in 8-bit unsigned integers and synthetically converted to 32-bit floating-point. In this sense, our methods are working below the capability, since the volumes do not use all 32 bits of the network. It is expected that in this situation the JPEG2000 provides better results as the codec was designed to image domain. Even under these conditions, the 3DSC can outperform the JPEG2000 for very low bit rates. Table 23 shows that for a bit rate target $r_t = 0.1$ bpv our method surpasses the codec for Opunake3D and Penobscot3D surveys. It evidences that in scenarios of extreme compression, the 3DSC benefits from the correlation present on these surveys to achieve high PSNR values at a low bit rate.

Table 23 – Comparison of 3DSC and JPEG2000 for all testing surveys, regarding PSNR and bpv, respectively. Reported results are the mean and standard deviation from 3 rounds for a bit rate target $r_t = 0.1$ bpv.

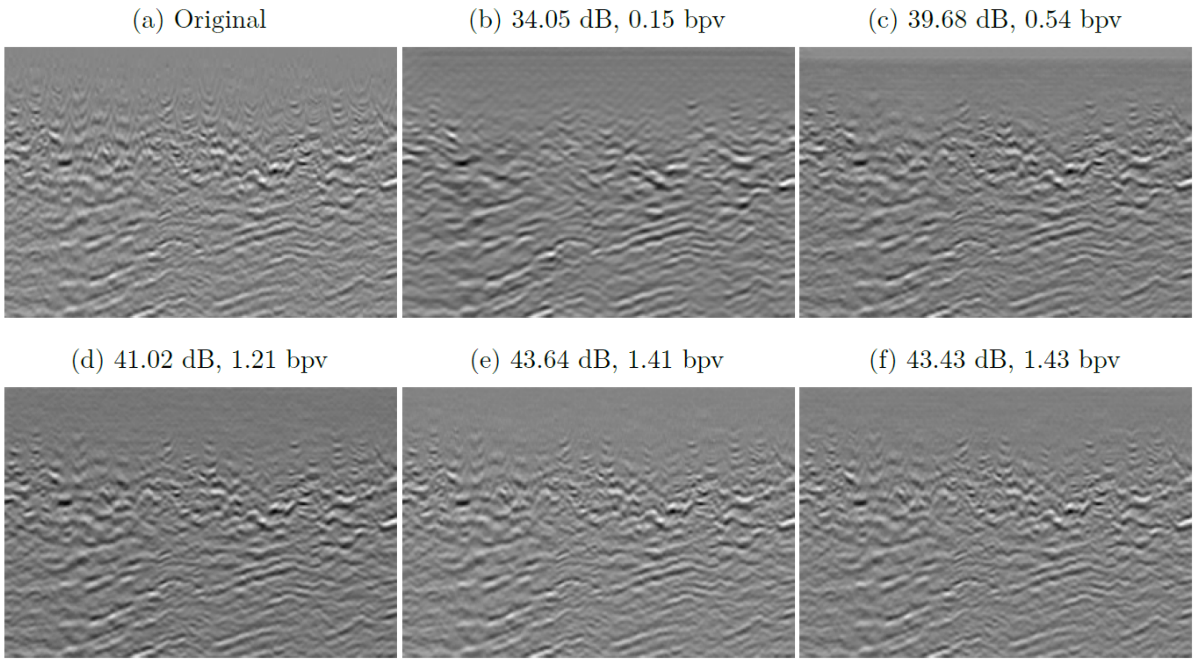
PSNR/bpv	Kahu3D	Opunake3D	Penobscot3D	Poseidon3D	Waihapa3D
3DSC	$38.47 \pm 5.42\text{e}{-1}$	$\mathbf{43.60 \pm 9.60}\text{e}{-1}$	$\mathbf{41.77 \pm 3.37}\text{e}{-1}$	$27.56 \pm 1.46\text{e}{-1}$	$35.10 \pm 9.93\text{e}{-1}$
	$0.13 \pm 1.71\text{e}{-2}$	$\mathbf{0.09 \pm 2.12}\text{e}{-2}$	$\mathbf{0.12 \pm 2.21}\text{e}{-2}$	$0.11 \pm 1.53\text{e}{-2}$	$0.14 \pm 1.81\text{e}{-2}$
JPEG2000	$\mathbf{39.23 \pm 4.17}\text{e}{-4}$	$43.16 \pm 1.08\text{e}{-3}$	$40.02 \pm 4.31\text{e}{-4}$	$\mathbf{27.67 \pm 5.66}\text{e}{-5}$	$\mathbf{35.62 \pm 7.58}\text{e}{-4}$
	$\mathbf{0.10 \pm 1.12}\text{e}{-5}$	$0.10 \pm 2.68\text{e}{-5}$	$0.10 \pm 1.49\text{e}{-5}$	$\mathbf{0.10 \pm 2.87}\text{e}{-6}$	$\mathbf{0.10 \pm 2.09}\text{e}{-5}$

