UNIVERSIDADE FEDERAL DE JUIZ DE FORA
INSTITUTO DE CIÊNCIAS EXATAS
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Kevyn Swhants dos Santos Ribeiro

Compressão de Dados Sísmicos Pós-Pilha usando uma Rede Adversária Generativa com Função de Perda Composta

Juiz de Fora

2022

Kevyn Swhants dos Santos Ribeiro

Compressão de Dados Sísmicos Pós-Pilha usando uma Rede Adversária Generativa com Função de Perda Composta

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

Orientador: Prof. Dr. Marcelo Bernardes Vieira

Coorientador: Prof. Dr. Saulo Moraes Villela

Juiz de Fora

2022

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

Ribeiro, Kevyn Swhants dos Santos.
Compressão de Dados Sísmicos Pós-Pilha usando uma Rede Adversária Generativa com Função de Perda Composta / Kevyn Swhants dos Santos Ribeiro. – 2022.
71 f. : il.

Orientador: Marcelo Bernardes Vieira
Coorientador: Saulo Moraes Villela
Dissertação (Mestrado) – Universidade Federal de Juiz de Fora, Instituto de Ciências Exatas. Programa de Pós-Graduação em Ciência da Computação, 2022.

1. Compressão de Dados Sísmicos. 2. Aprendizado Profundo. 3. Redes Adversárias Generativas. 4. Dados Sísmicos Pós-Pilha 3D. I. Vieira, Marcelo Bernardes, orient. II. Villela, Saulo Moraes, coorient. III. Título.

Kevyn Swhants dos Santos Ribeiro

Compressão de Dados Sísmicos Pós-Pilha usando uma Rede Adversária Generativa com Função de Perda Composta

Dissertação apresentada ao Programa de Pós-graduação em Ciência da Computação da Universidade Federal de Juiz de Fora como requisito parcial à obtenção do título de Mestre em Ciência da Computação. Área de concentração: Ciência da Computação.

Aprovada em 22 de setembro de 2022.

BANCA EXAMINADORA

Prof. Dr. Marcelo Bernardes Vieira - Orientador
Universidade Federal de Juiz de Fora

Prof. Dr. Saulo Moraes Villela - Coorientador
Universidade Federal de Juiz de Fora

Prof. Dr. Luiz Maurílio da Silva Maciel
Universidade Federal de Juiz de Fora

Prof. Dr. Hélio Pedrini
Universidade Estadual de Campinas

Juiz de Fora, 13/09/2022.

Documento assinado eletronicamente por **Marcelo Bernardes Vieira**, **Professor(a)**, em 20/03/2023, às 14:52, conforme horário oficial de Brasília, com fundamento no § 3º do art. 4º do Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por **Helio Pedrini**, **Usuário Externo**, em 20/03/2023, às 15:29, conforme horário oficial de Brasília, com fundamento no § 3º do art. 4º do Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por **Luiz Maurílio da Silva Maciel**, **Professor(a)**, em 20/03/2023, às 15:57, conforme horário oficial de Brasília, com fundamento no § 3º do art. 4º do Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por **Saulo Moraes Villela**, **Professor(a)**, em 21/03/2023, às 10:26, conforme horário oficial de Brasília, com fundamento no § 3º do art. 4º do Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por **Kevyn Swhants dos Santos Ribeiro**, **Usuário Externo**, em 24/03/2023, às 17:16, conforme horário oficial de Brasília, com fundamento no § 3º do art. 4º do Decreto nº 10.543, de 13 de novembro de 2020.

A autenticidade deste documento pode ser conferida no Portal do SEI-Ufjf (www2.ufjf.br/SEI) através do ícone Conferência de Documentos, informando o código verificador **0946841** e o código CRC **08554FEE**.

AGRADECIMENTOS

Primeiramente agradeço a Deus, por me permitir superar todos os obstáculos encontrados durante esta jornada. Aos meus pais, pelos incentivos e esforços para me possibilitar uma boa criação e educação. À Cecília, por sempre estar ao meu lado, me apoiando e me dando suporte emocional, além da compreensão dos meus vários momentos de ausência devido às obrigações acadêmicas. A todos os meus parentes, pelo encorajamento e apoio.

À Ana Paula Schiavon, pelo apoio, amizade e ajuda durante este trabalho. Aos meus amigos de mestrado, em especial ao Emanuel Antônio Parreiras e ao Lucas Rodrigues Frank, por tornarem o desafio do ensino remoto mais fácil, pelo conhecimento, ideias, e códigos compartilhados, além de proporcionarem distração e apoio durante momentos difíceis. À Andreza, pela amizade e paciência para me ensinar mais sobre a língua inglesa. Aos inúmeros amigos que conheci na Twitch, especialmente à Suzy, por todo carinho e sempre conseguir me fazer rir.

Aos professores Marcelo e Saulo, pela orientação, dedicação e paciência, sem a qual este trabalho não seria possível. À NVIDIA, em especial ao João Paulo Navarro e ao Pedro Mário Cruz e Silva, por fornecerem o poder computacional necessário para realizar os incontáveis experimentos feitos durante a pesquisa.

Agradeço também aos professores do PPGCC e do DCC que contribuíram para minha formação. Aos funcionários e funcionárias do ICE, por toda assistência prestada, em especial à Sarah Abreu e Silva pela atenção e cuidado em cada atendimento e solicitação. À UFJF e FAPEMIG, pelo apoio financeiro sem o qual eu não conseguiria realizar este mestrado e a todos que contribuíram de alguma forma para a minha conclusão.

"The potential benefits of artificial intelligence are huge, so are the dangers." (DAVE WATERS).

RESUMO

Dados sísmicos fornecem informações estruturais e estáticas da região de onde foram coletados, usadas para determinar regiões para explorar petróleo e gás. As melhorias nos métodos de aquisição, especialmente na alta qualidade dos sensores, também aumentaram o tamanho dos dados sísmicos. A motivação para comprimi-los vem da necessidade de centenas de terabytes para transmitir e armazenar dados sísmicos. Este trabalho apresenta um método para compressão de dados sísmicos pós-pilha tridimensional (3D) integrando um autocodificador baseado em convoluções 3D e uma rede adversária generativa (GAN). O principal desafio dos autocodificadores é como explorar a redundância volumétrica, mantendo as dimensões da representação latente. O método proposto é baseado em uma rede neural convolucional para compressão de dados sísmicos chamada 3DSC. A principal hipótese é que a arquitetura da 3DSC pode ser melhorada pelo treinamento adversário. Assim, é proposto um novo método de compressão de dados sísmicos baseado em 3D (3DSC-GAN) ao acoplar a rede 3DSC a uma GAN. O módulo decodificador é utilizado como um gerador de dados sísmicos pós-pilha integrado a um módulo discriminador para melhor explorar a redundância volumétrica presente no dado 3D. Também é proposta uma nova forma para calcular a distorção em dados multidimensionais, como dados sísmicos. Visto que funções genéricas ignoram a estrutura do dado 3D e o consideram como um vetor 1D, a ideia consiste em aplicar uma função de perda diferente para cada eixo, para uma redução de dimensionalidade que melhor capture o erro de acordo com sua grandeza. Para isso, é feito um estudo extensivo para analisar as possíveis combinações de funções para o problema de compressão de dados sísmicos pós-pilha 3D. Os resultados indicam que o método 3DSC-GAN supera os métodos anteriores para taxas de bits alvo muito baixas, aumentando a relação sinal-ruído de pico (PSNR) com alta qualidade visual de reconstrução. Além disso, os experimentos realizados aplicando a nova função de distorção mostram que ela auxilia no processo de aprendizado da rede, gerando uma reconstrução superior comparado com métodos que utilizam PSNR como função de distorção, em termos quantitativos e qualitativos.

Palavras-chave: Compressão de Dados Sísmicos. Aprendizado Profundo. Redes Adversárias Generativas. Dados Sísmicos Pós-Pilha 3D.

ABSTRACT

Seismic data provide structural and static information of the region where they were acquired, used to determine regions to explore oil and gas. The improvements in the acquisition methods, especially in the high quality of the sensors, have also increased the size of the seismic data. The motivation for compressing them comes from the demand for hundreds of terabytes to transmit and store seismic data. This work presents a method for three-dimensional (3D) poststack seismic data compression integrating a 3D convolution-based autoencoder to a generative adversarial network (GAN). The main challenge of the 3D convolutional autoencoder is how to exploit volumetric redundancy, keeping the latent representation dimensions. The proposed method is based on a convolutional neural network for seismic data compression called 3DSC. The main hypothesis is that 3DSC architecture can be improved by adversarial training. Thus, a new 3D-based seismic data compression method (3DSC-GAN) is proposed by coupling the 3DSC network to a GAN. The decoder module is used as a generator of poststack seismic data integrated with a discriminator module to better exploit the volumetric redundancy of the 3D data. Also, a new fashion to calculate the distortion for multidimensional data is proposed, such as seismic data. Since the generic functions ignore the 3D data structure and consider it as a 1D vector, the idea is to apply a different loss function for each axis, for a reduction in dimensionality that better captures the error according to its magnitude. For this purpose, an extensive study is performed to analyze the possible combinations of functions for the 3D poststack seismic data compression problem. Results indicate that the 3DSC-GAN method outperforms previous ones for very low target bit rates, increasing the peak signal-to-noise ratio (PSNR) with high visual reconstruction quality. In addition, the experiments using the new distortion function show that it benefits the network learning process, generating a superior reconstruction compared to methods that use PSNR as a distortion function, in quantitative and qualitative terms.

Keywords: Seismic Data Compression. Deep Learning. Generative Adversarial Networks. 3D Poststack Seismic Data.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

1D	Unidimensional
2D	Bidimensional
2DSC	*2D-based Seismic Data Compression*
3D	Tridimensional
3DSC	*3D-based Seismic Data Compression*
BPG	*Better Portable Graphics*
bpv	Bits Por Voxel
CNN	*Convolutional Neural Network*
dB	Decibel
DNN	*Deep Neural Network*
GAN	*Generative Adversarial Network*
GRU	*Gated Recurrent Unit*
HEVC	*High-Efficiency Video Coding*
JPEG	*Joint Photographic Experts Group*
JPEG2000	*Joint Photographic Experts Group 2000*
JPEG-XR	*Joint Photographic Experts Group Extended Range*
LPIPS	*Learned Perceptual Image Patch Similarity*
LSTM	*Long Short-Term Memory*
MCSC	*Multichannel Seismic Data Compression*
MSE	*Mean Squared Error*
MS-SSIM	*Multi-Scale Structural Similarity*
PSNR	*Peak Signal-to-Noise Ratio*
ReLU	*Rectified Linear Unit*
RNN	*Recurrent Neural Network*
ResNET	*Residual Neural Network*
SEG	*Society of Exploration Geophysicists*
SNR	*Signal-to-Noise Ratio*
VAE	*Variational Autoencoders*
VC	Visão Computacional

SUMÁRIO

1 INTRODUÇÃO

Dados sísmicos fornecem informações estruturais e estáticas da região de onde foram coletados. Eles têm uma grande importância para a exploração de petróleo e gás, causando um impacto comercial nos anos 1990, ao aumentar a taxa de perfuração com sucesso nas empresas do setor (BROWN, 2011). Ao longo dos anos, várias inovações tecnológicas surgiram para beneficiar principalmente o desenvolvimento de equipamentos de aquisição dos dados sísmicos e a sua interpretação.

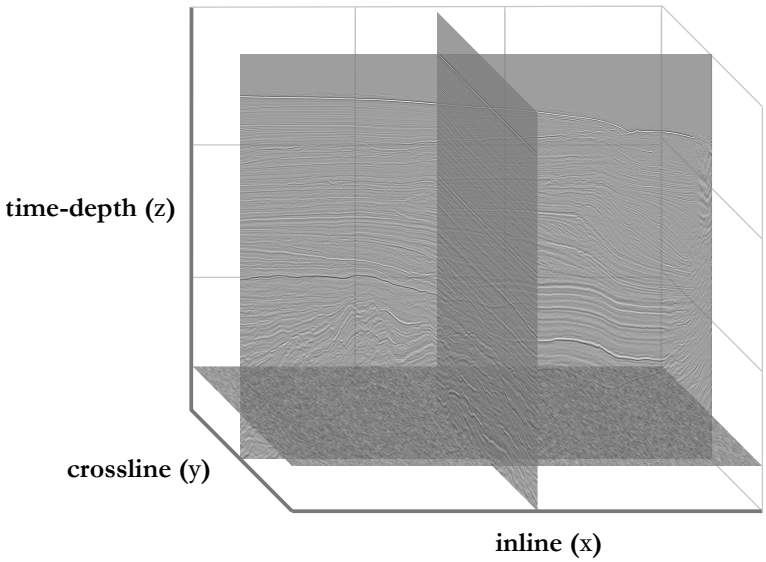
O processo de aquisição de dados sísmicos baseia-se em amostragem de sinais que medem as condições físicas do mundo real e as convertem em amostras numéricas que podem ser manipuladas por um computador (EVANS, 1997). Para isso, é necessário que ondas sonoras deixem a fonte e viajem para baixo da superfície da terra, onde serão encontradas as mudanças das camadas geológicas, resultando na reflexão dessas ondas sonoras até a superfície. Os receptores detectam os ecos e os convertem em sinais elétricos, os quais são processados e utilizados para produzir imagens das estruturas. Com essas imagens, é possível determinar qual tipo de rocha elas representam, além de poderem ser usadas para identificar se as rochas possuem algum recurso valioso.

Os avanços nos métodos de aquisição, em particular na qualidade dos sensores (DONDURUR, 2018), resultaram em um aumento significativo no tamanho dos dados sísmicos. Devido à alta qualidade do sinal capturado, os volumes podem exigir centenas de terabytes para serem armazenados e transmitidos. Dessa forma, é necessário aplicar algoritmos de compressão para tentar reduzir o tamanho necessário para utilizá-los. Existem dois tipos de dados sísmicos: pré-pilha e pós-pilha (VEEKEN; SILVA, 2004). O pré-pilha possui o dado bruto, onde é possível encontrar várias informações redundantes, permitindo alternativas para sua compressão. A principal desvantagem desse tipo é a complexidade para lidar com suas diferentes fontes e métodos numéricos requeridos. O pós-pilha é uma versão processada, geralmente sem redundâncias, diferentemente do pré-pilha. Este trabalho foca apenas na compressão de dados sísmicos pós-pilha tridimensional (3D) representados na forma matricial.

Dados sísmicos 3D são obtidos pela aquisição em linhas paralelas próximas entre si. Um volume 3D é gerado após a junção dessas linhas paralelas e os pontos de captura na direção perpendicular. A direção utilizada durante a captura é chamada de *inline* (em linha), e a direção perpendicular à direção *inline* é chamada de *crossline* (linha cruzada) (YILMAZ, 2001). Ocasionalmente, o dado sísmico 3D precisa ser interpolado na direção *crossline* para evitar serrilhamento. Com isso, o volume sísmico 3D passa a possuir as direções *inline* e *crossline* nos eixos verticais, e a direção *time-depth* (tempo-profundidade) no eixo horizontal. A Figura 1 ilustra um exemplo das direções de um volume sísmico 3D.

Compressão de dados consiste em reduzir o número de bits necessários para

Figura 1 – Exemplo das direções de um dado sísmico pós-pilha 3D.



Fonte: Elaborada pelo autor (2022).

representar o dado, tornando o armazenamento e transmissão mais viável (LELEWER; HIRSCHBERG, 1987). Aplicações que recorrem à compressão para transmissão são especialmente desafiadoras por serem em tempo real. Porém, as aplicações com foco em armazenamento permitem que seja feita uma redução maior, mesmo que exija mais tempo para comprimir/descomprimir. Existem dois tipos de métodos de compressão: a com perda e a sem perda de dados (JAYASANKAR; THIRUMAL; PONNURANGAM, 2021). Na compressão sem perda, o dado resultante da descompressão deve ser uma réplica exata da imagem original. Esse tipo de compressão possui diversas aplicações, como em documentos de negócios, arquivos médicos e radiografia digital, onde qualquer perda de informação pode resultar em um diagnóstico incorreto. A compressão com perda permite que informações do dado original sejam perdidas, tendo como objetivo reduzir drasticamente o tamanho do dado, mas conservando a qualidade da informação saliente na reconstrução. Este trabalho concentra-se em uma compressão extrema de dados sísmicos pós-pilha 3D, sendo assim, é utilizado a compressão com perda para permitir que alta taxas de compressão sejam atingidas.

Recentemente, técnicas de aprendizado profundo (*deep learning*) têm sido aplicadas de forma eficiente em problemas da área de Visão Computacional (VC). Alguns problemas que tiveram grande auxílio dessas técnicas são detecção de objetos (FELZENSZWALB et al., 2009), super-resolução (DONG et al., 2014), classificação (YANG et al., 2018; ZHONG; LIU; LIU, 2011) e compressão de imagens (MENTZER et al., 2018). Além disso, rede adversária generativa (*Generative Adversarial Network* – GAN) foi introduzida por Goodfellow et al. (2014) como uma alternativa promissora para aprendizado generativo devido à sua capacidade de aprender distribuições intratáveis (AGUSTSSON et al., 2019;

WANG et al., 2018). Esse método foi utilizado por Agustsson et al. (2019) ao combinar um autocodificador com um módulo discriminador para comprimir e descomprimir imagens naturais.

Na área de Geofísica, o aprendizado profundo tem sido utilizado para lidar com problemas como inversão (ZHANG; ALKHALIFAH, 2019), predição de fácies sísmicas (BESTAGINI; LIPARI; TUBARO, 2017), detecção de falhas sísmicas (WU et al., 2019) e compressão (SCHIAVON et al., 2020). Focando em apenas compressão, Schiavon et al. (2019) propuseram uma abordagem baseada em autocodificadores profundos. Esse método adapta a rede neural convolucional (*Convolutional Neural Network* – CNN) proposta por Mentzer et al. (2018) para comprimir fatias sísmicas bidimensionais (2D). Neste trabalho, denominamos como fatia uma sub-região 2D extraída do volume de dados sísmicos 3D. Após a extração das fatias, elas são comprimidas separadamente. Então, todo o volume é reconstruído, alcançando valores elevados de razão sinal-ruído de pico (*Peak Signal-to-Noise Ratio* – PSNR) para taxas de bits baixas. Em adição, no trabalho de Schiavon (2020), é apresentado o método *3D-based Seismic Data Compression* (3DSC) que expande essa ideia ao utilizar convoluções 3D para explorar a similaridade entre as fatias vizinhas. A principal vantagem do uso de convoluções 3D é capturar correlação em todas as direções, mantendo as mesmas dimensões de representação latente de sua contraparte 2D.

Devido aos resultados satisfatórios obtidos por Agustsson et al. (2019) ao utilizar GANs para compressão de imagens, neste trabalho é proposto uma integração do método 3DSC e uma GAN para compressão de dados sísmicos pós-pilha 3D. A hipótese principal é de que a compressão de um subvolume sísmico 3D pode ser aperfeiçoado pelo treinamento adversário em uma GAN. Isso não é evidente, visto que apenas o 3DSC não captura informação volumétrica suficiente para superar suas contrapartes 2D, como o método *Multichannel Seismic Data Compression* (MCSC) (SCHIAVON et al., 2020). Entretanto, teoricamente, subvolumes 3D devem permitir uma melhor estimativa de redundância, se comparado a sub-regiões, ou fatias, 2D comprimidas separadamente.

Em trabalhos recentes (NAVARRO et al., 2019; SCHIAVON et al., 2019; SCHIAVON, 2020; SCHIAVON et al., 2020), o PSNR é utilizado para avaliar a qualidade das reconstruções, como função de distorção, sugerindo que todos os eixos dos dados sísmicos possuem a mesma importância durante a avaliação da perda. Sendo assim, o subvolume 3D é achatado e considerado como um vetor unidimensional (1D) pela função de perda, ignorando as propriedades originais dos eixos principais. Com isso em mente, neste trabalho também é proposta uma nova função de distorção para dados sísmicos 3D. A ideia consiste em aplicar uma função de perda diferente para cada eixo, resultando em uma redução de dimensionalidade em cada eixo. Ou seja, a primeira redução gera um bloco 2D de uma entrada 3D, em seguida, é gerado um bloco 1D a partir do 2D e finalmente a terceira redução mapeia o vetor 1D em um escalar. Para isso, é proposta a utilização de uma função de perda 1D diferente em cada redução de dimensão, dependendo

do eixo escolhido.

1.1 DEFINIÇÃO DO PROBLEMA

O problema de compressão de dados sísmicos pós-pilha 3D com perda consiste em mapear um volume de entrada para uma representação com um custo de armazenamento menor, e em seguida tentar remapear essa representação para o domínio original. Esse problema pode ser definido como: dado um volume $\mathbf{v} \in \mathbb{R}^{H \times W \times D}$ de entrada, é necessário encontrar as funções:

$$f : \mathbb{R}^{H \times W \times D} \to \mathbb{R}^{N}, \tag{1.1}$$

$$q : \mathbb{R}^{N} \to \mathbb{C}^{N/M}, \tag{1.2}$$

e

$$g : \mathbb{C}^{N/M} \to \mathbb{R}^{H \times W \times D}, \tag{1.3}$$

de forma que $g(q(f(\mathbf{v}))) \approx \mathbf{v}$, onde $\mathbb{C} = \{c_1, \ldots, c_L\} \subset \mathbb{R}$. A representação latente quantizada $q(f(\mathbf{v}))$ tem a característica de necessitar menos armazenamento do que $\mathbf{v}$.

Na compressão com perda, o custo médio para representar $\mathbf{v}$ em uma forma comprimida é chamado taxa de bits $\mathbf{r}$. Nesse caso, é vantajoso que $q(f(\mathbf{v}))$ seja representado com o menor número de bits possível. Além disso, também é necessário que a reconstrução seja satisfatória para a análise por especialistas. A qualidade da reconstrução pode ser medida através de uma função de distorção $\mathbf{d}$. Essa função calcula a diferença entre o dado original e a reconstrução, de forma que o objetivo da compressão é minimizá-la. Ao minimizar o erro produzido durante a compressão e a quantidade de bits necessários, tem-se o balanceamento (*trade-off*) entre a distorção e a taxa de bits, dado por:

$$\mathbf{d}(\mathbf{v}, g(q(f(\mathbf{v})))) + \beta \mathbf{r}(q(f(\mathbf{v}))), \tag{1.4}$$

sendo β um escalar que controla a participação da taxa de bits $\mathbf{r}(q(f(\mathbf{v})))$ do sinal quantizado no balanceamento.

Nesta dissertação, é utilizado aprendizado profundo para comprimir dados sísmicos pós-pilha 3D. São utilizadas redes neurais profundas para encontrar as funções f, q e g. Assim como proposto por Mentzer et al. (2018), Schiavon (2020) propôs um sistema de compressão para dados sísmicos. Esse modelo possui módulos chamados: codificador, quantizador, e decodificador, que representam as funções f, q e g, respectivamente. A taxa de bits é estimada por outro módulo que calcula a distribuição de probabilidade conjunta da representação latente quantizada. Neste trabalho, o módulo que calcula a função de distribuição de probabilidade é chamado módulo de probabilidade. Com isso, o desafio é integrar o modelo 3DSC com uma GAN e definir uma função de perda composta por reduções em cada eixo 3D. A meta é aprimorar a reconstrução qualitativamente.

1.2 OBJETIVOS

Este trabalho possui dois objetivos principais para melhorar a eficácia e qualidade de reconstrução de dados sísmicos através de aprendizado profundo. O primeiro deles é integrar a estrutura de uma GAN com o método 3DSC proposto por Schiavon (2020) para melhor explorar a redundância volumétrica presente nos dados sísmicos pós-pilha 3D. Inclusive, vários experimentos são realizados para avaliar a eficácia do método e, por fim, os resultados obtidos são comparados com os métodos da literatura. O segundo objetivo é encontrar uma função de distorção para ser utilizada em dados multidimensionais, como dados sísmicos. A finalidade dessa nova função de perda composta é considerar a estrutura do dado original durante o cálculo do erro de reconstrução. Portanto, é feito um estudo extensivo em relação às possíveis combinações de funções para serem aplicadas no problema de compressão de dados sísmicos pós-pilha 3D.

1.3 CONTRIBUIÇÕES

As principais contribuições deste trabalho são:

- Integração de uma CNN 3D e uma GAN para compressão de dados sísmicos pós-pilha 3D, na qual resultou em uma publicação no IEEE *Geoscience and Remote Sensing Letters* (RIBEIRO et al., 2022);
- Uma nova forma para calcular a função de perda composta em dados multidimensionais, onde a informação nos eixos possui grandeza e importância distintas;
- Um estudo extensivo sobre possíveis combinações de funções para definir a função de perda composta;
- Ganhos consistentes em relação à literatura.

1.4 ORGANIZAÇÃO

Este trabalho está estruturado da seguinte forma: O Capítulo 2 apresenta os trabalhos presentes na literatura que se relacionam com esta dissertação e descreve os conceitos fundamentais que compõem os métodos propostos. O Capítulo 3 apresenta os métodos propostos neste trabalho. Todos os experimentos e resultados estão descritos no Capítulo 4. O Capítulo 5 apresenta as conclusões obtidas, além de trabalhos futuros.

2 TRABALHOS RELACIONADOS E FUNDAMENTAÇÃO TEÓRICA

Este capítulo apresenta os trabalhos na literatura que se relacionam com este trabalho, além dos fundamentos que constituem os métodos propostos. A Seção 2.1 aborda os trabalhos que se relacionam com compressão de dados sísmicos e imagens e a Seção 2.2 apresenta os dois principais fundamentos que suportam as propostas desta dissertação: o método 3DSC para compressão de dados sísmicos, e o método de redes neurais generativas adversárias (GANs).

2.1 TRABALHOS RELACIONADOS

Esta seção apresenta os trabalhos da literatura relacionados com compressão de dados sísmicos e imagens. É importante enfatizar que no limite do nosso conhecimento, trabalhos que especificamente comprimem dados sísmicos através de redes neurais não foram encontrados além das publicações do nosso grupo de pesquisa. A Subseção **2.1.1** aborda os métodos de compressão utilizados para compressão de dados sísmicos enquanto a Subseção **2.1.2** trata das abordagens baseadas em aprendizado profundo para comprimir imagens.

2.1.1 Compressão de dados sísmicos

O problema de compressão de dados sísmicos tem sido abordado por diversos métodos. As abordagens baseadas em transformação estão relacionadas principalmente com transformações fixas, onde o sinal original é representado em um novo domínio mais apropriado para compressão (LEE; YARLAGADDA, 1982; SPANIAS; JONSSON; STEARNS, 1991; VERMEER; BRAGSTAD; ORR, 1996). Além disso, existem também os métodos preditivos, sendo tradicionalmente utilizados para compressão de séries temporais, assim como fatias sísmicas (KIELY et al., 2010; NIJIM; STEARNS; MIKHAEL, 1996). Outras abordagens adaptam compressores estado-da-arte de imagem e vídeo para dados sísmicos, como *Joint Photographic Experts Group Extended Range* (JPEG-XR) e *High-Efficiency Video Coding* (HEVC)(LIU et al., 2016; RADOSAVLJEVIĆ et al., 2017).

O trabalho de Spanias, Jonsson e Stearns (1991) apresenta um estudo focado nas técnicas baseadas em transformadas para mapeamento de domínios. O estudo analisa o uso da Transformada Discreta de Fourier, Transformada Discreta do Cosseno, Tranformada de Walsh-Hadamard e Transformada de Karhunen-Loève. Os resultados mostraram que transformações sinusoidais conseguem atingir uma alta taxa de compressão para dados sísmicos. Além disso, a Transformada Discreta de Fourier e a Tranformada de Walsh-Hadamard entregaram um erro maior comparado as demais. A Transformada de Karhunen-Loève obteve o melhor resultado, mas é dependente do dado utilizado. Por outro lado, a Transformada Discreta do Cosseno teve uma eficácia similar, porém consistente

para todos os dados utilizados, entregando um baixo erro de reconstrução e uma alta taxa de compressão.

Uma abordagem baseada em métodos preditivos foi proposta por Stearns, Tan e Magotra (1993). Ela consiste em uma compressão de duas etapas, onde a primeira faz uso de um preditor linear para remover a correlação temporal entre as fatias e a segunda consiste em uma compressão sem perda do sinal residual. Os resultados obtidos indicam que a média de bits obtidos por fatia é próxima ao limite inferior, sendo uma taxa de compressão viável para aplicações de tempo real. Um método que usa uma predição mais sofisticada foi proposto por Magotra et al. (1995). Ele utiliza de um preditor adaptativo recursivo que possui 20 etapas. Apesar desse tipo de método possuir um custo computacional extra, ele geralmente é superior, em termos de taxa de compressão, comparado com as abordagens que utilizam preditor linear fixo.

Métodos que usam compressores originalmente designados para imagem e vídeo foram propostos para compressão de dados sísmicos. A abordagem proposta por Liu et al. (2016) consiste em utilizar o JPEG-XR para alcançar uma melhor eficácia comparada com os compressores utilizados comercialmente. Mesmo tratando-se de um compressor padrão de imagens, o JPEG-XR possui diversos aspectos que incluem compressão com e sem perda, quantização adaptativa, decodificação de região de interesse, e outras. O método proposto por Liu et al. (2016) tem o objetivo de utilizar a compressão de baixa complexidade que o JPEG-XR proporciona, em fatias sísmicas 1D e 2D. Além disso, foi proposto um algoritmo para controlar a qualidade da reconstrução, utilizando a relação sinal-ruído (*Signal-to-Noise Ratio* – SNR). Dessa forma, o fato dessa abordagem possuir uma baixa complexidade e ser de fácil manipulação, faz com que ela seja mais atrativa comercialmente. Outro método que explora compressores padrões para dados sísmicos foi proposto por Radosavljević et al. (2017). Esse trabalho aproveita-se do HEVC para comprimir dados sísmicos 3D de modo a explorar a redundância temporal presente no dado. Para isso, o dado 3D é considerado como uma sequência de imagens que são entregues para o HEVC. O método usa a estimação de movimento do HEVC para aplicar a codificação preditiva. Essa extensão elimina redundância entre as fatias encontrando blocos deslocados dentro de um quadro de referência.

O trabalho de Zizi e Turquais (2022) apresenta uma abordagem baseada em aprendizado de dicionário (*dictionary learning*) para compressão de dados sísmicos. O método armazena a forma da redundância do dado em um único coeficiente esparso. Nele são apresentados dois modos de parametrização distintos, dependendo da aplicação desejada na área de Geofísica. Os resultados indicam que o método atinge uma taxa de compressão superior para os volumes do conjunto de teste comparado com os algoritmos do pacote *Seismic Unix* e o programa *zfp*. Foram realizados experimentos que simulam aplicações em tempo real, mostrando que o método tem uma ligeira perda de informação em fatias 2D para taxas de compressão superiores a 20:1. Nas demais aplicações, os

resultados indicam que o método preserva o aspecto visual do dado mesmo com uma taxa de compressão de 95:1.

Um método baseado em decomposição de matriz deslocada (*Shifted-Matrix Decomposition*) é proposto por Brankovic et al. (2021) para comprimir e eliminar o ruído dos dados sísmicos. Para isso, os dados são armazenados em pares de vetores acoplados a um vetor de deslocamento com intuito de extrair um par de vetores únicos, demandando menos memória para ser armazenado. A compressão e eliminação do ruído são realizadas simultaneamente. Ao reconstruir o dado sísmico, o método cria uma versão sem ruído do dado original. Experimentos foram realizados em dados com ruído sintético, onde o método aumentou o valor de SNR para taxas de compressão de 5:1 e 20:1.

Recentemente, métodos que aplicam aprendizado profundo ao problema de compressão de dados sísmicos têm sido propostos (NAVARRO et al., 2019; SCHIAVON et al., 2019; SCHIAVON et al., 2020). A abordagem proposta por Schiavon et al. (2019) consiste em estender o trabalho de Mentzer et al. (2018) para comprimir fatias sísmicas 2D. O método usa duas redes neurais treinadas simultaneamente: um autocodificador para gerar e reconstruir a representação latente, e um módulo de probabilidade para estimar a entropia. A avaliação dos resultados foi feita em dados sísmicos reais alcançando uma alta taxa de compressão e um PSNR médio de 40 decibéis (dB). Pretendendo atingir uma taxa compressão mais alta, Schiavon (2020) propôs o uso de subvolumes 3D para compressão, de modo a explorar a similaridade entre a vizinhança delas. Essa abordagem preserva a qualidade de reconstrução e aumenta a taxa de compressão comparado com o método proposto por Schiavon et al. (2019).

Compressores baseados em aprendizado profundo têm potencial de superar as abordagens tradicionais, entregando uma reconstrução com qualidade visual superior, mesmo com taxa de bits extremas. Com isso, nesta dissertação, métodos focados em aprendizado profundo são aplicados em dados sísmicos para uma compressão extrema.

2.1.2 Compressão de imagens com aprendizado profundo

O problema de compressão de imagens utilizando aprendizado profundo tornou-se popular devido aos resultados obtidos por diversos métodos em problemas da área de VC. As arquiteturas mais populares são, autocodificadores (AGUSTSSON et al., 2019; BALLÉ; LAPARRA; SIMONCELLI, 2016; MENTZER et al., 2018; VANKDOTHU; HAMEED, 2022) e redes neurais recorrentes (*Recurrent Neural Networks* – RNNs) (CHOWDARY; KALAIYARASI; SARAVANAN, 2022; TODERICI et al., 2015; TODERICI et al., 2017). Esses modelos de rede aplicam uma redução de dimensionalidade e transformam a entrada em uma sequência de bits, que será comprimida usando métodos de codificação de entropia, como Huffman ou codificação aritmética.

Os métodos *Better Portable Graphics* (BPG) e *Joint Photographic Experts Group*

2000 (JPEG2000), apesar de não serem baseados em aprendizado profundo, são geralmente utilizados para comparação. Isso é devido ao fato desses compressores de propósito geral alcançarem resultados competitivos em termos de PSNR e taxa de compressão. O BPG é um formato de arquivo proposto por Bellard (2014) para compressão de imagens. Ele baseia-se no codec HEVC, atingindo uma compressão mais eficiente, em termos de qualidade de imagem e tamanho do arquivo, comparado com a entregue pelo *Joint Photographic Experts Group* (JPEG). O JPEG2000 é um compressor padrão de imagens capaz de alcançar uma qualidade de compressão superior ao JPEG (TAUBMAN; MARCELLIN, 2002). Ele utiliza a Transformada Discreta de Wavelets e de uma otimização no trucamento do bloco de codificação para melhorar a eficiência de compressão do seu antecessor. Com isso, ambos compressores, BPG e JPEG2000, são selecionados para comparação nesta dissertação.

Os primeiros resultados promissores usando aprendizado profundo para compressão de imagem foram atingidos pelo método proposto por Ballé, Laparra e Simoncelli (2016). A abordagem consiste em uma transformação de análise e síntese não linear e um quantizador uniforme. Por tratar-se de um autocodificador, temos que a transformação de análise é do módulo codificador, onde é feito a compressão da imagem de entrada, e a transformação de síntese trata-se do módulo decodificador que faz a descompressão. As transformações são constituídas por camadas convolucionais com funções de ativação. O modelo utiliza uma variação do gradiente descendente estocástico para otimizar considerando a taxa de distorção obtida na base de dados de treinamento. Experimentos mostraram que o método possui uma melhor eficácia comparado ao JPEG e JPEG2000 em termos de PSNR e similaridade estrutural multi-escala (*Multi-Scale Structural Similarity* – MS-SSIM) para a maior parte da base de dados utilizada, obtendo uma reconstrução visual superior.

O método proposto por Ballé et al. (2018) baseia-se em autocodificadores variacionais (*Variational Autoencoders* — VAEs). O modelo utiliza uma distribuição de probabilidade *a priori* nos hiperparâmetros com objetivo de capturar dependências espaciais na representação latente. A abordagem consiste em uma extensão direta do método apresentado por Ballé, Laparra e Simoncelli (2016) ao utilizar uma distribuição *a priori*. Esse método pode ser otimizado pelas perdas de distorção que são mais complexas do que métricas baseadas em pixel como erro quadrático médio (*Mean Squared Error* – MSE). Os resultados mostraram que essa abordagem supera o estado-da-arte quando comparado utilizando a métrica MS-SSIM, mas não atinge uma eficácia semelhante ao do BPG em termos de PSNR.

A abordagem proposta por Toderici et al. (2017) usa RNNs para o módulo codificador e decodificador. Eles comparam a eficácia dos tipos de RNNs, como, *Long Short-Term Memory* (LSTM) e LSTM associativa, e inclusive apresentam uma mescla da *Gated Recurrent Unit* (GRU) com a rede neural residual (*Residual Neural Network* – ResNET). A imagem de entrada é primeiramente codificada, em seguida é passada para o módulo binarizador para gerar a representação em códigos binários, e então é decodificado.

Esse modelo possui um diferencial que permite entregar uma taxa de compressão variável utilizando uma rede treinada apenas uma única vez.

Uma otimização para o problema de compressão de imagens com perda utilizando autocodificadores foi proposta por Theis et al. (2017). A derivada de arredondamento utilizada na retropropagação (*backpropagation*) foi substituída por uma derivada de uma aproximação, permitindo que a quantização fosse aplicada. Também foi utilizado um treinamento incremental, onde uma rede treinada é sujeita a um ajuste fino no parâmetro β da equação da taxa de distorção, permitindo que o modelo comprima com outras taxas de bits sem a necessidade de treinar novamente. Em termos de qualidade perceptual, essa abordagem consegue ser similar ou até superar o JPEG2000 na base de dados utilizada para teste.

Mentzer et al. (2018) propuseram um novo modo de controlar o balanceamento entre a distorção e a taxa de bits durante a otimização do autocodificador, modelando o cálculo da entropia da representação latente utilizando uma CNN 3D. Essa rede tem o objetivo de aprender um modelo de probabilidade condicional da distribuição latente do autocodificador. Além disso, seus pesos são atualizados durante o treinamento para aprender as dependências entre os símbolos na representação latente. Os resultados mostram que o método atinge o estado da arte para compressão de imagens em termos de MS-SSIM, superando BPG e JPEG2000 e métodos baseados em aprendizado profundo.

No trabalho de Agustsson et al. (2019) é apresentado um sistema de compressão de imagens baseado em GANs para taxas de bits extremamente baixas, baseado na arquitetura proposta no trabalho de Wang et al. (2018). O método consiste em um codificador, um decodificador/gerador e um discriminador multi-escala, treinados simultaneamente. Além disso, para manter um alto nível de detalhes, essa abordagem usa mapas semânticos, onde a rede gera as informações da imagem e preserva os dados originais em regiões definidas. Os modelos foram treinados com 188.000 imagens naturais da base de dados *Open Images* (KUZNETSOVA et al., 2020) e avaliados na base de dados de imagens da Kodak. Visto que métricas como PSNR ou MS-SSIM não são adequadas para esse tipo de problema, foi realizado um estudo de usuário para avaliar a qualidade perceptual das imagens reconstruídas. Os resultados mostraram que o método atinge uma compressão extrema, obtendo uma qualidade visual superior aos métodos otimizados por funções clássicas, como MS-SSIM e MSE.