Source: created by the author.

To evaluate the 3DSC qualitatively, we reconstruct the same seismic section from the Waihapa3D survey at different bit rates, as shown in Figure 34. We can see that as the PSNR increases, the differences between the original and reconstructed slices are decreased. For PSNR smaller than 40 dB (Figs. 34a and 34b), some distortions are added and the high-frequencies are attenuated. For PSNR above to 40 dB (Fig. 34d) most of the information is preserved, with some distortion in noisy regions. With a PSNR higher than 43 dB (Figs. 34e and 34f), we can assume that the reconstruction errors are visually attenuated.

In short, the main advantage of our method is the flexibility of learning from data. It is possible to create general or specific models from the data to be compressed, which is not reasonable to be done by using handcrafted approaches. Besides, it is possible to perform fine-tuning in a generalist model so that it can improve results at a lower cost compared to training the network from scratch. Seismic data compression differs from image compression task, since that the objective of an image compressor is to be more

Figure 34 – Qualitative result for 3DSC method for a slice of Waihapa3D.



Source: created by the author.

generalist as possible and the seismic data compressor can have not this need. Considering that the seismic data occupies about hundreds of gigabytes of storage, we ask about the possibility of training specific models for each survey intending to get higher accuracy in terms of bit rate and PSNR. Experiments show that better models are obtained by training on the same testing data or data with similar characteristics. Table 15 shows that for the Poseidon3D, only a specific model would reach the minimum acceptable PSNR to achieve most of the information retrieved by the decoder. Hypothetically, if this data occupied 100GB, at a 1:27 compression rate from Table 15, only 4GB would be needed to store it. To store the network coefficients for the 3DSC model, about 34MB of storage would be required, which is much smaller than the data size. On the other hand, training a specific network requires available hardware and time for training. Considering that it is not necessary to train on all 100GB of data to achieve good results, it is reasonable to assume that specific models, in which a model is trained for each survey, can be more suitable to deal with the compression of the seismic data than a generalist approach, in which a single model is used to compress different surveys.

5 CONCLUSIONS AND FUTURE WORK

In this work, we presented a deep learning approach to 3D post-stack seismic compression. We proposed four methods, two based on bi-dimensional compression, called 2DSC and 2DSC-MR, and two based on three-dimensional compression, called 3DSC and 3DSC-VQ. These methods consist of adaptations of a compression method designed for general-purpose images to the seismic domain. They are composed of two networks, an autoencoder for dealing with the rate-distortion trade-off and a 3D probabilistic model, for entropy estimation. The bit rate of the compressed volume is controlled by hyper-parameter tuning. We performed individual experiments for each method to find the best hyper-parameters setting for a compression rate of 1.0 bpv.

Experimental results show that the 2DSC method achieved PSNR values higher than 40 dB with a bit rate next to 1.0 bpv for most of the testing surveys. In general, the performance is degraded for higher frequency data, but even under these conditions it is possible to improve the quality by using more bits. Better results are achieved by training on surveys with the same characteristics of the test set. However, even under a generalist situation, the method achieves good results if the training set has a balance between high and low frequencies samples.

The 2DSC-MR was designed from the hypothesis that the information across multiple resolutions can improve the compression. Our tests evidenced that the 2DSC method outperforms the 2DSC-MR approach. However, the 3DSC-MR presented the possibility of improvement for higher bit rates. For a training set composed of only a few regions of high-frequency, the multi-resolution can reduce the model overfitting, improving the compression of surveys with a balance of high and low frequencies. Thus, further investigation about multi-resolution using neural networks in the compression problem is needed.