Assim como o trabalho de Mentzer et al. (2018) foi adaptado para comprimir fatias sísmicas (SCHIAVON et al., 2019; SCHIAVON et al., 2020), é esperado que outras abordagens focadas em compressão de imagens possam alcançar resultados superiores ao serem modificados para comprimir dados sísmicos. Logo, nesta dissertação é explorado um modelo baseado em GANs, pois elas aprendem quais são as informações salientes do dado, resultando em uma reconstrução de qualidade visual superior aos métodos tradicionais.

2.2 FUNDAMENTAÇÃO TEÓRICA

Nesta seção, são apresentados os fundamentos que sustentam as propostas desta dissertação: o método 3DSC para compressão de dados sísmicos, e o método de redes neurais generativas adversárias (GANs). A Subseção **2.2.1** apresenta o método base desta pesquisa em detalhes (SCHIAVON, 2020). Além disso, a Subseção **2.2.2** aborda a teoria sobre as GANs, bem como os seus módulos e treinamento adversário.

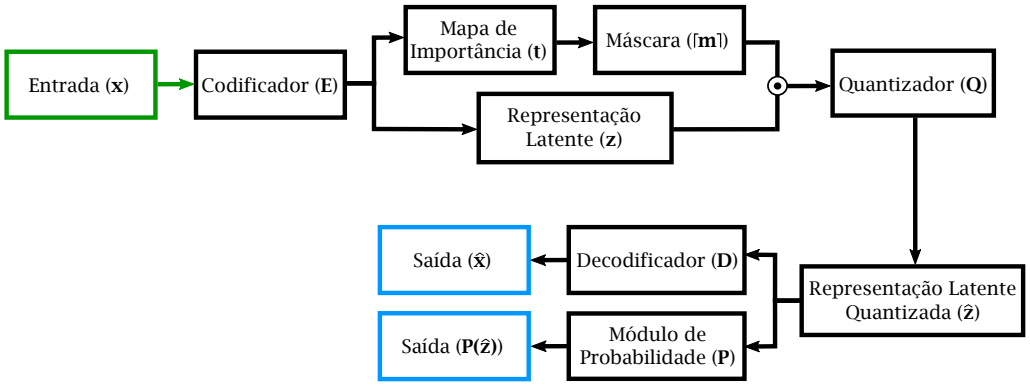
2.2.1 *3D-based Seismic Data Compression* (3DSC)

Schiavon (2020) propôs utilizar redes neurais profundas (*Deep Neural Networks* – DNNs) para compressão de dados sísmicos 3D pós-pilha. Sua principal hipótese era de que redes neurais poderiam comprimir esse tipo de dado com uma baixa taxa de bits mantendo as principais informações estruturais. Para isso, o método focado em compressão de imagens proposto por Mentzer et al. (2018) foi adaptado para compressão de dados sísmicos. Esse novo método chama-se 3DSC e sua entrada é um subvolume 3D composto por um conjunto de fatias 2D empilhadas. Além disso, são utilizadas camadas convolucionais 3D para que a rede tenha uma maior capacidade de encontrar correlação entre as fatias da entrada, auxiliando a rede a atingir uma taxa de compressão maior.

A Figura 2 apresenta uma visão geral do método 3DSC. Um dado sísmico é fornecido como entrada para ser comprimido pelo módulo codificador. Esse módulo gera uma representação latente e um mapa de importância. O mapa de importância é expandido para uma máscara que será combinada com a representação latente com objetivo de torná-la discreta. Em seguida, é aplicado uma quantização para discretizar os elementos dessa combinação, resultando na representação latente quantizada. Então, o dado discretizado é passado para o decodificador e para o módulo de probabilidade. O módulo decodificador é responsável por reconstruir a entrada, resultando na reconstrução. Por outro lado, o módulo de probabilidade calcula a distribuição de probabilidade conjunta da representação latente quantizada para estimar a taxa de bits necessária para representá-lo.

O método 3DSC se baseia em um autocodificador, assim como o proposto por Mentzer et al. (2018). Entretanto, ao invés da entrada da rede ser uma imagem RGB, ela é um subvolume sísmico $\mathbf{x} \in \mathbb{R}^{h \times w \times d}$, onde h e w são as dimensões espaciais e d representa o número de fatias 2D que o compõe. O módulo codificador $\mathbf{E} : \mathbb{R}^{h \times w \times d} \rightarrow \mathbb{R}^{h/8 \times w/8 \times C}$ mapeia a entrada $\mathbf{x}$ em uma representação latente $\mathbf{z} = \mathbf{E}(\mathbf{x})$. A redução que ocorre nas dimensões espaciais de $\mathbf{x}$ é devido ao fator de subamostragem da rede. O parâmetro C indica o número de mapas de características (*feature maps*) obtidos ao aplicar C filtros na entrada da última camada convolucional de $\mathbf{E}$. O quantizador $\mathbf{Q} : \mathbb{R}^{h/8 \times w/8 \times C} \rightarrow \mathcal{C}^{h/8 \times w/8 \times C}$ discretiza os valores de $\mathbf{z}$, resultando na representação latente quantizada $\hat{\mathbf{z}} = \mathbf{Q}(\mathbf{z})$. Então, o decodificador $\mathbf{D} : \mathcal{C}^{h/8 \times w/8 \times C} \rightarrow \mathbb{R}^{h \times w \times d}$ reconstrói a entrada, gerando o dado reconstruído $\hat{\mathbf{x}} = \mathbf{D}(\hat{\mathbf{z}})$.

Figura 2 – Fluxograma do método 3DSC.



Fonte: Elaborada pelo autor (2022).

Imagens naturais possuem uma grande variação espacial, com isso, existem regiões delas que necessitam mais bits do que outras. Um exemplo disso é uma imagem com um céu azul de fundo e um rosto. O primeiro necessita de poucos bits para ser representado e o último precisa de mais bits para preservar os detalhes. Isso também ocorre com os dados sísmicos, já que existem regiões que possuem mais informações estruturais com altas frequências do que outras. Com isso, assim como no trabalho de Mentzer et al. (2018), é adaptado o mapa de importância proposto por Li et al. (2018) para auxiliar a rede com a alocação de bits. Li et al. (2018) propôs que a geração desse mapa fosse feita por uma rede separada. Entretanto, Mentzer et al. (2018) simplificou essa ideia ao adicionar o mapa de importância $\mathbf{t} \in \mathbb{R}^{h/8 \times w/8 \times 1}$ à saída da última camada do codificador. Em seguida, com o objetivo de produzir mais entradas nulas na representação latente $\mathbf{z}$, $\mathbf{t}$ é expandido para uma máscara $\mathbf{m} \in \mathbb{R}^{h/8 \times w/8 \times C}$ na seguinte forma:

$$m_{i,j,k} \begin{cases} 1 & \text{se } k < t_{i,j} \\ (t_{i,j} - k) & \text{se } k \leq t_{i,j} \leq k+1 \\ 0 & \text{se } k+1 > t_{i,j}, \end{cases} \tag{2.1}$$

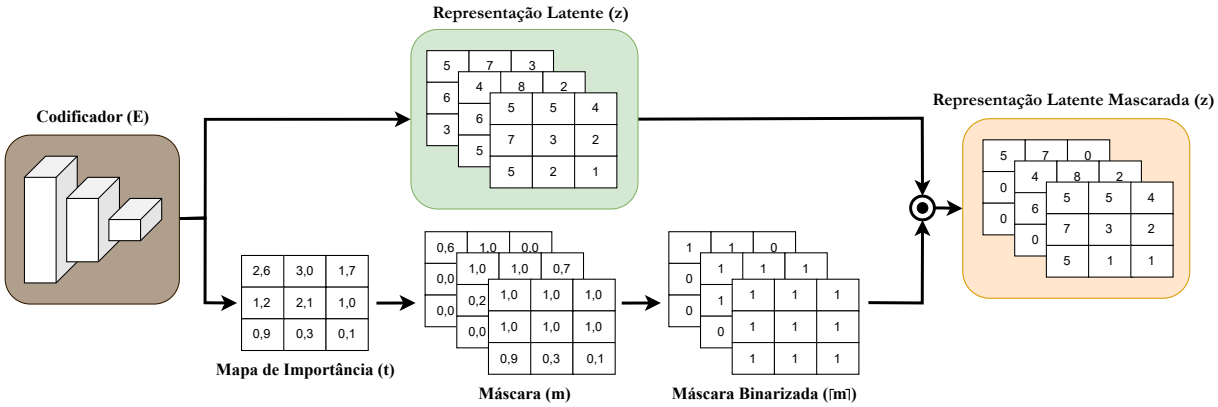
onde $t_{i,j}$ representa o valor de $\mathbf{t}$ na posição (i, j). Em seguida, é realizado o processo de binarização de $\mathbf{m}$, resultando na máscara binarizada $\lceil \mathbf{m} \rceil$. Então, $\mathbf{z}$ é mascarado ao ser multiplicado ponto a ponto com $\lceil \mathbf{m} \rceil$, ou seja:

$$\mathbf{z} \leftarrow \mathbf{z} \odot \lceil \mathbf{m} \rceil. \tag{2.2}$$

Dessa forma, visto que a função teto $\lceil . \rceil$ não é diferenciável, a identidade é utilizada durante a retropropagação. A Figura 3 ilustra esse processo de mascaramento feito ao combinar $\lceil \mathbf{m} \rceil$ e $\mathbf{z}$.

A quantização é a mesma aplicada por Mentzer et al. (2018), onde é feita uma adaptação no método de Agustsson et al. (2017) utilizando ideias de Theis et al. (2017). O processo de quantização consiste em associar um conjunto finito de centroides $\mathcal{C} =$

Figura 3 – Processo de mascaramento da representação latente.



Fonte: Adaptação de Schiavon (2020).

$\{c_1, \cdots, c_L\} \subset \mathbb{R}$ aprendidos pela rede com os elementos da representação latente $\mathbf{z}$. A associação é feita por meio do vizinho mais próximo, definida por:

$$\hat{z}_i = Q(z_i) := \arg\min_j \|z_i - c_j\|, \tag{2.3}$$

onde $c_j \in \mathcal{C}$. Uma quantização suave é utilizada durante a retropropagação, visto que arg min não é diferenciável, dada por:

$$\hat{z}_i = \sum_{j=1}^{L} \frac{\exp(-\sigma\|z_i - c_j\|)}{\sum_{l=1}^{L} \exp(-\sigma\|z_i - c_l\|)} c_j = \sum_{j=1}^{L} \text{softmax}(-\sigma\|z_i - c_j\|)c_j. \tag{2.4}$$

Assim, a quantização é realizada em um conjunto finito e aprendido de centroides como proposto por Agustsson et al. (2017), e não em um conjunto fixo (não aprendido) de inteiros, proposto por Theis et al. (2017). Além disso, a quantização suave proposta por Theis et al. (2017) é mais simples, sendo aplicada apenas durante a retropropagação, ao invés de optar por escolher um cronograma de resfriamento (*annealing*) para σ, como proposto por Agustsson et al. (2017).

A estimativa do número de bits necessários para representar o subvolume sísmico comprimido é feita ao calcular a distribuição de probabilidade conjunta p da representação latente quantizada $\hat{\mathbf{z}}$. Para isso, a distribuição $p(\hat{\mathbf{z}})$ é fatorada em um produto de distribuições condicionais, definida por:

$$p(\hat{\mathbf{z}}) = \prod_{i=1}^{n} p(\hat{z}_i | \hat{z}_{i-1}, \ldots, \hat{z}_1), \tag{2.5}$$

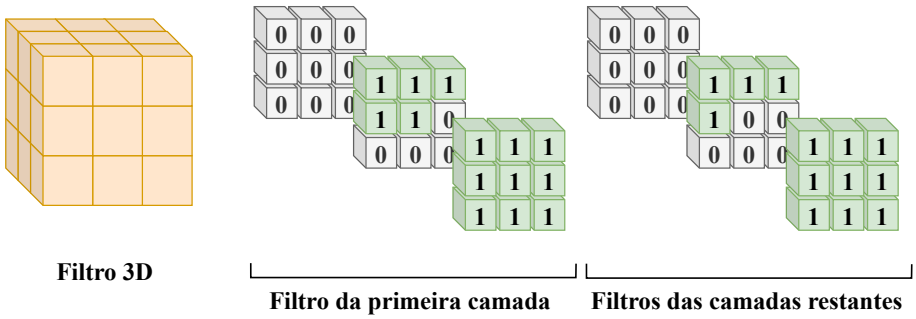
onde $n = w/8 \cdot h/8 \cdot C$. Em seguida, o módulo de probabilidade $\mathbf{P}$ é usado para estimar a distribuição condicional de $p(\hat{\mathbf{z}})$. Logo, $\mathbf{P}$ é treinada para predizer as probabilidades de classificar $\hat{z}_i$ para cada centroide c_j de $\mathcal{C}$ de acordo com suas similaridades. Com isso, uma CNN 3D $\mathbf{P}$ prediz cada termo $p(\hat{z}_i | \hat{z}_{i-1}, \ldots, \hat{z}_1)$, denotada por:

$$P_{i,l}(\hat{\mathbf{z}}) \approx p(\hat{z}_i = c_l | \hat{z}_{i-1}, \ldots, \hat{z}_1), \tag{2.6}$$

onde $P_{i,l}$ é a probabilidade de associar cada $\hat{z}_i$ para cada c_l.

Note que as distribuições condicionais $p(\hat{z}_i|\hat{z}_{i-1}, \ldots, \hat{z}_1)$ apenas dependem dos valores anteriores $\hat{z}_{i-1}, \ldots, \hat{z}_1$ e a rede neural precisa lidar com essa restrição. A proposta feita por Mentzer et al. (2018) foi de utilizar filtros mascarados nas convoluções, assim como os estudados nos trabalhos da rede PixelCNN (OORD; KALCHBRENNER; KAVUKCUOGLU, 2016; OORD et al., 2016). Com isso, apenas a região ao redor do voxel selecionado é considerada. A Figura 4 ilustra a proposta de Mentzer et al. (2018) para filtros 3D de dimensões $3 \times 3 \times 3$. A proposta consiste em aplicar um filtro distinto na primeira camada convolucional e outro nas camadas restantes da rede **P**.

Figura 4 – Composição dos filtros das camadas convolucionais do módulo de probabilidade.

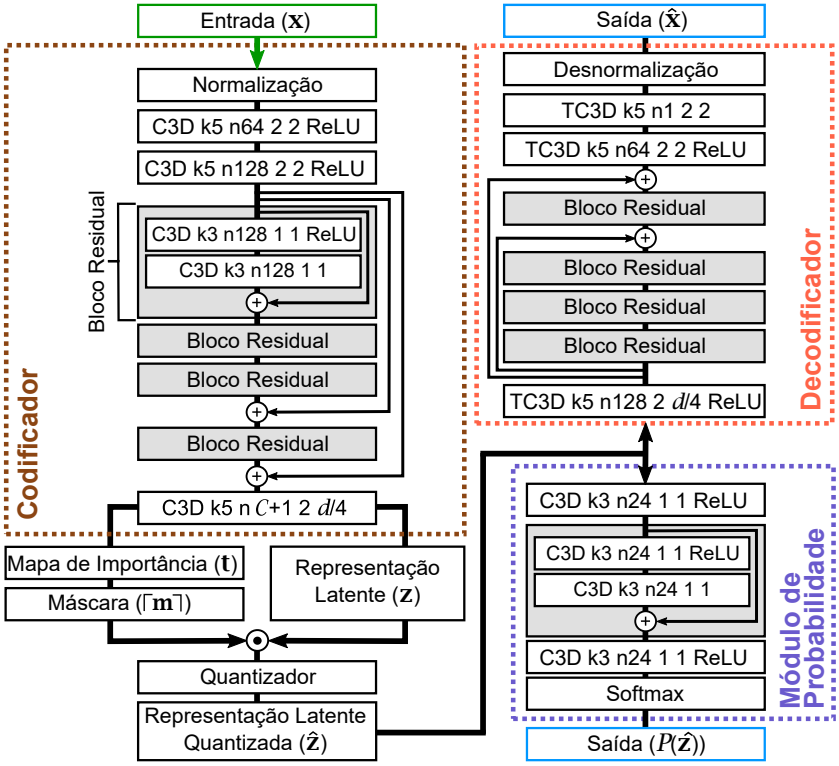


Fonte: Adaptação de Schiavon (2020).

A arquitetura da rede 3DSC é apresentada na Figura 5. Um conjunto de fatias de entrada **x** é fornecido como entrada do módulo codificador **E**, onde é aplicado uma normalização em **x** utilizando a média e o desvio-padrão, denotado pelo bloco "Normalização". Em seguida, "C3D k5 n64 2 2 ReLU" indica uma convolução 3D com 64 filtros de tamanho 5, salto 2 em todas as dimensões e a unidade linear retificada (*Rectified Linear Unit* – ReLU) é utilizada como função de ativação. Após isso, quatro blocos residuais são utilizados e uma combinação dos resultados é feita antes e após o último bloco. A estrutura das camadas de cada bloco é ilustrada no primeiro bloco residual de **E**. A última camada desse módulo aplica uma convolução 3D com $C+1$ filtros e salto $d/4$ na dimensão profundidade (*time-depth*), resultando em uma representação latente **z** e um mapa de importância **t**. O parâmetro C representa o número de mapas de características de **z**. Com isso, **z** é combinado com a máscara $\lceil \mathbf{m} \rceil$ gerada através de **t** e quantizado no módulo quantizador **Q**, gerando a representação latente quantizada $\hat{\mathbf{z}}$. O módulo decodificador **D** e o módulo de probabilidade **P** recebem essa representação como entrada. **D** espelha as camadas de **E** para expandir e reconstruir **x**. Para isso, são aplicadas convoluções transpostas, denotadas por "TC3D". O bloco "Desnormalização" aplica a operação inversa do bloco "Normalização". Por fim, cabe ao módulo **P** estimar o total de bits necessários para representar $\hat{\mathbf{z}}$. A camada "Softmax" gera a distribuição de probabilidade para cada

elemento de $\hat{\mathbf{z}}$.

Figura 5 – Arquitetura do método 3DSC.



Fonte: Elaborada pelo autor (2022).

A função de perda utilizada para treinar o módulo de probabilidade $\mathbf{P}$ é dada por:

$$\mathcal{L}_P = \mathbf{d}(\mathbf{x}, \hat{\mathbf{x}}) - \beta \frac{1}{n} \sum_{i=1}^{n} \sum_{l=1}^{L} \mathbf{y}_{i,l} \log P_{i,l}(\hat{\mathbf{z}}), \tag{2.7}$$

onde, assim como na Equação (1.4), $\mathbf{d}(\mathbf{x}, \hat{\mathbf{x}})$ é a função de distorção que calcula diferença entre $\mathbf{x}$ e $\hat{\mathbf{x}}$. O termo negativo é a perda de entropia cruzada (*cross-entropy*) que calcula o número de bits necessários para representar $\hat{\mathbf{z}}$ para um dado conjunto de rótulos $\mathbf{y}$. A Figura 6 ilustra a estimativa de bits feita por $\mathbf{P}$.

A abordagem proposta por Mentzer et al. (2018) utiliza MS-SSIM como função de distorção $\mathbf{d}(\mathbf{x}, \hat{\mathbf{x}})$. Essa métrica é bastante utilizada em problemas de compressão de imagens por ser uma comparadora perceptual. Entretanto, a função utilizada no método 3DSC é o PSNR, definida por:

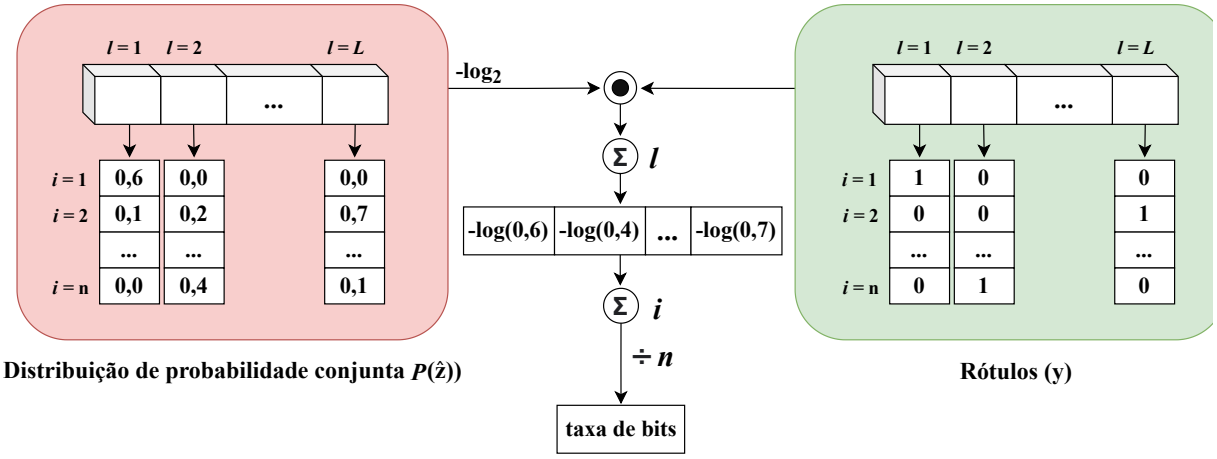
$$\text{PSNR}(\mathbf{x}, \hat{\mathbf{x}}) = 10 \cdot \log_{10} \frac{\text{MAX}^2}{\text{MSE}(\mathbf{x}, \hat{\mathbf{x}})}, \tag{2.8}$$

onde MAX representa o valor máximo dos dados sísmicos utilizados e a função MSE é dada por:

$$\text{MSE}(\mathbf{x}, \hat{\mathbf{x}}) = \sum_{i=1}^{h} \sum_{i=1}^{w} \sum_{i=1}^{d} (\mathbf{x}_{i,j,k} - \hat{\mathbf{x}}_{i,j,k})^2. \tag{2.9}$$

Visto que o PSNR é uma métrica genérica utilizada para avaliar a similaridade entre dois campos escalares, ela é utilizada para medir o erro dos subvolumes sísmicos.

Figura 6 – Estimativa do número de bits feita pelo módulo de probabilidade.



Fonte: Adaptação de Schiavon (2020).

O autocodificador, composto pelo módulo codificador $\mathbf{E}$ e decodificador $\mathbf{D}$, e o conjunto de centroides $\mathcal{C}$ utilizados para inicializar o quantizador $\mathbf{Q}$ são treinados com a função de perda definida por:

$$\mathcal{L}_{\mathbf{E},\mathbf{D},\mathbf{Q}} = \mathbf{d}(\mathbf{x},\hat{\mathbf{x}}) - \beta \underbrace{\frac{1}{n}\sum_{i=1}^{n}\sum_{l=1}^{L} \lceil m_i \rceil \, \mathbf{y}_{i,l} \log P_{i,l}(\hat{\mathbf{z}})}_{\mathbf{r}(\hat{\mathbf{z}})} + R_{AE} \cdot \ell_2(\mathbf{W}_{AE}) + R_{\mathcal{C}} \cdot \ell_2(\mathcal{C}), \quad (2.10)$$

onde $\mathbf{P}$ é utilizado para estimar a taxa de bits no balanceamento da Equação (1.4). A função ℓ_2 representa a perda de regularização e os parâmetros R_{AE} e $R_{\mathcal{C}}$ são os fatores de regularização do autocodificador e do conjunto de centroides, respectivamente. Além disso, a matriz de pesos do autocodificador é representada por $\mathbf{W}_{AE}$. A regularização é aplicada para prevenir sobre-ajuste (*overfitting*) do modelo. As probabilidades de $P(\hat{\mathbf{z}})$ são ponderadas com a máscara $\lceil \mathbf{m} \rceil$, proporcionando uma forma simples de controlar a entropia de $\hat{\mathbf{z}}$. Ao aumentar/diminuir os valores do mapa de importância $\mathbf{t}$, é possível obter menos ou mais entradas de valor zero na máscara.

2.2.2 Redes Adversárias Generativas

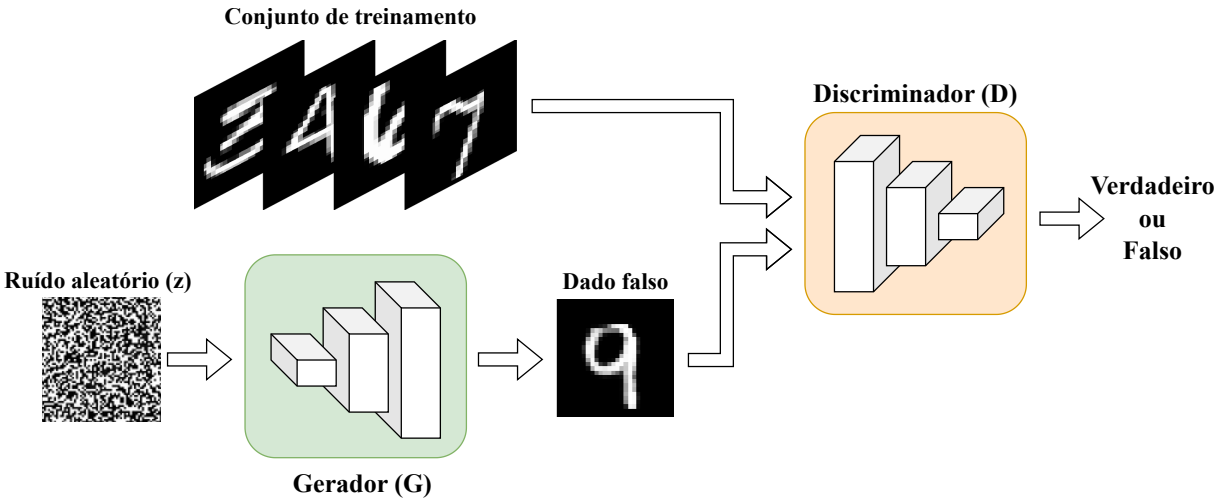
A ideia de aproveitar o treinamento de duas redes neurais concorrentes para um objetivo final foi proposta por Goodfellow et al. (2014). As GANs pertencem à classe de modelos generativos, sendo capazes de aprender a simular qualquer distribuição de dados. Em outras palavras, elas são ensinadas a elaborar/gerar um conteúdo novo. As GANs podem ser ensinadas a criar imagens, vídeos, fala, músicas e mais. Por isso, esse método é

considerado possuir um grande potencial em aplicações na área de VC. Alguns exemplos de aplicações são: síntese de texto para imagem (TAO et al., 2022), de imagem para texto (GUO et al., 2022), envelhecimento da face (WU et al., 2022) e classificação de objetos com nuvem de pontos 3D (HUANG; YUAN; QIAO, 2022), entre outras.

As GANs são compostas por duas DNNs, chamadas gerador e discriminador. Essas duas redes possuem objetivos opostos e distintos, por isso o termo "adversárias". A rede discriminadora é responsável por classificar seus dados de entrada. Isto é, ela tenta mapear a entrada para uma categoria que ela pertença. Por outro lado, a rede geradora preocupa-se em fazer o oposto, ao invés de predizer a classe de um dado de entrada, ela tenta prever o dado de uma classe. Com isso, ela tenta criar dados os mais semelhantes possíveis aos dados originais com o intuito de confundir o discriminador.

A Figura 7 ilustra o funcionamento de uma GAN. Um ruído aleatório **z** é fornecido como entrada para o gerador **G**, onde é gerado o dado falso. Em seguida, o discriminador **D** recebe os dados reais e falsos como entrada. Seu objetivo é rotular se eles pertencem ou não ao conjunto de treinamento. Assim, **D** avalia a autenticidade das instâncias criadas por **G**. Nesse exemplo, o que estão sendo gerados são números escritos à mão, assim como os presentes na base de dados MNIST (DENG, 2012). Enquanto **D** tenta distinguir as entradas e classificá-las corretamente, **G** gera novas imagens falsas, visando que elas sejam consideradas verdadeiras.

Figura 7 – Fluxograma de uma GAN.



Fonte: Elaborada pelo autor (2022).

O treinamento adversário consiste em colocar as redes **G** e **D** uma contra a outra. Para isso, elas são treinadas simultaneamente e tentam otimizar uma função de perda que represente os seus objetivos. Com isso, elas auxiliam-se durante o treinamento, onde **D** incita **G** a gerar imagens mais fiéis ao conjunto de treinamento para confundir **D**.

Desse modo, se $\mathbf{D}$ muda o seu comportamento para derrotar $\mathbf{G}$, $\mathbf{G}$ também se adapta para superar $\mathbf{D}$. Inicialmente, a entrada do discriminador pode pertencer ao conjunto de treinamento ou ser um dado gerado pelo gerador. Se o dado for real, ele pertence à distribuição de probabilidade do conjunto de treinamento p_{data}, ou seja, $x \sim p_{data}$. Caso o dado seja falso, ele pertence à distribuição de probabilidade do gerador p_g, logo $x \sim p_g$.

Diferentes funções objetivos podem ser usadas em GANs, mas a proposta original de Goodfellow et al. (2014) faz uso da Minimax. A função é dada por:

$$\min_{\mathbf{G}} \max_{\mathbf{D}} \mathbb{E}_{\mathbf{x} \sim p_{data}(\mathbf{x})}[\log \mathbf{D}(\mathbf{x})] + \mathbb{E}_{\mathbf{z} \sim p_z(\mathbf{z})}[\log(1 - \mathbf{D}(\mathbf{G}(\mathbf{z}))], \tag{2.11}$$

onde $\mathbf{D}(\mathbf{x})$ retorna a probabilidade de $\mathbf{x}$ pertencer a p_{data} ao invés de p_g. A rede $\mathbf{D}$ maximiza a log-probabilidade de classificar as entradas $\mathbf{x}$ e $\mathbf{G}(\mathbf{z})$, e a $\mathbf{G}$ minimiza a log-probabilidade de $\mathbf{D}$ classificar corretamente a entrada $\mathbf{G}(\mathbf{z})$.

Uma proposta de otimização foi feita por Goodfellow et al. (2014) para evitar que $\mathbf{D}$ sofra sobre-ajuste. Ela consiste em otimizar $\mathbf{D}$ a cada k passos do treinamento, mantendo a otimização de $\mathbf{G}$ para cada passo. Além disso, a Equação (2.11) pode ser modificada para que auxilie $\mathbf{G}$ durante o início do treinamento, visto que $\mathbf{D}$ pode classificar os dados falsos com uma maior confiança por eles serem evidentemente distintos do conjunto de treinamento. Para isso, nas primeiras épocas, ao invés de treinar $\mathbf{G}$ para minimizar $\log(1 - \mathbf{D}(\mathbf{G}(\mathbf{z}))$, pode-se treiná-la para maximizar $\log(\mathbf{D}(\mathbf{G}(\mathbf{z}))$. Mesmo mantendo os objetivos adversários entre as redes, esta alteração provê gradientes mais competitivos para $\mathbf{G}$ durante o início do treinamento. A nova função objetivo passa a ser dada por:

$$\max_{\mathbf{D}}[\mathbb{E}_{\mathbf{x} \sim p_{data}(\mathbf{x})}[\log \mathbf{D}(\mathbf{x})] + \mathbb{E}_{\mathbf{z} \sim p_z(\mathbf{z})}[\log(1 - \mathbf{D}(\mathbf{G}(\mathbf{z}))]] + \max_{\mathbf{G}} \mathbb{E}_{\mathbf{z} \sim p_z(\mathbf{z})}[\log \mathbf{D}(\mathbf{G}(\mathbf{z}))]. \tag{2.12}$$

3 MÉTODOS PROPOSTOS

Neste capítulo, são apresentados os métodos propostos nesta dissertação. Na Seção 3.1 é descrito o método proposto, chamado *3D-based Seismic Data Compression Using Generative Adversarial Network* (3DSC-GAN), além de sua arquitetura e funções de perda. Em seguida, na Seção 3.2, é explicada a proposta da nova função de perda para dados multidimensionais.

3.1 *3D-BASED SEISMIC DATA COMPRESSION USING GENERATIVE ADVERSARIAL NETWORK* (3DSC-GAN)

Um dos principais objetivos deste trabalho é integrar a estrutura de uma GAN com o método 3DSC proposto por Schiavon (2020). A abordagem original é composta por três módulos principais: codificador, decodificador e módulo de probabilidade. Cada um desses módulos possui um objetivo para auxiliar na compressão e descompressão de um subvolume sísmico 3D. O codificador mapeia o subvolume de entrada em uma representação latente representada por um espaço com dimensão reduzida. Em seguida, o decodificador remapeia essa representação latente no espaço original da entrada. E por fim, o módulo de probabilidade é responsável por estimar a quantidade de bits necessária para representar o subvolume comprimido.

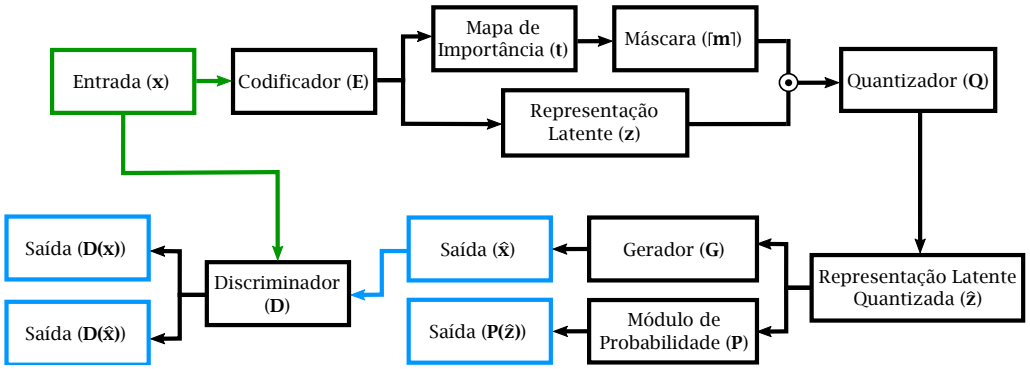
A integração do método 3DSC com uma GAN foi inspirada no trabalho de Agustsson et al. (2019). Nesse trabalho, os autores propõem um compressor de imagens naturais para taxas de bits extremas utilizando uma GAN. Para isso, eles utilizam um autocodificador para comprimir, onde sua arquitetura foi proposta por Wang et al. (2018), que por sua vez foi baseada na arquitetura de Johnson, Alahi e Fei-Fei (2016). A inserção da estrutura GAN nesse autocodificador foi feita ao reinterpretar o módulo decodificador para se tornar um módulo gerador. Além disso, eles adicionaram um módulo ao final da rede para diferenciar a entrada original e sua reconstrução, chamado discriminador.

Assim como no trabalho de Agustsson et al. (2019), a integração do 3DSC com uma GAN é realizada adaptando o módulo decodificador como um gerador e inserindo o novo módulo discriminador. O método resultante da inserção de uma GAN no 3DSC é chamado 3DSC-GAN. Essa nova abordagem consiste em quatro módulos, que são: codificador, módulo de probabilidade, gerador e discriminador. Os módulos codificador e de probabilidade possuem uma descrição similar às relatadas anteriormente. O gerador tenta gerar a reconstrução a partir da representação latente e o discriminador tenta classificar se a sua entrada pertence ao dado original ou se é uma reconstrução gerada pelo gerador.

A Figura 8 ilustra uma visão geral do método 3DSC-GAN. Assim como o 3DSC, um dado sísmico é entrada para ser comprimido pelo módulo codificador, onde é gerada a

representação latente e o mapa de importância. O mapa de importância é expandido para uma máscara que é combinada com a representação latente para torná-la discreta. Após isso, a quantização é aplicada para discretizar os elementos dessa combinação, resultando na representação latente quantizada. Então, o dado discretizado é passado para o gerador, o qual tenta reconstruir o dado de entrada, e também para o módulo de probabilidade, onde é estimada a taxa de bits necessária para codificar a representação latente quantizada. Ao final, o volume de entrada e a sua reconstrução são fornecidos como entrada para o discriminador. Esse módulo rotula as entradas como sendo do conjunto de treinamento, ou como sendo gerada pelo módulo gerador. O método 3DSC-GAN foi publicado no IEEE *Geoscience and Remote Sensing Letters* (RIBEIRO et al., 2022) cujo código está disponível em <https://github.com/GCG-UFJF/3DSC-GAN>.

Figura 8 – Fluxograma do método proposto.



Fonte: Elaborada pelo autor (2022).

Similar ao proposto por Schiavon (2020), a entrada da rede é um subvolume sísmico $\mathbf{x} \in \mathbb{R}^{h \times w \times d}$, com h e w sendo as dimensões espaciais e d o número de fatias 2D que o compõe. O autocodificador é composto por um codificador $\mathbf{E} : \mathbb{R}^{h \times w \times d} \rightarrow \mathbb{R}^{h/8 \times w/8 \times C}$ que mapeia a entrada para uma representação latente $\mathbf{z}$. O número 8 indica o fator de subamostragem da rede para reduzir as dimensões espaciais de $\mathbf{x}$. O parâmetro C representa o número de mapas de características obtidos pela última camada convolucional de $\mathbf{E}$. O quantizador $\mathbf{Q} : \mathbb{R}^{h/8 \times w/8 \times C} \rightarrow \mathcal{C}^{h/8 \times w/8 \times C}$ é responsável por discretizar os valores de $\mathbf{z}$, resultando em uma representação latente quantizada $\hat{\mathbf{z}} = \mathbf{Q}(\mathbf{z}) \in \mathcal{C}^{h/8 \times w/8 \times C}$ que é mais adequada para compressão. O gerador $\mathbf{G} : \mathcal{C}^{h/8 \times w/8 \times C} \rightarrow \mathbb{R}^{h \times w \times d}$ gera o dado reconstruído $\hat{\mathbf{x}} = \mathbf{G}(\hat{\mathbf{z}}) \in \mathbb{R}^{h \times w \times d}$.

A quantização é a mesma aplicada em Schiavon (2020), originalmente proposta por Mentzer et al. (2018), na qual a discretização é realizada depois que algumas entradas de $\mathbf{z}$ são zeradas utilizando uma máscara. Essa discretização permite que a alocação de bits seja espacialmente adaptativa. Para isso, um mapa de importância $\mathbf{t}$ é incorporado a última camada do codificador com dimensões espaciais $h/8 \times w/8$. Esse mapa é responsável por aprender as regiões de $\mathbf{z}$ que fornecem a informação mais relevante, que necessita de mais

bits para ser representada. Com isso, $\mathbf{t}$ é expandido em uma máscara $\lceil \mathbf{m} \rceil$ com as mesmas dimensões da representação latente. Então, é realizada a multiplicação ponto a ponto que preserva os coeficientes de maior entropia e zera os coeficientes com menos informação. Em seguida, dado um conjunto finito de centroides $\mathcal{C} = \{c_1, \cdots, c_L\} \subset \mathbb{R}$ aprendidos pelo autocodificador, a discretização atribui cada valor de $\mathbf{z}$ ao centroide mais próximo, o que resulta na representação latente quantizada $\hat{\mathbf{z}}$. Como a quantização não é diferenciável, é utilizada uma aproximação suave na retropropagação (Equação (2.4)).

O total de bits necessários para representar $\hat{\mathbf{z}}$ é calculado a partir da distribuição conjunta de probabilidade p da representação latente, definida pela Equação (2.5). O objetivo do módulo de probabilidade $\mathbf{P}$ é estimar a distribuição condicional $p(\hat{\mathbf{z}})$. Para isso, $\mathbf{P}$ é treinado para prever as probabilidades de atribuir $\hat{z}_i$ a cada centroide $c_j \in \mathcal{C}$, considerando a sua similaridade. A CNN 3D $\mathbf{P}$ é treinada utilizando a perda descrita na Equação (2.7). Além disso, o autocodificador utiliza o módulo de probabilidade para estimar a taxa de bits $\mathbf{r}$ da Equação (1.4). As probabilidades de $P(\hat{\mathbf{z}})$ são ponderadas com a máscara $\lceil \mathbf{m} \rceil$ permitindo controlar o custo de codificação. A função de perda do autocodificador é descrita na Equação (2.10).