The 3DSC method extended the 2DSC to deal with volumes instead of slices. Experimental results show that even with a depth reduction on the input samples, this method can compress the seismic data using fewer bit rates than the bi-dimensional ones. For most of surveys the 3DSC surpasses all methods for bit rates smaller than 1.0 bpv. However, for high-frequency data, the method did not achieve reasonable PSNR values, even for high bit rates. In this case, 2D approaches are more suitable to compress noisy information.

The 3DSC-VQ extended the 3DSC method by considering a higher dimensional latent representation. In this sense, the representative capacity of this method was expected to be superior to the 3DSC. Experiments show that the 3DSC-VQ method outperforms the 3DSC only in scenarios in which the number of channels of the latent space is small. The 3DSC-VQ benefits from the small number of vectors to learn more accurate centroids.

Further investigations on new ways to avoid 4D convolutions are needed.

Overall, the performance of the methods was good even in a generalist scenario. For models trained in sets with a balance between high and low frequencies it is possible to compress different surveys, achieving high PSNR values. Better results can be obtained by training on the same testing survey. Experiments show that no method provides the best result for all situations. A method can be more beneficial for a situation than others, and it depends on the data and the objective to be reached. Due to the local nature of the methods, the reported results were not dependent on the size of the input data. The 3DSC provides better results for most of the cases. However, for noisy surveys or when bit rates above 1.0 are desired, the 2D approaches are more adequate, as they can reach a better reconstruction quality. The main advantage of our methods is the flexibility of learning from data. It is possible to compress the seismic survey in a generalist or specific approach, which is not reasonable by using handcrafted approaches. Specific models can be trained to achieve high PSNR and smaller bit rates. In comparison to JPEG2000, our methods were worse for all surveys. However, for seismic data with low frequencies, the 3DSC method was able to outperform the JPEG2000 for a bit rate target of 0.1 bpv.

As future work, we intend to investigate the performance of our method over seismic surveys sampled in 32-bit floating-points. We also intend to evaluate the multi-resolution approach for high bit rates, as it presented possibilities of improvement. Other ways to compress the seismic data avoiding a 4D probabilistic model can be explored. The fine-tuning of the whole network, or of the centroids only, can provide improvement in terms of accuracy and bit rate at a lower cost than training the network from scratch. As the pre-stack is a massive data, there exists a demand for their compression. We intend to address it in a future work, for instance, considering pre- and post-stack models to improve both data encoding. In addition, expert analysis is of great importance, for example, to develop specialized loss functions for the seismic domain.

REFERENCES

ACKLEY, David H.; HINTON, Geoffrey E.; SEJNOWSKI, Terrence J. A learning algorithm for boltzmann machines. **Cognitive Science**, v. 9, n. 1, p. 147–169, 1985. ISSN 0364-0213.

AGUSTSSON, Eirikur; MENTZER, Fabian; TSCHANNEN, Michael; CAVIGELLI, Lukas; TIMOFTE, Radu; BENINI, Luca; GOOL, Luc V. Soft-to-hard vector quantization for end-to-end learning compressible representations. In: **Advances in Neural Information Processing Systems 30**. [S.l.]: Curran Associates, Inc., 2017. p. 1141–1151.

AGUSTSSON, Eirikur; TSCHANNEN, Michael; MENTZER, Fabian; TIMOFTE, Radu; GOOL, Luc Van. Generative adversarial networks for extreme learned image compression. In: **Proceedings of the IEEE International Conference on Computer Vision**. [S.l.: s.n.], 2019. p. 221–231.

AHARONI, Gil; AVERBUCH, Amir; COIFMAN, Ronald; ISRAELI, Moshe. Local cosine transform - a method for the reduction of the blocking effect in jpeg. In: **Wavelet Theory and Application**. [S.l.]: Springer, 1993. p. 7–38.

AVERBUCH, A. Z.; MEYER, F.; STROMBERG, J. .; COIFMAN, R.; VASSILIOU, A. Low bit-rate efficient compression for seismic data. **IEEE Transactions on Image Processing**, v. 10, n. 12, p. 1801–1814, Dec 2001. ISSN 1057-7149.

BALLARD, Dana H. Modular learning in neural networks. In: **Proceedings of the Sixth National Conference on Artificial Intelligence - Volume 1**. [S.l.]: AAAI Press, 1987. (AAAI'87), p. 279–284. ISBN 0934613427.

BALLÉ, Johannes; LAPARRA, Valero; SIMONCELLI, Eero. End-to-end optimized image compression. In: **5th International Conference on Learning Representations, ICLR 2017**. [S.l.: s.n.], 2017.

BALLÉ, Johannes; LAPARRA, Valero; SIMONCELLI, Eero P. Density modeling of images using a generalized normalization transformation. In: **4th International Conference on Learning Representations, ICLR**. [S.l.: s.n.], 2016.

BALLÉ, Johannes; MINNEN, David; SINGH, Saurabh; HWANG, Sung Jin; JOHNSTON, Nick. Variational image compression with a scale hyperprior. In: **6th International Conference on Learning Representations, ICLR**. [S.l.: s.n.], 2018.

BRUN-GOUANVIC, Claire le et al. **Ystoria sancti Thome de Aquino de Guillaume de Tocco (1323)**. [S.l.]: Pontificial Institute of Mediaeval Studies, 1996.

CARREIRA, Joao; ZISSERMAN, Andrew. Quo vadis, action recognition? a new model and the kinetics dataset. In: **proceedings of the IEEE Conference on Computer Vision and Pattern Recognition**. [S.l.: s.n.], 2017. p. 6299–6308.

COVER, Thomas M.; THOMAS, Joy A. **Elements of Information Theory (Wiley Series in Telecommunications and Signal Processing)**. USA: Wiley-Interscience, 2006. ISBN 0471241954.

EVANS, Brian J. **A handbook for seismic data acquisition in exploration**. [S.l.]: Society of exploration geophysicists, 1997.

GENG, Zhicheng; WU, Xinming; SHI, Yunzhi; FOMEL, Sergey. Relative geologic time estimation using a deep convolutional neural network. In: **SEG Technical Program Expanded Abstracts 2019**. [S.l.]: Society of Exploration Geophysicists, 2019. p. 2238–2242.

GONZALEZ, Rafael C.; WINTZ, Paul. **Digital Image Processing (2nd Ed.)**. USA: Addison-Wesley Longman Publishing Co., Inc., 1987. ISBN 0201110261.

HABIBIAN, Amirhossein; ROZENDAAL, Ties van; TOMCZAK, Jakub M; COHEN, Taco S. Video compression with rate-distortion autoencoders. In: **Proceedings of the IEEE International Conference on Computer Vision**. [S.l.: s.n.], 2019. p. 7033–7042.

HE, K.; ZHANG, X.; REN, S.; SUN, J. Deep residual learning for image recognition. In: **2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)**. [S.l.: s.n.], 2016. p. 770–778. ISSN 1063-6919.

HORNIK, Kurt; STINCHCOMBE, Maxwell; WHITE, Halbert. Multilayer feedforward networks are universal approximators. **Neural Networks**, v. 2, n. 5, p. 359–366, 1989. ISSN 0893-6080.

HU, J.; SHEN, L.; SUN, G. Squeeze-and-excitation networks. In: **2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition**. [S.l.: s.n.], 2018. p. 7132–7141. ISSN 1063-6919.

HUANG, Chao; LIU, Haojie; CHEN, Tong; PU, Shiliang; SHEN, Qiu; MA, Zhan. Extreme image coding via multiscale autoencoders with generative adversarial optimization. In: **2019 IEEE Visual Communications and Image Processing (VCIP)**. [S.l.: s.n.], 2019. p. 1–4.

KRIZHEVSKY, Alex; SUTSKEVER, Ilya; HINTON, Geoffrey E. Imagenet classification with deep convolutional neural networks. In: **Advances in Neural Information Processing Systems 25**. [S.l.]: Curran Associates, Inc., 2012. p. 1097–1105.