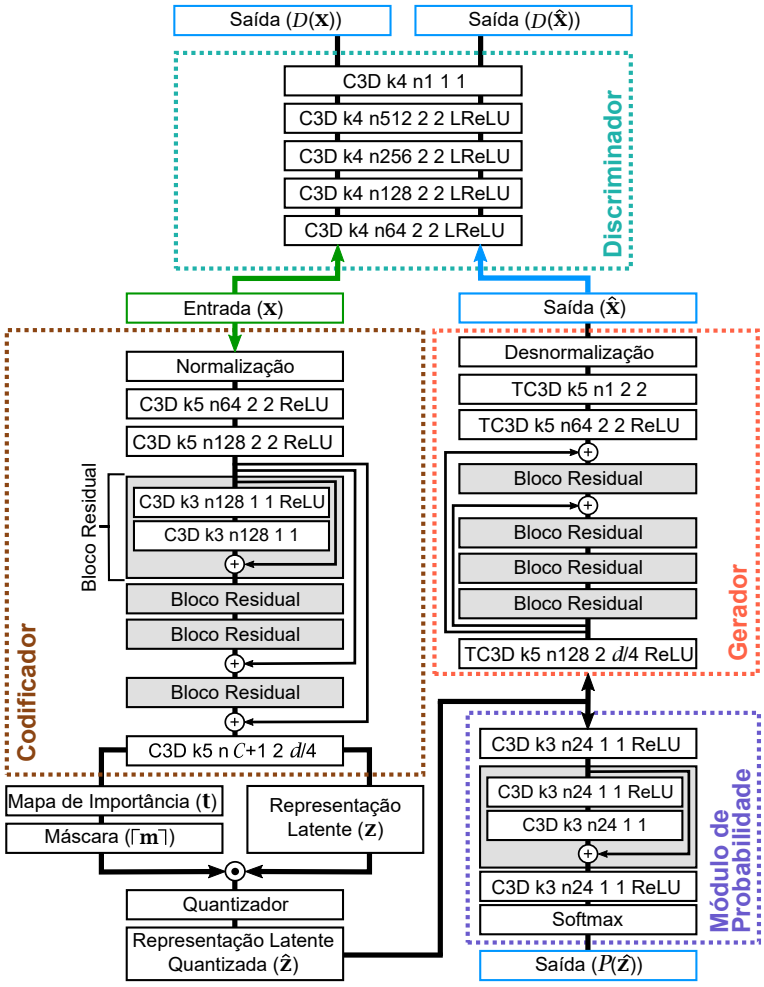
A estrutura GAN é composta pelos módulos gerador $\mathbf{G}$ e discriminador $\mathbf{D} : \mathbb{R}^{h \times w \times d} \to \mathbb{R}$. Assim como em Agustsson et al. (2019), é utilizado o discriminador proposto por Wang et al. (2018), mas adaptado para aplicar convoluções 3D. Para aplicar o treinamento adversário, é necessário colocar as redes uma contra a outra usando objetivos diferentes e opostos. O discriminador encontra a diferença entre o dado real e sua reconstrução. Por outro lado, o gerador minimiza a distorção produzida ao recuperar a entrada a partir da representação latente. O seu objetivo é fornecer uma reconstrução de alta fidelidade para enganar o discriminador. As funções objetivo utilizadas são uma adaptação das perdas por mínimos quadrados GAN, definida por:

$$\min_{\mathbf{G}} \mathbb{E}[(\mathbf{D}(\mathbf{G}(\hat{\mathbf{z}})) - 1)^2],$$
$$\min_{\mathbf{D}} \mathbb{E}[(\mathbf{D}(\mathbf{x}) - 1)^2] + \mathbb{E}[(\mathbf{D}(\mathbf{G}(\hat{\mathbf{z}})))^2],$$

onde $\mathbf{G}$ tenta minimizar a acurácia de $\mathbf{D}$, e $\mathbf{D}$ tenta classificar as entradas corretamente. Vale ressaltar que ambas as redes são treinadas simultaneamente e que a entrada do discriminador é alternada para prevenir sobre-ajuste, visto que ele pode sempre classificar a primeira entrada como sendo o dado original. Essa alternância é feita para aperfeiçoar a estimativa de redundância volumétrica 3D do autocodificador do 3DSC. A arquitetura do método 3DSC-GAN é ilustrada na Figura 9.

Assim como no método 3DSC, o dado de entrada $\mathbf{x} \in \mathbb{R}^{h \times w \times d}$ é fornecido como entrada do módulo codificador, onde ele é normalizado utilizando a média e o desvio-padrão, denotado pelo bloco "Normalização". Após isso, "C3D k5 n64 2 2 ReLU" indica uma convolução 3D com 64 filtros de tamanho 5, salto 2 em todas as dimensões e a função de ativação é a ReLU. Em seguida, quatro blocos residuais são utilizados com uma

Figura 9 – Arquitetura do método 3DSC-GAN.



Fonte: Elaborada pelo autor (2022).

combinação dos resultados antes e após o último bloco. A estrutura de camadas de cada bloco é apresentada no primeiro bloco residual do módulo codificador. A última camada do codificador aplica uma convolução 3D com $C + 1$ filtros e salto $d/4$ na dimensão *time-depth*, resultando em uma representação latente $\mathbf{z}$ e um mapa de importância $\mathbf{t}$, onde C representa o número de mapas de características de $\mathbf{z}$. Logo depois, $\mathbf{t}$ é expandido, gerando a máscara $\lceil \mathbf{m} \rceil$. A combinação da máscara com $\mathbf{z}$ é quantizada no módulo quantizador $\mathbf{Q}$, gerando a representação latente quantizada $\hat{\mathbf{z}}$, que é fornecida como entrada para o gerador e o módulo de probabilidade. O gerador espelha o codificador para reconstruir $\hat{\mathbf{z}}$ com convoluções transpostas, denotadas por "TC3D". O bloco "Desnormalização" aplica a operação inversa do "Normalização". O módulo de probabilidade estima a quantidade total de bits necessária para representar $\hat{\mathbf{z}}$. O discriminador recebe $\mathbf{x}$ e a reconstrução $\hat{\mathbf{x}}$ como entrada. Esse módulo utiliza a função de ativação *leaky* ReLU, denotada por "LReLU". Todas as camadas convolucionais do codificador, gerador e módulo de probabilidade são normalizadas utilizando a normalização em lote (*batch*). As camadas do discriminador,

exceto a primeira, são normalizadas com normalização de instância, na qual cada amostra do lote é normalizada com seus respectivos valores de média e variância.

3.2 FUNÇÃO DE PERDA MULTIDIMENSIONAL COMPOSTA

Devido à necessidade observada em trabalhos recentes (NAVARRO et al., 2019; SCHIAVON et al., 2019; SCHIAVON, 2020; SCHIAVON et al., 2020), o aprendizado para compressão de dados sísmicos tem a carência de funções de perda mais adequadas para o problema. A métrica PSNR é aplicada como função de perda nos trabalhos de Schiavon et al. (2019) e Schiavon et al. (2020), onde é realizado uma diferença ponto a ponto entre o volume sísmico original e sua reconstrução. Essa métrica é baseada na função MSE (Equação (2.9)) a qual necessita da paridade dos dados para obter uma baixa distorção. Além disso, a métrica PSNR desconsidera informação topológica e geométrica dos dados multidimensionais, assim como os dados sísmicos pós-pilha 3D. Assim, ao utilizar esta métrica, todos os eixos são considerados da mesma forma por ser uma função baseada em pixel.

Dados sísmicos possuem diferentes espectros da informação alinhada a cada eixo, podendo ser analisados de modos distintos. No trabalho de Schiavon (2020), a autora aponta que a utilização do eixo z (*time-depth*) durante o treinamento resulta em uma divergência do aprendizado e uma baixa qualidade de reconstrução. Portanto, esse eixo pode ter um peso menor durante o treinamento, em relação aos demais (eixos x e y). Teoricamente, isso pode beneficiar a rede a aprender os padrões mais importantes durante a reconstrução do dado, e com isso, entregar um PSNR maior mesmo sem utilizá-lo como função objetivo.

Neste trabalho, é proposta uma nova função de perda para dados sísmicos em que os eixos são reduzidos com até três funções diferentes. Mais especificamente, a função de perda 3D proposta neste trabalho reduz a dimensão de cada eixo através de uma função específica para cada um deles. A Figura 10 ilustra esse processo.

Considere um volume $\mathbf{x} \in \mathbb{R}^{h \times w \times d}$ e sua reconstrução $\hat{\mathbf{x}} \in \mathbb{R}^{h \times w \times d}$. A primeira etapa é obter o volume da diferença entre os dados, resultando em um campo escalar $\mathbf{E}(\mathbf{x}, \hat{\mathbf{x}}) \in \mathbb{R}^{h \times w \times d}$, dado por:

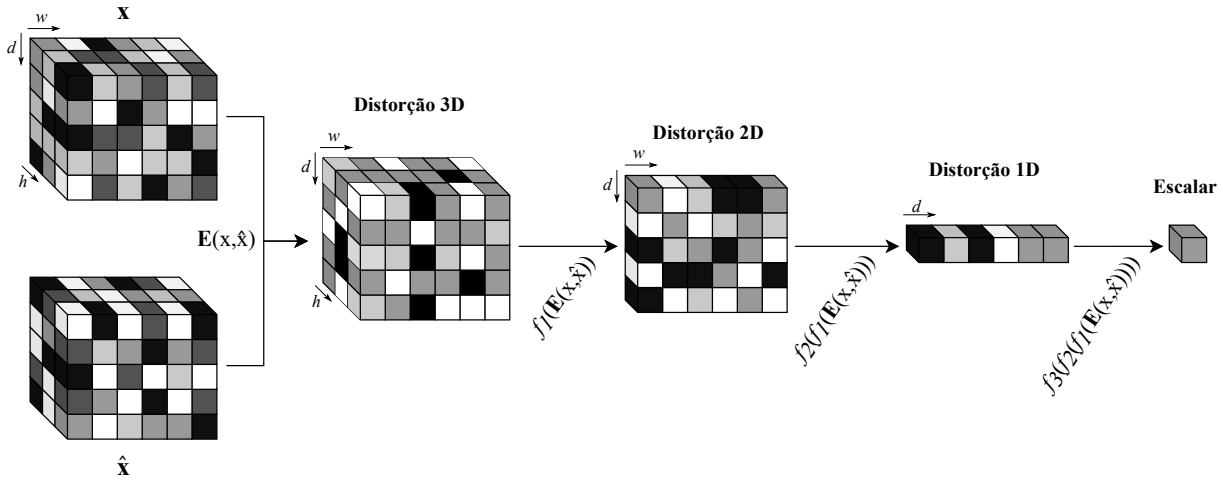
$$\mathbf{E}(\mathbf{x}, \hat{\mathbf{x}}) = \mathbf{x} - \hat{\mathbf{x}}. \tag{3.1}$$

Agora considere $f_1 : \mathbb{R}^{h \times w \times d} \to \mathbb{R}^{w \times d}$, $f_2 : \mathbb{R}^{w \times d} \to \mathbb{R}^d$, e $f_3 : \mathbb{R}^d \to \mathbb{R}$, três funções cujo objetivo é agregar o erro de vetores 1D de um eixo específico em um escalar. Neste trabalho, as funções de agregação do erro unidimensional são definidas da seguinte forma:

$$f_1(\mathbf{E}(\mathbf{x}, \hat{\mathbf{x}})) = \sum_{1}^{h} M_1(\mathbf{E}(\mathbf{x}, \hat{\mathbf{x}})), \tag{3.2}$$

$$f_2(f_1(\mathbf{E}(\mathbf{x}, \hat{\mathbf{x}}))) = \sum_{1}^{w} M_2(f_1(\mathbf{E}(\mathbf{x}, \hat{\mathbf{x}}))), \tag{3.3}$$

Figura 10 – Reduções aplicadas para o cálculo da distorção.



Fonte: Elaborada pelo autor (2022).

e

$$f_3(f_2(f_1(\mathbf{E}(\mathbf{x},\hat{\mathbf{x}})))) = \sum_{1}^{d} M_3(f_2(f_1(\mathbf{E}(\mathbf{x},\hat{\mathbf{x}})))), \tag{3.4}$$

onde M_1, M_2 e M_3 são funções de distorção. A função f_1 calcula o erro relacionado com cada vetor 1D obtido na direção h do volume da diferença original $\mathbf{E}(\mathbf{x},\hat{\mathbf{x}})$, cujos valores são dados pela função M_1, gerando uma saída 2D. Já a função f_2 calcula o erro agregado dos vetores 1D adquiridos no eixo w através da função M_2, gerando uma saída 1D. Finalmente, a função f_3 utiliza a síntese dos erros de $\mathbf{E}(\mathbf{x},\hat{\mathbf{x}})$ nas direções h e w, e calcula o erro correspondente desse vetor no eixo d com a função M_3, resultando em um escalar. O primeiro eixo a ser reduzido h deve ser aquele que melhor caracteriza as estruturas de interesse do dado sísmico, podendo ser tanto o da direção *inline* como a *crossline*. Já o eixo d deve ser aquele em que as variações não caracterizam as estruturas subjacentes de interesse no dado sísmico (*time-depth*).

Uma vez definida a função de perda multidimensional composta, note que ela pode ser incorporada ao método 3DSC-GAN (Seção 3.1) ao substituir a métrica PSNR na função de distorção $\mathbf{d}(\mathbf{x},\hat{\mathbf{x}})$. A distorção $\mathbf{d}(\mathbf{x},\hat{\mathbf{x}})$ está presente na função de perda do módulo de probabilidade (Equação (2.7)) e do autocodificador (Equação (2.10)). Portanto, a nova função de distorção é definida por:

$$\mathbf{d_{composta}}(\mathbf{x},\hat{\mathbf{x}}) = (f_3 \circ f_2 \circ f_1)(\mathbf{x},\hat{\mathbf{x}}) = f_3(f_2(f_1(\mathbf{E}(\mathbf{x},\hat{\mathbf{x}})))). \tag{3.5}$$

O desafio é definir as funções M_1, M_2, e M_3, de forma que as características dos eixos h, w, e d sejam exploradas no cálculo da perda. Neste trabalho, as funções *Huber* (HUBER, 1992), *L1*, *L2*, *LogCosh* e *Média* são selecionadas para fins de comparação. Elas são

definidas pelas seguintes funções:

$$Huber(v) = \begin{cases} \sum_{i=1}^{N} \frac{1}{2} v_i^2 & \text{se} \quad |v_i| \leq \delta \\ \sum_{i=1}^{N} \delta(|v_i| - \frac{1}{2}\delta) & \text{se} \quad |v_i| > \delta \end{cases}, \tag{3.6}$$

$$L1(v) = \sum_{i=1}^{N} |v_i|, \tag{3.7}$$

$$L2(v) = \sum_{i=1}^{N} v_i^2, \tag{3.8}$$

$$LogCosh(v) = \frac{1}{N} \sum_{i=1}^{N} \log(\cosh(v_i)) \approx \frac{1}{N}(v_i + \log(\exp(-2v_i) + 1) - \log(2)), \tag{3.9}$$

e

$$Média(v) = \frac{1}{N} \sum_{i=1}^{N} v_i, \tag{3.10}$$

onde v é um vetor unidimensional de tamanho N. O objetivo é encontrar experimentalmente qual a combinação de funções provê a melhor eficácia e aprimora a capacidade de reconstrução do método 3DSC-GAN.

4 EXPERIMENTOS E RESULTADOS

Neste capítulo são apresentados os experimentos realizados para avaliar os métodos propostos. Na Seção 4.1 é apresentada uma descrição geral das bases de dados utilizadas durante os experimentos, com o protocolo de treinamento e teste. A Seção 4.2 contém os experimentos realizados para avaliação de eficácia do método 3DSC-GAN. Na Seção 4.3, são descritos os experimentos para estudar o comportamento da função de perda composta para dados multidimensionais, aplicada no problema de compressão de dados sísmicos.

4.1 BASE DE DADOS

Os métodos foram treinados e testados com dados sísmicos pós-pilha 3D reais, disponibilizados pela *Society of Exploration Geophysicists* (SEG) *Open Data Repository* (SEG, 2022). Os dados escolhidos foram: Kahu3D, Kerry3D, Netherlands F3-Block, Opunake3D, Parihaka3D, Penobscot3D, Poseidon3D e Waihapa3D. A Tabela 1 mostra os detalhes de cada um deles. Note que a dificuldade para compressão é definida conforme a presença de altas e baixas frequências no dado. Dados cujos espectros têm a presença de mais altas frequências são mais desafiadores para compressão. Por outro lado, os volumes sísmicos com maior predominância de baixas frequências permitem que o dado seja comprimido com maior facilidade.

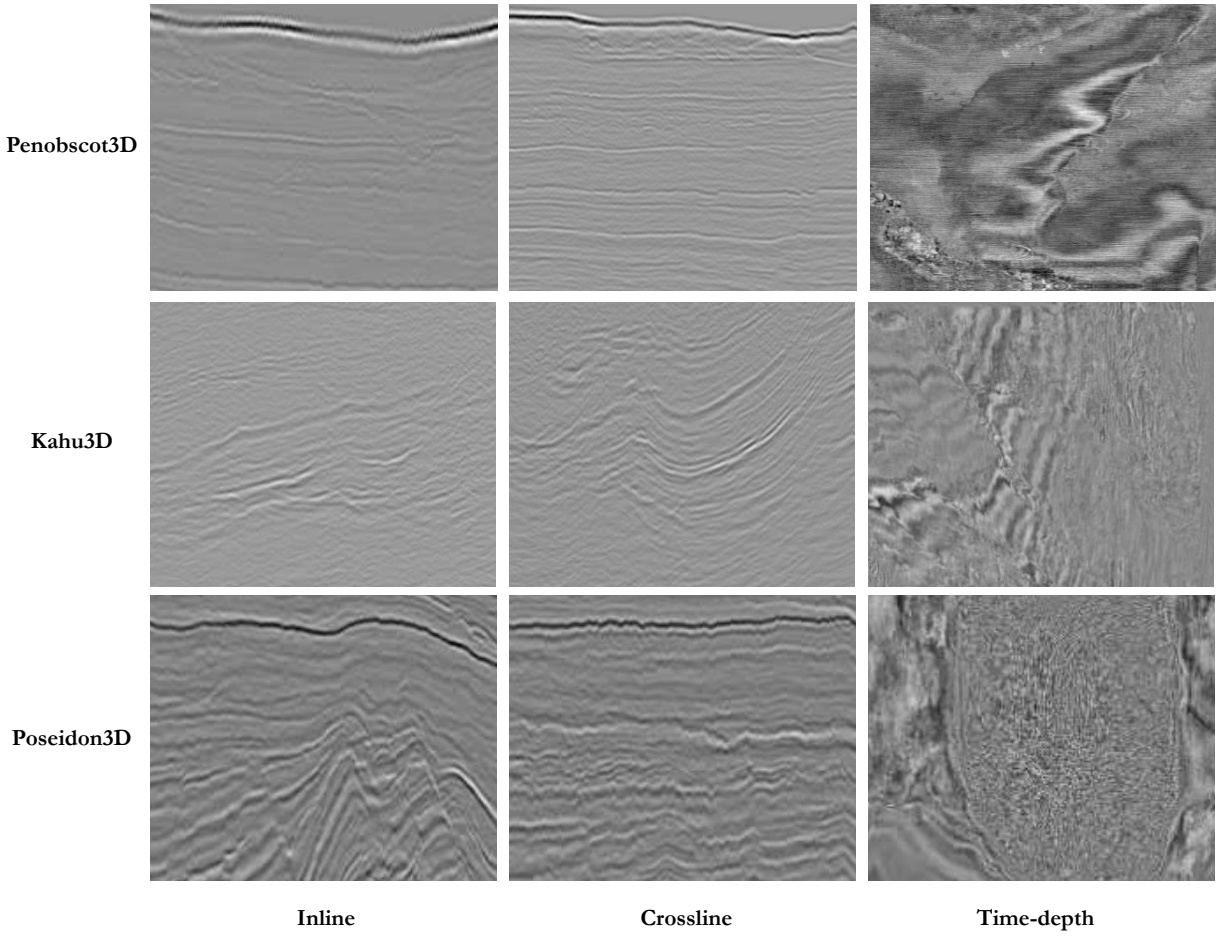
Tabela 1 – Características da base de dados.