LECUN, Yann; BOSER, Bernhard E.; DENKER, John S.; HENDERSON, Donnie; HOWARD, R. E.; HUBBARD, Wayne E.; JACKEL, Lawrence D. Handwritten digit recognition with a back-propagation network. In: **Advances in Neural Information Processing Systems 2**. [S.l.]: Morgan-Kaufmann, 1990. p. 396–404.

LI, Mu; ZUO, Wangmeng; GU, Shuhang; ZHAO, Debin; ZHANG, David. Learning convolutional networks for content-weighted image compression. In: **Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition**. [S.l.: s.n.], 2018. p. 3214–3223.

LI, Mu; ZUO, Wangmeng; GU, Shuhang; YOU, Jane; ZHANG, David. Learning content-weighted deep image compression. **arXiv preprint arXiv:1904.00664**, 2019.

LI, Wei; ZHAO, Rui; XIAO, Tong; WANG, Xiaogang. Deepreid: Deep filter pairing neural network for person re-identification. In: **The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)**. [S.l.: s.n.], 2014.

LIU, Yiding; XIONG, Zixiang; LU, Ligang; HOHL, Detlef. Fast snr and rate control for jpeg xr. In: IEEE. **Signal Processing and Communication Systems (ICSPCS), 2016 10th International Conference on**. [S.l.], 2016. p. 1–7.

MENTZER, Fabian; AGUSTSSON, Eirikur; TSCHANNEN, Michael; TIMOFTE, Radu; GOOL, Luc Van. Conditional probability models for deep image compression. In: **IEEE Conference on Computer Vision and Pattern Recognition (CVPR)**. [S.l.: s.n.], 2018.

MHASKAR, H. N.; POGGIO, T. Deep vs. shallow networks: An approximation theory perspective. **Analysis and Applications**, v. 14, n. 06, p. 829–848, 2016.

MINNEN, David; BALLÉ, Johannes; TODERICI, George D. Joint autoregressive and hierarchical priors for learned image compression. In: **Advances in Neural Information Processing Systems 31**. [S.l.]: Curran Associates, Inc., 2018. p. 10771–10780.

NAIR, Vinod; HINTON, Geoffrey E. Rectified linear units improve restricted boltzmann machines. In: **Proceedings of the 27th International Conference on International Conference on Machine Learning**. Madison, WI, USA: Omnipress, 2010. (ICML'10), p. 807–814. ISBN 9781605589077.

NAKANISHI, Ken M; MAEDA, Shin-ichi; MIYATO, Takeru; OKANOHARA, Daisuke. Neural multi-scale image compression. In: SPRINGER. **Asian Conference on Computer Vision**. [S.l.], 2018. p. 718–732.

NAVARRO, JP; SCHIAVON, AP; VIEIRA, M; SILVA, PM. Deep seismic compression. In: EUROPEAN ASSOCIATION OF GEOSCIENTISTS & ENGINEERS. **81st EAGE Conference and Exhibition 2019**. [S.l.], 2019. v. 2019, n. 1, p. 1–5.

NUHA, Hilal H; LIU, Bo; MOHANDES, M; DERICHE, M. Seismic data compression using signal alignment and pca. In: IEEE. **2017 9th IEEE-GCC Conference and Exhibition (GCCCE)**. [S.l.], 2017. p. 1–6.

OORD, Aaron Van Den; KALCHBRENNER, Nal; ESPEHOLT, Lasse; VINYALS, Oriol; GRAVES, Alex et al. Conditional image generation with pixelcnn decoders. In: **Advances in Neural Information Processing Systems**. [S.l.: s.n.], 2016. p. 4790–4798.

OORD, Aaron Van Den; KALCHBRENNER, Nal; KAVUKCUOGLU, Koray. Pixel recurrent neural networks. In: **International Conference on Machine Learning**. [S.l.: s.n.], 2016. p. 1747–1756.

RADOSAVLJEVIĆ, Miloš; XIONG, Zixiang; LU, Ligang; VUKOBRATOVIĆ, Dejan. High bit-depth image compression with application to seismic data. In: IEEE. **Visual Communications and Image Processing (VCIP), 2016**. [S.l.], 2016. p. 1–4.