Volume	Tamanho (GB)	Dimensões (voxels)	Dificuldade para Compressão
Kahu3D	5,74	$(584 \times 1695 \times 1498)$	Média
Kerry3D	0,77	$(226 \times 711 \times 1218)$	Alta
Netherlands F3-Block	1,16	$(631 \times 951 \times 463)$	Média
Opunake3D	2,50	$(501 \times 988 \times 1301)$	Fácil
Parihaka3D	3,60	$(920 \times 1124 \times 874)$	Fácil
Penobscot3D	0,59	$(401 \times 301 \times 1251)$	Fácil
Poseidon3D	1,33	$(301 \times 967 \times 1176)$	Alta
Waihapa3D	0,28	$(201 \times 291 \times 1238)$	Média

Fonte: Elaborada pelo autor (2022).

Os dados sísmicos pós-pilha 3D são representados por pontos flutuantes de 32 bits em uma representação matricial $\mathbf{v} \in \mathbb{R}^{H \times W \times D}$, onde H indica a direção *inline*, composta pelos eixos (y, z), W refere-se à direção *crossline*, constituída pelos eixos (x, z), e D é relacionado à direção *time-depth*, composta pelos eixos (x, y). Na Figura 11 são apresentados exemplos de fatias obtidas nas direções *inline*, *crossline*, e *time-depth* para os volumes com dificuldades distintas para compressão, Penobscot3D (fácil), Kahu3D (média), e Poseidon3D (difícil). Note que a direção *time-depth* é mais ruidosa comparada com as demais para todos os volumes.

Figura 11 – Exemplos de fatias extraídas dos volumes Penobscot3D, Kahu3D, e Poseidon3D nas direções *inline*, *crossline*, e *time-depth*

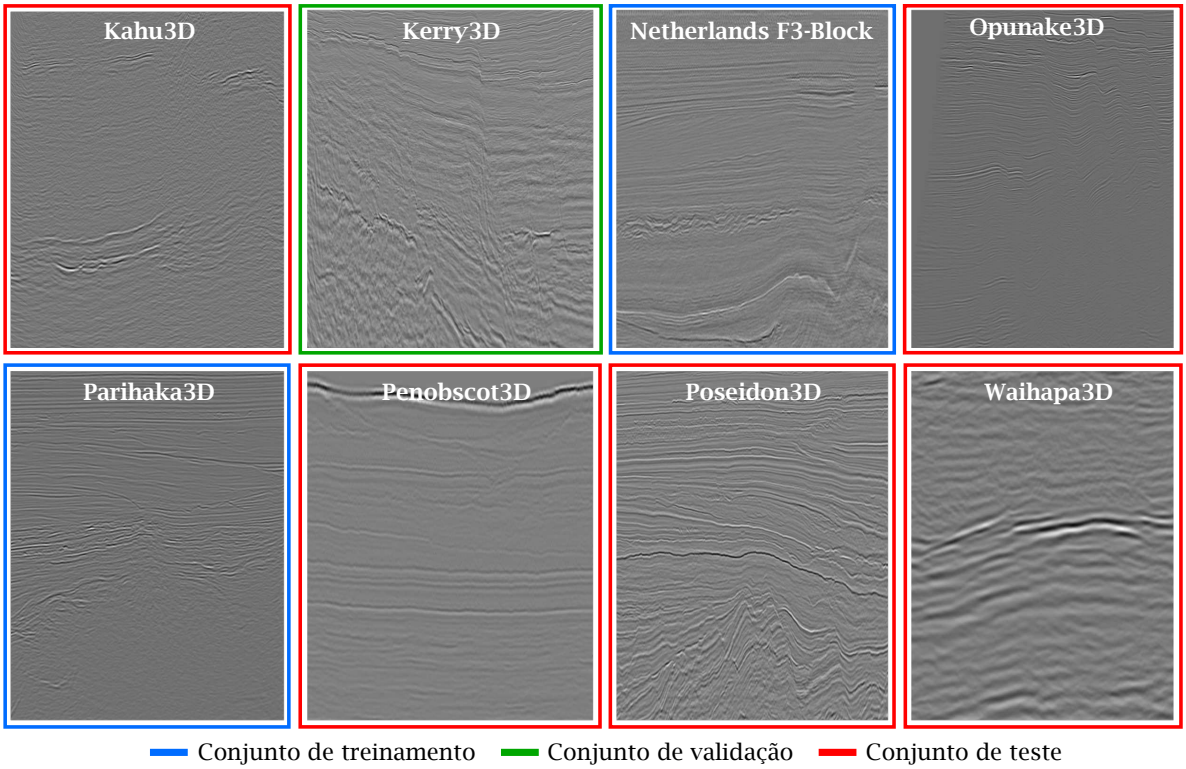


Fonte: Elaborada pelo autor (2022).

O conjunto de treinamento é composto pelos volumes Netherlands F3-Block e Parihaka3D por eles possuírem uma complexidade para compressão média e baixa, respectivamente. O volume Kerry3D com complexidade alta pertence ao conjunto de validação. O conjunto de teste consiste dos volumes Kahu3D, Opunake3D, Penobscot3D, Poseidon3D e Waihapa3D. Os conjuntos são divididos dessa forma devido à dificuldade de treinamento de uma rede que utiliza GAN. A adição de mais volumes ao conjunto de treinamento dificulta a convergência do método, tornando os parâmetros mais sensíveis. A Figura 12 mostra exemplos de fatias da direção *inline* de cada volume. Cada fatia está contornada segundo o conjunto que o seu respectivo volume pertence. Azul indica que os dados pertencem ao conjunto de treinamento, verde representa o de validação, e vermelho aponta o conjunto de teste. O volume Poseidon3D é bastante ruidoso e com isso, mais complexo para comprimir. Todos os volumes são normalizados para o intervalo de $[0, 1]$ com seus respectivos valores de máximos e mínimos. Essa normalização permite que seja possível treinar e testar em diferentes volumes, uma vez que seus valores mínimos e máximos são

distintos.

Figura 12 – Exemplos de fatias extraídas na direção *inline* dos volumes presentes na base de dados.

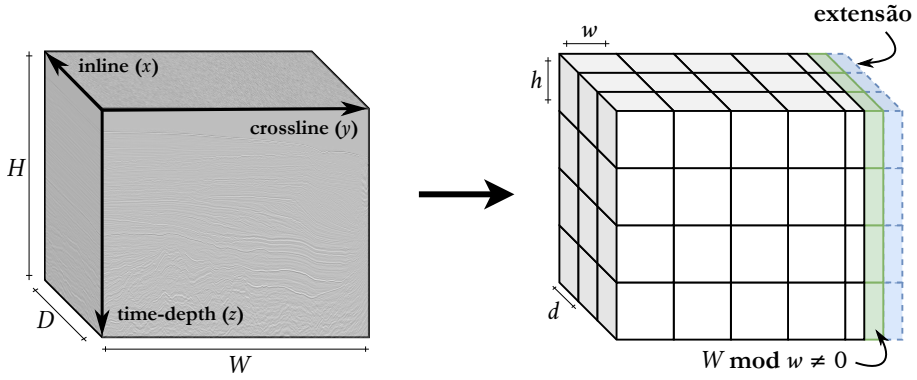


Fonte: Elaborada pelo autor (2022).

A extração dos subvolumes aplicada nos volumes descritos na Tabela 1 é baseada nos procedimentos propostos por Schiavon (2020). Dado um volume $\mathbf{v} \in \mathbb{R}^{H \times W \times D}$, os subvolumes extraídos possuem as dimensões $h \times w \times d$ sendo utilizados como entrada do método durante o treinamento e validação. Esse processo pode considerar as direções *inline*, *crossline* e *time-depth*, que representam os eixos x, y e z, respectivamente. Por exemplo, um subvolume extraído na direção *inline* é composto por d fatias de tamanho $h \times w$ extraídos através do eixo x. De maneira similar, os subvolumes nas direções *crossline* e *time-depth* são extraídos dos eixos y e z, respectivamente. Caso alguma dimensão H, W ou D do volume não seja divisível pelo seu valor correspondente do subvolume, h, w ou d, os voxels restantes são sobrepostos utilizando os voxels simétricos obtidos no volume original. Em outras palavras, os voxels ausentes são preenchidos aplicando a extensão de borda simétrica. Com isso, as estruturas dos dados sísmicos são melhor preservados. A Figura 13 ilustra o procedimento de extração realizado e um exemplo do processo de sobreposição onde a dimensão W não é divisível por w. Os subvolumes originais são estendidos até a área em azul, aplicando-se a sobreposição usando a área em verde. O mesmo ocorre com as direções *inline* e *time-depth*. Vale ressaltar que a extração dos

subvolumes é feita utilizando a varredura *scan*, e cada subvolume extraído não possui sobreposição com os demais.

Figura 13 – Extração de subvolumes aplicada nos dados sísmicos.



Fonte: Adaptação de Schiavon (2020).

Os subvolumes utilizados durante a etapa de teste são distintos dos extraídos para o treinamento e validação, devido às suas dimensões. As dimensões dos subvolumes do conjunto de teste são $H \times W \times d$. Visto que a extração dos subvolumes de teste é feita em conjunto com os de treinamento e validação, as dimensões que não são divisíveis pelo seu valor correspondente do subvolume de treinamento/validação, h, w ou d, são estendidas simetricamente na borda. Dessa forma, para avaliar a reconstrução da rede é necessário remover as extensões aplicadas e, em seguida, o volume reconstruído é desnormalizado para então medir a precisão da reconstrução.

O eixo principal selecionado para extração dos dados é importante em 3D, devido às variações típicas dos dados sísmicos em cada eixo (Figura 11). Neste trabalho, assim como no trabalho de Schiavon (2020), apenas os subvolumes extraídos nas direções *inline* e *crossline* são utilizadas para treinamento e validação. Porém, apenas os subvolumes da direção *inline* foram considerados durante a etapa de teste, visto que a rede atinge resultados similares utilizando os subvolumes da direção *crossline* (SCHIAVON, 2020). Os subvolumes da direção *time-depth* não são utilizados por possuírem padrões que dificultam a convergência do método.

4.2 EXPERIMENTOS PARA O MÉTODO 3DSC-GAN COM FUNÇÃO DE DISTORÇÃO PSNR

4.2.1 Configuração do treinamento

O método 3DSC-GAN foi implementado na plataforma TensorFlow. Todos os modelos foram executados utilizando uma GPU NVIDIA Quadro GV100 de 32 GB e um Core i7 870 com 16 GB de RAM. Eles foram treinados do zero com o otimizador ADAM

por 60 épocas. Para cada experimento, o único modelo salvo foi o com a menor perda durante a etapa de validação.

O autocodificador (codificador e gerador) e o módulo de probabilidade possuem uma taxa de aprendizado inicial de $8 \cdot 10^{-5}$ e etapa de decaimento de 0,05 a cada 10 épocas. O módulo discriminador utiliza uma taxa de aprendizado inicial de $2 \cdot 10^{-4}$, decaimento exponencial de 0,5 e a *leaky* ReLU com inclinação de 0,2.

O tamanho da entrada do modelo é de $160 \times 160 \times d$ com um tamanho do lote B específico para cada experimento. A representação latente de tamanho $C = 64$ foi treinada com $L = 50$ centroides (SCHIAVON, 2020). O valor de β foi fixado em 100 para controlar o balanceamento entre a distorção e a taxa de bits.

Além disso, assim como sugerido por Mentzer et al. (2018), o termo taxa de bits $\mathbf{r}$ da Equação (2.10) foi substituído por $\max(\beta\mathbf{r}(\hat{\mathbf{z}}), r_t)$ para forçar a taxa de bits alvo r_t. É importante ressaltar que a função PSNR foi utilizada como função de distorção $\mathbf{d}(\mathbf{x}, \hat{\mathbf{x}})$ para avaliar a distorção entre as entradas em decibéis.

4.2.2 Resultados

O tamanho do lote influencia o treinamento adversário diretamente durante as primeiras épocas. Inicialmente, um tamanho de lote grande favorece o discriminador, pois uma grande quantidade de amostras do conjunto de treinamento é fornecida de uma só vez. Dessa forma, o discriminador pode classificar seus subvolumes de entrada corretamente durante as primeiras épocas, impactando negativamente no aprendizado do gerador, e consequentemente, resultando em um desbalanceamento inicial entre os dois módulos. Um primeiro estudo é necessário para definir o tamanho de lote adequado para reduzir este problema antes de analisar e variar outros parâmetros. Para isso, o modelo foi treinado no conjunto de treinamento e avaliado em todos os volumes de teste apenas variando o tamanho do lote B. Empiricamente, a taxa de bits alvo r_t foi fixada em 1,0 bits por voxel (bpv) e o número de fatias utilizadas (tamanho da profundidade) como $d = 4$.

A Tabela 2 mostra os resultados de compressão para os tamanhos de lotes $B = \{1, 2, 4, 8, 10, 16, 32\}$ nos volumes do conjunto de teste. O melhor resultado em termos de PSNR para cada volume (coluna) estão destacados em cinza, onde células mais escuras indicam uma melhor compressão. Com esses resultados é possível notar que $B = 8$ atinge uma qualidade de reconstrução melhor que os demais para todos os volumes de teste. Além disso, para os tamanhos de lotes 10 e 16, temos que eles possuem a tendência de ficarem próximos ao melhor valor ($B = 8$), exceto para o volume Poseidon3D, onde o terceiro melhor resultado é obtido com $B = 2$. Esse volume possui uma alta dificuldade para compressão por ser bastante ruidoso. Nesse caso, um tamanho de lote menor pode favorecer sua reconstrução, e como pode ser observado na Tabela 2, o bpv resultante é o mais próximo da taxa de bits alvo r_t. Visto que o melhor resultado foi obtido com $B = 8$,

esse valor foi fixado para todos os experimentos subsequentes.

Tabela 2 – Variação do tamanho do lote para a taxa de bits alvo $r_t = 1{,}0$ bpv. Resultados reportados como PSNR/bpv. Células em cinza destacam os melhores resultados em termos de PSNR para cada volume (quanto mais escura, melhor).

Tamanho do lote (B)	Volumes				
	Kahu3D	Opunake3D	Penobscot3D	Poseidon3D	Waihapa3D
1	45,78 / 0,99	49,62 / 0,80	46,40 / 0,89	30,77 / 1,32	44,11 / 1,14
2	47,68 / 0,99	47,42 / 0,84	46,32 / 0,93	31,34 / 1,07	44,80 / 1,04
4	47,44 / 1,12	48,95 / 0,95	46,52 / 1,00	30,83 / 1,48	45,49 / 1,36
8	50,01 / 1,03	54,61 / 0,77	49,50 / 0,92	32,30 / 1,16	47,69 / 1,10
10	49,22 / 1,19	53,56 / 1,10	49,06 / 1,10	31,56 / 1,41	46,95 / 1,31
16	48,74 / 1,12	52,67 / 1,11	49,24 / 1,07	30,43 / 1,49	46,27 / 1,22
32	47,41 / 1,12	52,17 / 1,02	48,04 / 1,04	21,34 / 1,51	44,68 / 1,24

Fonte: Elaborada pelo autor (2022).

Um dos parâmetros mais importantes do 3DSC-GAN é o tamanho da profundidade d, o qual controla quantas fatias 2D de dimensões 160×160 são usados para compressão. A análise desse parâmetro é conduzida treinando a rede com $d = \{4, 8, 12, 16, 20, 24\}$. Outro parâmetro importante é a taxa de bits alvo r_t. O método é treinado com as taxas de bits alvo $r_t = \{0{,}05, 0{,}1, 0{,}25, 0{,}5, 0{,}75, 1{,}0\}$ bpv. A Tabela 3 mostra os resultados da compressão para o volume Waihapa3D do conjunto de teste. Por esse volume possuir uma complexidade média para compressão, ele é escolhido para essa primeira análise. O processo de treinamento/teste é aplicado 3 vezes para se determinar a eficácia média para compressão dos modelos. As melhores médias de PSNR para cada taxa de bits alvo (linha) são destacados em cinza, onde as células mais escuras significam melhor resultado.

Tabela 3 – Variação do tamanho da profundidade para o volume Waihapa3D. Resultados correspondentes a uma média de 3 testes, reportados como PSNR/bpv. Células em cinza destacam os melhores resultados em termos de PSNR para cada taxa de bits alvo (quanto mais escura, melhor).

Taxa de bits alvo	Tamanho da profundidade (d)					
	4 fatias	8 fatias	12 fatias	16 fatias	20 fatias	24 fatias
0,05	37,52 / 0,26	35,08 / 0,12	34,42 / 0,08	33,93 / 0,07	34,65 / 0,07	33,96 / 0,07
0,10	37,08 / 0,24	36,75 / 0,16	37,41 / 0,14	37,53 / 0,13	37,94 / 0,14	37,67 / 0,14
0,25	40,90 / 0,36	41,51 / 0,34	41,81 / 0,34	40,95 / 0,31	40,07 / 0,25	39,20 / 0,23
0,50	44,49 / 0,60	43,95 / 0,58	42,48 / 0,47	41,12 / 0,35	39,91 / 0,29	39,07 / 0,24
0,75	46,86 / 0,86	43,36 / 0,71	42,54 / 0,48	41,32 / 0,35	39,76 / 0,28	38,63 / 0,24
1,00	47,82 / 1,13	44,03 / 0,71	42,87 / 0,48	41,50 / 0,36	40,22 / 0,28	38,70 / 0,24

Fonte: Elaborada pelo autor (2022).

Os resultados sugerem uma melhor compressão com 4 fatias. Porém, a taxa de bits alvo é melhor atingida com 8 fatias com valores semelhantes de PSNR. Note que taxas de

bits alvo maiores ($r_t \geq 0{,}5$ bpv) são mais difíceis de serem alcançadas pela rede à medida que o tamanho da profundidade aumenta. Ou seja, essa arquitetura não melhora o PSNR quando mais informações são consideradas, convergindo para uma taxa de bits mais baixa. Experimentalmente, o termo β que controla o balanceamento entre a distorção e a taxa de bits (Equação (2.10)) não altera esse comportamento, e o mesmo ocorre ao se modificar o tamanho da representação latente. Isso sugere que a arquitetura deve ser adaptada para melhor alcançar uma taxa de bits alvo, preferencialmente melhorando a qualidade da reconstrução. Observe que 0,25 bpv é alcançado aproximadamente por todos os modelos e, portanto, essa taxa de bits alvo é interessante para avaliar o método.