RADOSAVLJEVIĆ, Miloš; XIONG, Zixiang; LU, Ligang; HOHL, Detlef; VUKOBRATOVIĆ, Dejan. Hevc-based compression of high bit-depth 3d seismic data. In: IEEE. **Image Processing (ICIP), 2017 IEEE International Conference on**. [S.l.], 2017. p. 4028–4032.

RIPPEL, Oren; BOURDEV, Lubomir. Real-time adaptive image compression. In: **International Conference on Machine Learning**. [S.l.: s.n.], 2017. p. 2922–2930.

RIPPEL, Oren; NAIR, Sanjay; LEW, Carissa; BRANSON, Steve; ANDERSON, Alexander G; BOURDEV, Lubomir. Learned video compression. In: **Proceedings of the IEEE International Conference on Computer Vision**. [S.l.: s.n.], 2019. p. 3454–3463.

RØSTEN, Tage; RAMSTAD, Tor A; AMUNDSEN, Lasse. Optimization of sub-band coding method for seismic data compression. **Geophysical Prospecting**, v. 52, n. 5, p. 359–378, 2004.

SCHIAVON, Ana Paula; NAVARRO, João Paulo; VIEIRA, Marcelo Bernardes; SILVA, Pedro Mário Cruz e. Low bit rate 2d seismic image compression with deep autoencoders. In: SPRINGER. **International Conference on Computational Science and Its Applications**. [S.l.], 2019. p. 397–407.

SEG. **Open data**. 2019. <http://wiki.seg.org/wiki/Open_data>. Accessed: 2019-01-10.

SHANNON, Claude E. Coding theorems for a discrete source with a fidelity criterion. **IRE Nat. Conv. Rec**, v. 4, n. 142-163, p. 1, 1959.

SHI, Yunzhi; WU, Xinming; FOMEL, Sergey. Automatic salt-body classification using a deep convolutional neural network. In: **SEG Technical Program Expanded Abstracts 2018**. [S.l.]: Society of Exploration Geophysicists, 2018. p. 1971–1975.

SKODRAS, A.; CHRISTOPOULOS, C.; EBRAHIMI, T. The jpeg 2000 still image compression standard. **IEEE Signal Processing Magazine**, v. 18, n. 5, p. 36–58, Sep. 2001. ISSN 1558-0792.

SPANIAS, Andreas S; JONSSON, Stefan B; STEARNS, Samuel D. Transform methods for seismic data compression. **IEEE Transactions on Geoscience and Remote sensing**, IEEE, v. 29, n. 3, p. 407–416, 1991.

SULLIVAN, Gary J; OHM, Jens-Rainer; HAN, Woo-Jin; WIEGAND, Thomas. Overview of the high efficiency video coding(hevc) standard. **IEEE Transactions on circuits and systems for video technology**, v. 22, n. 12, p. 1649–1668, 2012.

THEIS, Lucas; SHI, Wenzhe; CUNNINGHAM, Andrew; HUSZÁR, Ferenc. Lossy image compression with compressive autoencoders. In: **International Conference on Learning Representations**. [S.l.: s.n.], 2017.

TODERICI, George; VINCENT, Damien; JOHNSTON, Nick; HWANG, Sung Jin; MINNEN, David; SHOR, Joel; COVELL, Michele. Full resolution image compression with recurrent neural networks. In: **Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition**. [S.l.: s.n.], 2017. p. 5306–5314.

TRIPPA, Lorenzo; WALDRON, Levi; HUTTENHOWER, Curtis; PARMIGIANI, Giovanni et al. Bayesian nonparametric cross-study validation of prediction methods. **The Annals of Applied Statistics**, Institute of Mathematical Statistics, n. 1, p. 402–428, 2015.

WU, X.; SHI, Y.; FOMEL, S.; LIANG, L.; ZHANG, Q.; YUSIFOV, A. Z. Faultnet3d: Predicting fault probabilities, strikes, and dips with a single convolutional neural network. **IEEE Transactions on Geoscience and Remote Sensing**, p. 1–18, 2019.

YILMAZ, Öz. **Seismic data analysis: Processing, inversion, and interpretation of seismic data**. [S.l.]: Society of exploration geophysicists, 2001.