A Tabela 4 mostra os resultados obtidos ao fixar $r_t = 0{,}25$ e variar $d = \{4, 8, 12, 16, 20, 24\}$ para todos os volumes do conjunto de teste. O processo de treinamento/teste é aplicado 3 vezes para capturar a eficácia média para compressão dos modelos. Curiosamente, a taxa de bits alvo é melhor alcançada ao se utilizar $d = 20$ para todos os volumes. Entretanto, em todos os casos, a melhor compressão em termos de PSNR é obtida com $d = 12$ fatias, sendo fixada para os próximos experimentos. O volume Poseidon3D é claramente o mais difícil de comprimir devido à presença de ruído. Sua melhor reconstrução obteve um PSNR de 29,39 dB com $d = 12$, enquanto os demais volumes possuem um PSNR superior a 39,19 dB. Com isso, esse volume é adequado para comparações qualitativas (Figura 17).

Tabela 4 – Variação do tamanho da profundidade para a taxa de bits alvo $r_t = 0{,}25$ bpv. Resultados correspondentes a uma média de 3 testes, reportados como PSNR/bpv. Células em cinza destacam os melhores resultados em termos de PSNR para cada taxa de bits alvo (quanto mais escura, melhor).

Volume	Tamanho da profundidade (d)					
	4 fatias	8 fatias	12 fatias	16 fatias	20 fatias	24 fatias
Kahu3D	42,99 / 0,31	42,92 / 0,29	44,33 / 0,29	42,72 / 0,26	41,48 / 0,25	40,73 / 0,23
Opunake3D	44,28 / 0,13	48,31 / 0,18	49,33 / 0,18	48,90 / 0,23	47,98 / 0,25	47,03 / 0,23
Penobscot3D	43,53 / 0,23	45,08 / 0,23	45,61 / 0,24	45,19 / 0,23	44,54 / 0,24	43,87 / 0,23
Poseidon3D	29,30 / 0,34	29,19 / 0,32	29,39 / 0,34	28,98 / 0,34	28,47 / 0,25	28,12 / 0,23
Waihapa3D	40,90 / 0,36	41,51 / 0,34	41,81 / 0,34	40,95 / 0,31	40,07 / 0,25	39,20 / 0,23

Fonte: Elaborada pelo autor (2022).

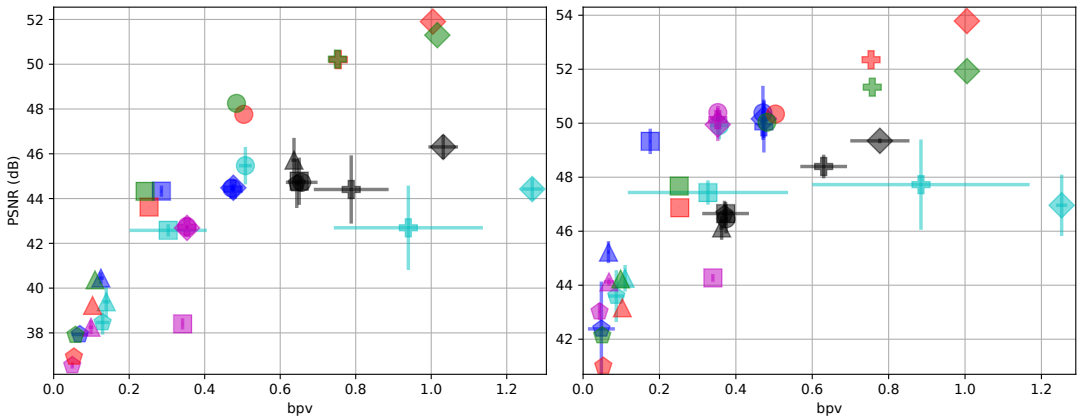
Uma comparação do 3DSC-GAN com outros métodos de compressão é apresentada na Figura 14 para todos os volumes do conjunto de teste. Outros três métodos baseados em aprendizado profundo são considerados: *2D-based Seismic Data Compression* (2DSC) (SCHIAVON et al., 2019), 3DSC (SCHIAVON, 2020), e MCSC (SCHIAVON et al., 2020). Eles foram treinados conforme os melhores parâmetros reportados em seus respectivos trabalhos.

Compressores de imagens e de uso geral são muito competitivos. Portanto, são

Figura 14 – Resultados da reconstrução para todos os volumes do conjunto de teste. Resultados correspondentes a uma média de 3 testes. Linhas verticais (PSNR) e horizontais (bpv) representam o desvio-padrão.

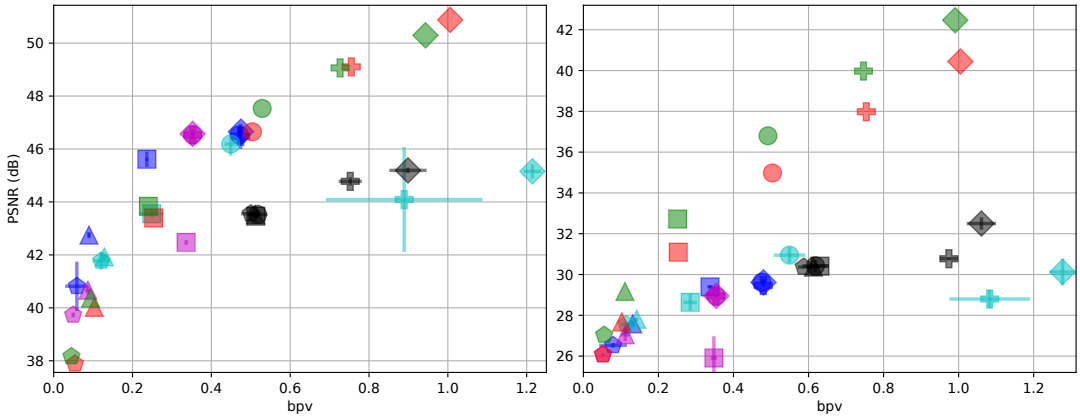
(a) Kahu3D

(b) Opunake3D

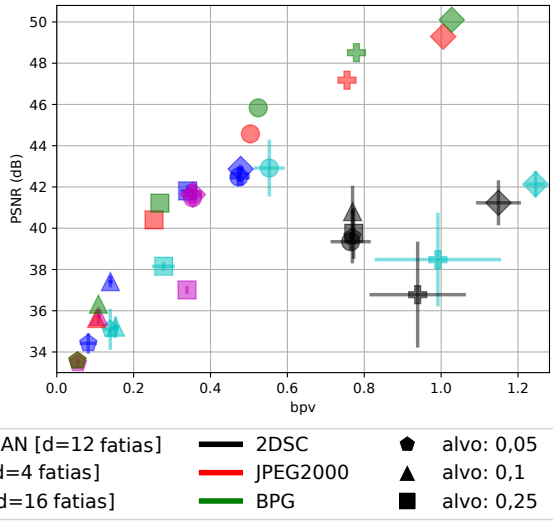


(c) Penobscot3D

(d) Poseidon3D



(e) Waihapa3D



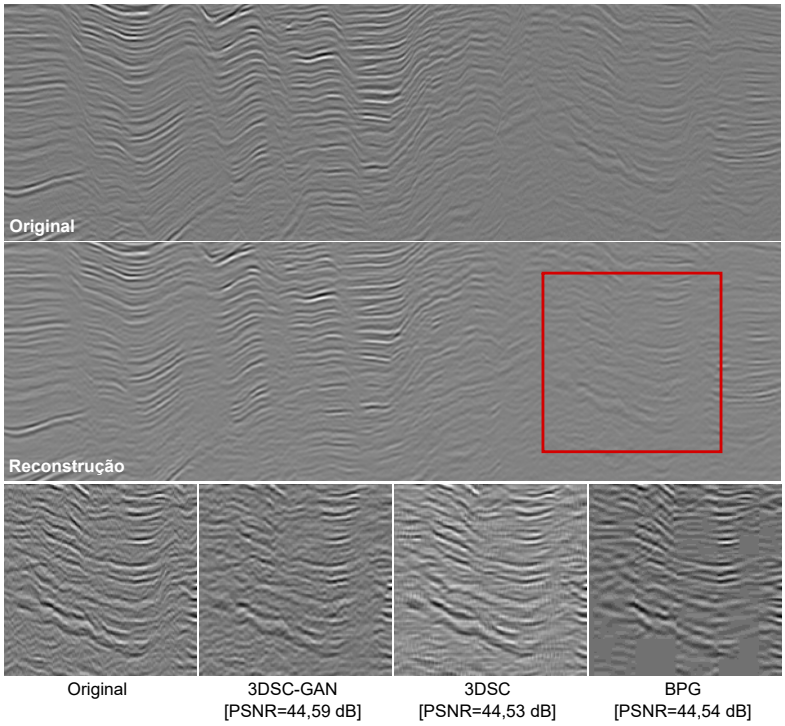
Fonte: Elaborada pelo autor (2022).

apresentados resultados para o compressor BPG, que se baseia no codec HEVC, e para o JPEG2000. Em ambos compressores, fatias 2D dos volumes são comprimidos separadamente. Todos os métodos baseados em aprendizado profundo, incluindo o 3DSC-GAN, são treinados e testados 3 vezes. Os símbolos presentes na Figura 14 representam o resultado médio para o PSNR e bpv. O desvio-padrão é representado por linhas verticais (PSNR) e horizontais (bpv). Em todos os casos, para uma dada taxa de bits alvo, a compressão é mais estável para o JPEG2000 e BPG em termos de alcance da taxa de bits alvo.

Os métodos que utilizam aprendizado profundo nem sempre conseguem atingir a taxa de bits alvo para um dado conjunto de hiperparâmetros. Entretanto, eles se desempenham bem na compressão com taxas de bits mais baixas ($r_t \leq 0{,}50$ bpv). Todos os métodos baseados em aprendizado profundo convergem para taxas de bits alvo mais baixas. Considerando apenas essas taxas de bits ($r_t \leq 0{,}50$ bpv), o 3DSC-GAN é mais eficiente comparado com as outras abordagens que utilizam aprendizado profundo para todos os volumes. Além disso, ele obteve uma eficácia equivalente aos compressores padrões para os volumes Kahu3D e Waihapa3D, como pode ser observado nas Figuras 14a e 14e, respectivamente. Nos volumes Opunake3D e Penobscot3D, apesar de possuírem uma complexidade baixa para compressão, os compressores de propósito geral são superados pelo 3DSC-GAN em taxas de bits alvo mais baixas ($r_t \leq 0{,}25$ bpv). O método tem uma tendência de preservar melhor as baixas frequências e suavizar as mais altas, o que explica o seu PSNR mais baixo para o volume mais ruidoso Poseidon3D (Figura 14d).

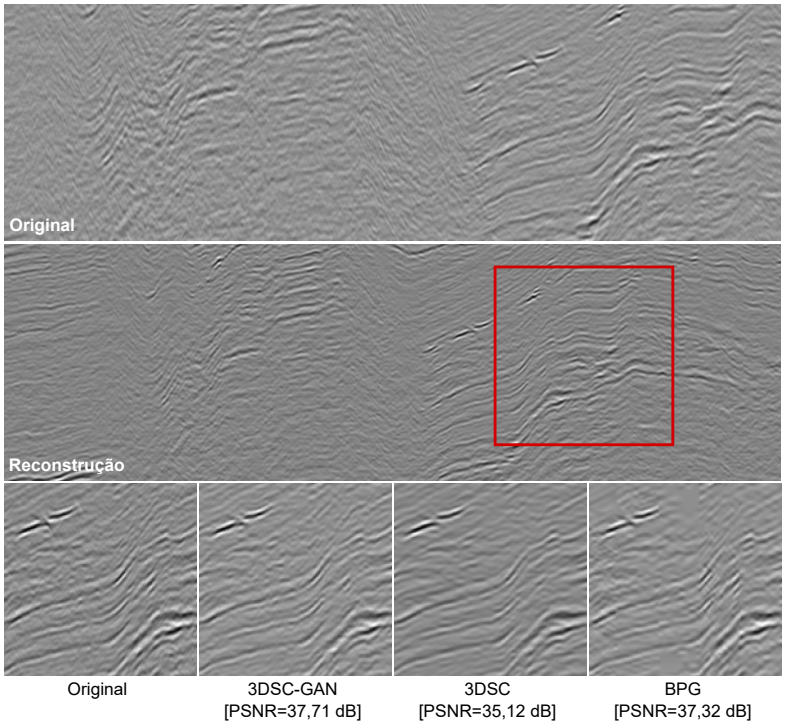
Uma comparação qualitativa é realizada com o objetivo de avaliar visualmente a eficácia do método 3DSC-GAN para compressão. As Figuras 15, 16 e 17 apresentam fatias reconstruídas com $r_t = 0{,}1$ bpv para os volumes Opunake3D, Kahu3D e Poseidon3D, respectivamente, representando cada nível de complexidade para compressão. Além disso, uma visão mais detalhada da reconstrução é selecionada (quadrado vermelho) e comparada com os métodos 3DSC e BPG, visto que esse foi mais eficiente que o JPEG2000 (Figura 14). Observando as figuras, é notável que o método 3DSC-GAN melhora o 3DSC em altas frequências, obtendo uma melhor reconstrução qualitativa e quantitativa para todos os volumes. O mesmo acontece ao se comparar o 3DSC-GAN com o BPG, onde o 3DSC-GAN atinge uma qualidade visual superior ao do BPG e obtém um valor de PSNR superior para os volumes Opunake3D e Kahu3D, sendo um pouco inferior ao BPG para o volume Poseidon3D (Figura 17). Embora o BPG consiga obter um resultado quantitativo ligeiramente superior em termos de PSNR (29,57 dB) para o volume Poseidon3D (Figura 17), é possível observar que ele suaviza demasiadamente estruturas horizontais com frequências intermediárias. Também é evidente que o BPG introduz artefatos no canto inferior direito da região selecionada. Outras regiões suavizadas e com artefatos podem ser observadas no volume Opunake3D (Figura 15) e no Kahu3D (Figura 16). Nesse sentido, o 3DSC-GAN manteve melhor os detalhes estruturais em comparação com os outros dois métodos, possivelmente por considerar a similaridade da vizinhança do dado 3D.

Figura 15 – Comparação da reconstrução de uma fatia 2D (200 × 650 pixels) do volume Opunake3D comprimida com uma taxa de bits alvo extremamente baixa (0,1 bpv).



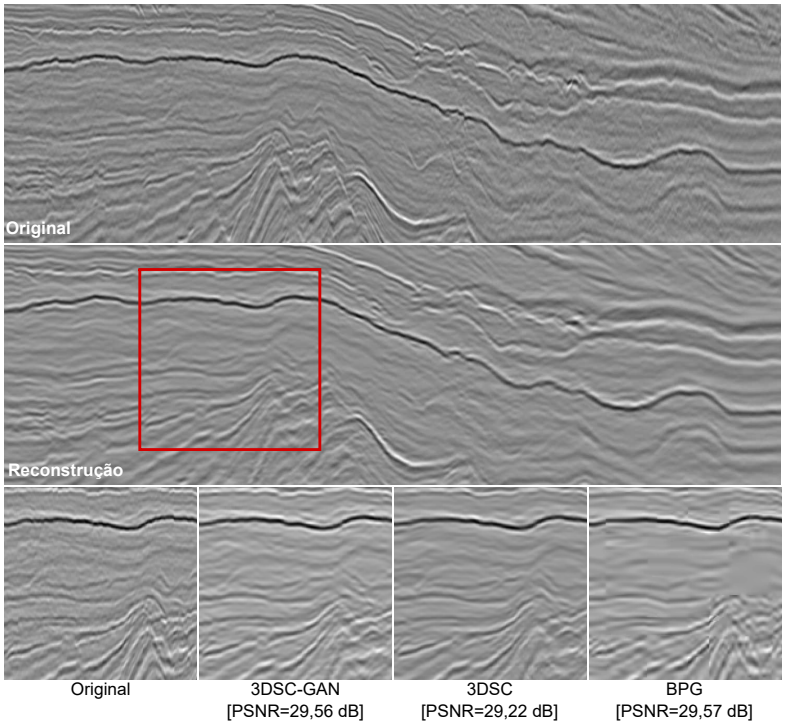
Fonte: Elaborada pelo autor (2022).

Figura 16 – Comparação da reconstrução de uma fatia 2D (200 × 650 pixels) do volume Kahu3D comprimida com uma taxa de bits alvo extremamente baixa (0,1 bpv).



Fonte: Elaborada pelo autor (2022).

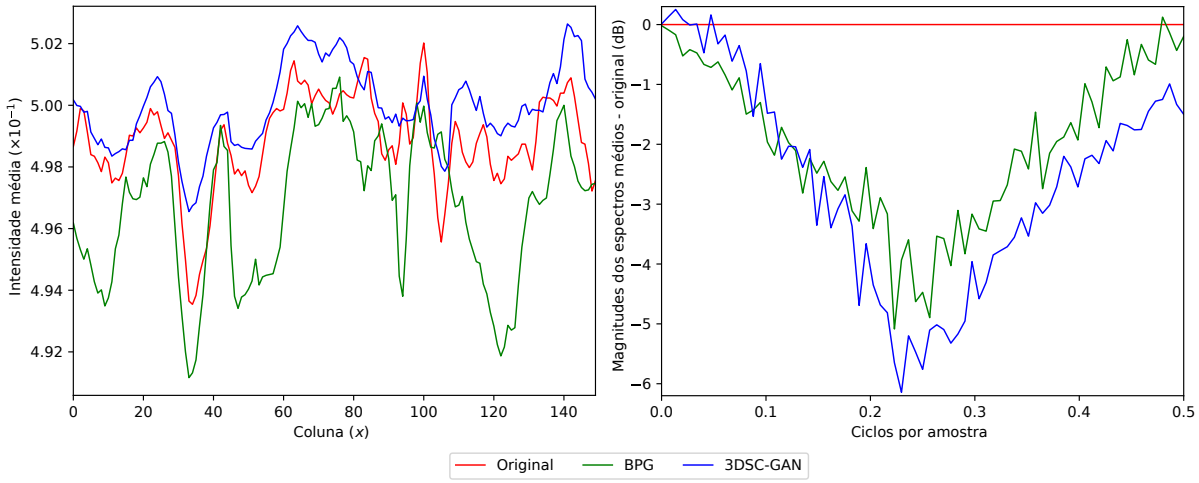
Figura 17 – Comparação da reconstrução de uma fatia 2D (200 × 650 pixels) do volume Poseidon3D comprimida com uma taxa de bits alvo extremamente baixa (0,1 bpv).



Fonte: Elaborada pelo autor (2022).

Outro experimento realizado tem o intuito de comparar os perfis de linha dos métodos 3DSC-GAN e BPG. A Figura 18 apresenta a média das linhas horizontais (esquerda) dentro do quadrado vermelho da Figura 17, e a média da magnitude dos espectros médios das linhas horizontais subtraídas da média do sinal original (direita). Considerando os perfis do gráfico à esquerda, o 3DSC-GAN (azul) é mais aderente à média do sinal original (vermelho), preservando melhor as oscilações médias. O BPG (verde) claramente distorce o sinal médio entre as colunas 110 e 140. Perfis similares podem ser encontrados em outras regiões da mesma fatia 2D do Poseidon3D (Figura 17). No gráfico à direita, a média dos espectros do sinal original é subtraída das três médias dos espectros: original (vermelho), reconstrução com o 3DSC-GAN (azul), e reconstrução com o BPG (verde). O espectro médio original consequentemente passa a ser zero para uma melhor comparação. A média do 3DSC-GAN é mais próxima da média original nas baixas frequências. O BPG suaviza mais das baixas até parte das médias frequências, mas preserva o espectro médio entre parte das médias até às altas frequências, provavelmente adicionando distorções e artefatos, explicando seu PSNR mais alto. Estes resultados sugerem que a compressão através de aprendizado profundo é uma área promissora, e que uma função de perda mais adequada pode melhorar a eficácia desses tipos de métodos.

Figura 18 – Média de todas as linhas horizontais do interior do quadrado vermelho (Figura 17) para o dado original e suas reconstruções para os métodos 3DSC-GAN e BPG: perfis de linha médios (esquerda) e espectro médio subtraído do original (direita).



Fonte: Elaborada pelo autor (2022).

4.3 EXPERIMENTOS PARA O MÉTODO 3DSC-GAN COM FUNÇÃO DE PERDA MULTIDIMENSIONAL COMPOSTA

Nesta seção, a mesma arquitetura 3DSC-GAN apresentada na Seção 3.1 é utilizada nos experimentos. Porém, a função de distorção $\mathbf{d}(\mathbf{x}, \hat{\mathbf{x}})$ é substituída pela função de perda multidimensional composta $\mathbf{d_{composta}}(\mathbf{x}, \hat{\mathbf{x}})$ (Equação (3.5)). Para isso, $\mathbf{d_{composta}}(\mathbf{x}, \hat{\mathbf{x}})$ é utilizada nas funções de perda do módulo de probabilidade (Equação (2.7)) e do autocodificador (Equação (2.10)). Com isso, é esperado que essas alterações aprimorem a capacidade de reconstrução do método 3DSC-GAN. Vale ressaltar que em todas as tabelas e gráficos, as comparações quantitativas são apresentadas com o valor de PSNR entre o dado original e o reconstruído, pois ele é interessante para se comparar a capacidade da função $\mathbf{d_{composta}}(\mathbf{x}, \hat{\mathbf{x}})$ de promover a reconstrução do dado.

Esta seção está dividida em 6 subseções que abordam a configuração da rede e os experimentos realizados com a função $\mathbf{d_{composta}}(\mathbf{x}, \hat{\mathbf{x}})$. A Subseção **4.3.1** apresenta a configuração da rede utilizada no treinamento. Em seguida, a Subseção **4.3.2** aborda os experimentos que consistem em descobrir a melhor combinação de funções. Com esse intuito, são selecionadas as 5 funções apresentadas na Seção 3.2: *Huber* (Equação (3.6)), $L1$ (Equação (3.7)), $L2$ (Equação (3.8)), *LogCosh* (Equação (3.9)) e *Média* (Equação (3.10)). Na Subseção **4.3.3**, os experimentos focam em encontrar qual a ordem das direções *inline*, *crossline*, e *time-depth* que melhor beneficia a reconstrução do dado. Na Subseção **4.3.4** é explorado a ponderação da função $\mathbf{d_{composta}}(\mathbf{x}, \hat{\mathbf{x}})$ aplicando os termos α, θ, e γ. Em seguida, a Subseção **4.3.5** apresenta uma comparação entre os métodos e discussão

dos resultados. Por fim, a Subseção **4.3.6** aborda os resultados obtidos ao aplicar um ajuste fino para cada volume sísmico utilizado nesta dissertação.

4.3.1 Configurações da rede e do treinamento

Todos os modelos foram treinados e testados com o melhor conjunto de hiperparâmetros encontrado na Seção 4.2. Com isso, a entrada do modelo possui as dimensões $160 \times 160 \times 12$ com o tamanho do lote $B = 8$. Durante os primeiros experimentos foi observado uma dificuldade de convergência para as taxas de bits r_t utilizadas na Seção 4.2. Consequentemente, os parâmetros β e o tamanho da representação latente foram alterados para 1 e 32, respectivamente, para que a rede consiga alcançar o valor $r_t = 0{,}1$ bpv. Essa taxa de bits alvo é considerada extrema que é o foco desta dissertação.

4.3.2 Combinação de funções para função de perda composta

O primeiro experimento consiste em buscar qual é a combinação de funções $Huber$, $L1$, $L2$, $LogCosh$ e *Média* que resultam em uma melhor reconstrução. Todas as permutações sem repetição composta por três funções foram utilizadas para treinamento e teste. Entretanto, dados os resultados dos experimentos iniciais e o alto custo computacional, decidiu-se utilizar também as permutações com repetição apenas para os combinações mais promissoras que utilizam as funções $L1$ e $L2$ na primeira redução. O procedimento de treinamento/teste é aplicado 3 vezes para capturar a eficácia média de todos os modelos. Dado que a função *Média* não é propriamente uma métrica, ela não gera bons resultados ao ser utilizada pela primeira redução, logo ela só é aplicada nas demais funções de $\mathbf{d_{composta}}(\mathbf{x}, \hat{\mathbf{x}})$. Neste experimento, a ordem dos eixos é fixada com (x, y, z) representado as direções *inline*, *crossline*, e *time-depth*, respectivamente.

A Tabela 5 apresenta os resultados de reconstrução utilizando a função $L1$ na primeira redução. As melhores médias de PSNR para cada volume do conjunto de teste (coluna) são destacados em cinza, onde as células mais escuras indicam melhor resultado. É possível notar que ao aplicar as funções $L1$ e $L2$ na primeira e segunda redução, respectivamente, são obtidos os melhores resultados de reconstrução para todos os volumes, exceto para o Opunake3D. As combinações que utilizam $LogCosh$ na segunda e terceira reduções possuem os piores resultados dentre os presentes nesta tabela, sugerindo que essa função não preserva ou ressalta as distorções calculadas pelas reduções aplicadas anteriores a ela. Além disso, os melhores resultados para o volume Opunake3D são alcançados quando as funções $L1$ e *Média* são aplicada na segunda redução. Isso indica que essas funções podem auxiliar a atingir uma reconstrução com maior qualidade, mesmo se tratando de um volume com complexidade baixa para compressão. Considerando os valores médios de PSNR, a combinação $(L1, L2, L1)$ possui o melhor resultado. Seus resultados são próximos aos dos melhores para todos volumes. Para o Kahu3D, essa combinação obteve o maior

valor médio.

Tabela 5 – Combinação de funções utilizando a função $L1$ na primeira redução para a taxa de bits alvo $r_t = 0{,}1$ bpv. Resultados correspondentes a uma média de 3 testes, reportados como PSNR/bpv. Células em cinza destacam os melhores resultados em termos de PSNR para cada volume (quanto mais escura, melhor).

Combinação	Volumes				
	Kahu3D	Opunake3D	Penobscot3D	Poseidon3D	Waihapa3D
$(L1, L1, L1)$	40,60 / 0,14	48,32 / 0,12	44,10 / 0,14	27,39 / 0,16	37,98 / 0,16
$(L1, L1, L2)$	40,94 / 0,14	48,70 / 0,12	43,90 / 0,14	28,44 / 0,17	39,01 / 0,16
$(L1, L1, LogCosh)$	40,77 / 0,13	48,30 / 0,10	44,09 / 0,12	27,79 / 0,15	38,34 / 0,15
$(L1, L1, Huber)$	40,76 / 0,16	48,72 / 0,13	43,87 / 0,15	27,38 / 0,17	39,06 / 0,17
$(L1, L1,$ *Média*$)$	40,72 / 0,15	48,15 / 0,13	43,27 / 0,14	27,32 / 0,16	38,50 / 0,16
$(L1, L2, L1)$	41,29 / 0,14	48,49 / 0,11	44,16 / 0,13	28,48 / 0,16	39,38 / 0,16
$(L1, L2, L2)$	40,59 / 0,14	47,85 / 0,11	42,82 / 0,13	28,22 / 0,18	39,05 / 0,16
$(L1, L2, LogCosh)$	41,15 / 0,16	47,75 / 0,14	43,97 / 0,16	28,46 / 0,18	39,31 / 0,18
$(L1, L2, Huber)$	40,98 / 0,14	48,32 / 0,11	44,17 / 0,13	28,47 / 0,16	39,40 / 0,16
$(L1, L2,$*Média*$)$	41,09 / 0,14	48,35 / 0,11	44,05 / 0,13	28,53 / 0,16	39,13 / 0,16
$(L1, LogCosh, L1)$	38,27 / 0,07	44,66 / 0,04	41,66 / 0,06	26,98 / 0,08	35,88 / 0,09
$(L1, LogCosh, L2)$	39,21 / 0,09	44,67 / 0,05	41,42 / 0,06	28,04 / 0,14	37,70 / 0,13
$(L1, LogCosh, LogCosh)$	37,06 / 0,05	43,16 / 0,03	40,50 / 0,04	26,70 / 0,08	34,78 / 0,07
$(L1, LogCosh, Huber)$	38,08 / 0,07	44,65 / 0,04	41,61 / 0,06	27,09 / 0,08	35,95 / 0,08
$(L1, LogCosh,$ *Média*$)$	38,00 / 0,07	44,58 / 0,04	41,40 / 0,05	27,07 / 0,08	35,67 / 0,08
$(L1, Huber, L1)$	40,88 / 0,15	48,67 / 0,12	43,71 / 0,14	27,88 / 0,17	38,95 / 0,17
$(L1, Huber, L2)$	40,82 / 0,14	48,43 / 0,11	43,00 / 0,13	28,42 / 0,16	38,81 / 0,16
$(L1, Huber, LogCosh)$	40,59 / 0,13	48,37 / 0,11	44,13 / 0,13	27,44 / 0,16	39,02 / 0,15
$(L1, Huber, Huber)$	40,55 / 0,13	48,58 / 0,11	43,58 / 0,13	28,14 / 0,15	37,40 / 0,15
$(L1, Huber,$ *Média*$)$	40,65 / 0,14	48,27 / 0,11	43,35 / 0,13	27,67 / 0,15	38,10 / 0,16
$(L1,$ *Média*$, L1)$	40,44 / 0,13	47,94 / 0,11	43,20 / 0,12	27,73 / 0,15	38,31 / 0,14
$(L1,$ *Média*$, L2)$	41,02 / 0,14	48,50 / 0,12	42,82 / 0,14	28,36 / 0,16	39,07 / 0,16
$(L1,$ *Média*$, LogCosh)$	40,01 / 0,12	48,10 / 0,10	43,57 / 0,12	27,81 / 0,15	36,49 / 0,14
$(L1,$ *Média*$, Huber)$	40,72 / 0,14	48,73 / 0,12	44,12 / 0,13	28,05 / 0,16	38,57 / 0,16
$(L1,$ *Média*$,$ *Média*$)$	40,26 / 0,13	48,24 / 0,11	43,52 / 0,12	27,71 / 0,15	38,38 / 0,15

Fonte: Elaborada pelo autor (2022).

A Tabela 6 mostra os resultados de reconstrução com a função $L2$ na primeira redução. As melhores médias de PSNR para cada volume do conjunto de teste (coluna) são destacados em cinza, onde as células mais escuras indicam melhor resultado. Novamente, os piores resultados em termos de PSNR são obtidos ao utilizar a função *LogCosh* nas segunda e terceira reduções, reforçando que *LogCosh* não mantém a distorção capturada pelas reduções anteriores. O melhor resultado obtido utilizando $L2$ na primeira redução é atingido pela combinação ($L2$, *Média, Huber*), que alcança o maior PSNR médio para os volumes Kahu3D, Penobscot3D e Poseidon3D.

Tabela 6 – Combinação de funções utilizando a função $L2$ na primeira redução para a taxa de bits alvo $r_t = 0{,}1$ bpv. Resultados correspondentes a uma média de 3 testes, reportados como PSNR/bpv. Células em cinza destacam os melhores resultados em termos de PSNR para cada volume (quanto mais escura, melhor).

Combinação	Volumes				
	Kahu3D	Opunake3D	Penobscot3D	Poseidon3D	Waihapa3D
($L2, L1, L1$)	41,49 / 0,15	46,18 / 0,08	43,74 / 0,12	28,60 / 0,17	39,57 / 0,17
($L2, L1, L2$)	40,74 / 0,13	45,07 / 0,08	43,51 / 0,12	28,52 / 0,17	38,80 / 0,16
($L2, L1, LogCosh$)	37,67 / 0,06	42,78 / 0,03	40,52 / 0,04	27,92 / 0,14	35,89 / 0,09
($L2, L1, Huber$)	41,44 / 0,14	47,23 / 0,08	43,74 / 0,11	28,58 / 0,17	39,67 / 0,17
($L2, L1,$ *Média*)	41,30 / 0,14	43,53 / 0,06	43,66 / 0,12	28,47 / 0,17	39,36 / 0,17
($L2, L2, L1$)	37,99 / 0,06	41,69 / 0,02	39,71 / 0,03	28,24 / 0,15	36,71 / 0,10
($L2, L2, L2$)	35,79 / 0,06	42,15 / 0,03	39,20 / 0,03	27,86 / 0,16	36,58 / 0,13
($L2, L2, LogCosh$)	35,86 / 0,05	40,99 / 0,02	38,58 / 0,03	27,86 / 0,16	35,41 / 0,10
($L2, L2, Huber$)	37,64 / 0,06	41,29 / 0,02	39,62 / 0,03	28,21 / 0,15	36,84 / 0,11
($L2, L2,$ *Média*)	37,79 / 0,06	41,74 / 0,02	39,82 / 0,03	28,25 / 0,15	37,17 / 0,11
($L2, LogCosh, L1$)	34,98 / 0,04	40,43 / 0,03	38,45 / 0,03	26,78 / 0,14	32,49 / 0,07
($L2, LogCosh, L2$)	32,33 / 0,05	33,00 / 0,04	32,65 / 0,05	24,06 / 0,05	29,21 / 0,05
($L2, LogCosh, LogCosh$)	30,88 / 0,05	32,14 / 0,03	32,29 / 0,04	22,90 / 0,04	27,36 / 0,06
($L2, LogCosh, Huber$)	33,02 / 0,04	37,14 / 0,03	35,68 / 0,04	25,34 / 0,09	29,73 / 0,06
($L2, LogCosh,$*Média*)	34,92 / 0,04	40,60 / 0,02	38,21 / 0,02	26,88 / 0,13	32,69 / 0,07
($L2, Huber, L1$)	41,09 / 0,13	46,89 / 0,08	43,57 / 0,11	28,45 / 0,17	39,13 / 0,16
($L2, Huber, L2$)	40,50 / 0,12	42,71 / 0,06	43,16 / 0,11	28,46 / 0,17	38,62 / 0,15
($L2, Huber, LogCosh$)	37,73 / 0,06	42,36 / 0,03	40,29 / 0,03	27,84 / 0,12	36,06 / 0,09
($L2, Huber, Huber$)	41,45 / 0,14	45,01 / 0,07	43,27 / 0,10	28,59 / 0,17	39,47 / 0,17
($L2, Huber,$ *Média*)	41,32 / 0,13	45,25 / 0,07	43,30 / 0,10	28,57 / 0,17	39,43 / 0,16
($L2,$ *Média*, $L1$)	41,62 / 0,13	45,46 / 0,07	43,81 / 0,11	28,66 / 0,17	39,54 / 0,16
($L2,$*Média*, $L2$)	40,84 / 0,13	45,28 / 0,08	43,51 / 0,12	28,50 / 0,17	39,08 / 0,15
($L2,$ *Média*, $LogCosh$)	37,80 / 0,06	42,21 / 0,03	40,53 / 0,03	27,96 / 0,12	35,81 / 0,09
($L2,$ *Média*, $Huber$)	41,63 / 0,13	46,38 / 0,08	43,82 / 0,11	28,67 / 0,17	39,53 / 0,16
($L2,$ *Média*, *Média*)	41,50 / 0,13	47,22 / 0,08	43,71 / 0,11	28,56 / 0,16	39,48 / 0,16

Fonte: Elaborada pelo autor (2022).

Os resultados de reconstrução aplicando a função $LogCosh$ na primeira redução são apresentados na Tabela 7. As melhores médias de PSNR para cada volume do conjunto de teste (coluna) são destacados em cinza, onde as células mais escuras indicam melhor resultado. Claramente os valores quantitativos das reconstruções são inferiores ao se utilizar a função $LogCosh$, o que sugere que essa função não é adequada para calcular a distorção entre dados sísmicos pós-pilha 3D. A combinação ($LogCosh$, *Média*, $L1$) alcança o melhor PSNR médio para todos os volumes. Entretanto, estes valores ainda são inferiores a todos os presentes na Tabela 5.

A Tabela 8 apresenta os resultados de reconstrução aplicando a função $Huber$ na

Tabela 7 – Combinação de funções utilizando a função *LogCosh* na primeira redução para a taxa de bits alvo $r_t = 0{,}1$ bpv. Resultados correspondentes a uma média de 3 testes, reportados como PSNR/bpv. Células em cinza destacam os melhores resultados em termos de PSNR para cada volume (quanto mais escura, melhor).

Combinação	Volumes				
	Kahu3D	Opunake3D	Penobscot3D	Poseidon3D	Waihapa3D
($LogCosh, L1, L2$)	34,13 / 0,05	38,31 / 0,04	37,09 / 0,04	25,33 / 0,06	30,35 / 0,06
($LogCosh, L1, Huber$)	35,58 / 0,05	41,95 / 0,03	39,10 / 0,03	26,91 / 0,13	33,48 / 0,08
($LogCosh, L1,$ *Média*)	36,57 / 0,05	42,43 / 0,03	39,96 / 0,03	26,94 / 0,09	34,38 / 0,07
($LogCosh, L2, L1$)	30,93 / 0,03	31,77 / 0,03	31,76 / 0,03	22,92 / 0,05	27,68 / 0,05
($LogCosh, L2, Huber$)	30,57 / 0,03	29,00 / 0,04	29,31 / 0,04	22,30 / 0,05	27,98 / 0,05
($LogCosh, L2,$ *Média*)	31,29 / 0,04	32,35 / 0,04	31,92 / 0,04	22,97 / 0,05	27,80 / 0,05
($LogCosh, Huber, L1$)	33,10 / 0,06	34,29 / 0,05	34,08 / 0,06	24,59 / 0,05	30,36 / 0,06
($LogCosh, Huber, L2$)	30,67 / 0,04	33,25 / 0,04	31,92 / 0,04	22,92 / 0,05	27,30 / 0,05
($LogCosh, Huber,$ *Média*)	33,41 / 0,05	36,34 / 0,04	35,25 / 0,05	24,73 / 0,05	29,81 / 0,05
($LogCosh,$ *Média*, $L1$)	36,77 / 0,05	42,97 / 0,03	40,21 / 0,04	27,03 / 0,10	34,77 / 0,07
($LogCosh,$ *Média*, $L2$)	33,73 / 0,04	37,81 / 0,04	36,10 / 0,04	25,42 / 0,08	30,64 / 0,07
($LogCosh,$ *Média*, $Huber$)	35,32 / 0,04	41,65 / 0,03	38,74 / 0,03	26,89 / 0,13	32,88 / 0,07

Fonte: Elaborada pelo autor (2022).

primeira redução. As melhores médias de PSNR para cada volume do conjunto de teste (coluna) são destacados em cinza, onde as células mais escuras indicam melhor resultado. Similar aos resultados obtidos ao utilizar a função *LogCosh* na primeira redução, a função $Huber$ não captura de maneira eficaz a diferença entre os volumes sísmicos. Porém, os resultados obtidos ao aplicá-la na segunda ou terceira redução são superiores aos que utilizam a *LogCosh*. A combinação ($Huber, L1, L2$) alcança um PSNR médio maior nos volumes com baixa dificuldade para compressão. Entretanto, o melhor resultado é obtido com a combinação ($Huber, L2, LogCosh$) que alcança o maior PSNR médio em volumes mais difíceis, como Poseidon3D e Waihapa3D. Esta combinação também alcança valores próximos de PSNR nos demais volumes. Vale ressaltar que esse resultado ainda é inferior ao obtido pela combinação (*LogCosh*, *Média*, $L1$) (Tabela 7). Conclui-se que as funções *LogCosh* e $Huber$ não são adequadas na primeira redução no contexto desta dissertação.

Uma comparação qualitativa é realizada com o objetivo de avaliar visualmente as melhores combinações encontradas no experimento anterior, de acordo com cada tabela (Tabelas 5, 6, 7 e 8). Nesse sentido, as reconstruções dos modelos são examinadas no volume Kahu3D, que possui uma complexidade média para compressão e também uma boa variação de estruturas sísmicas. A Figura 19 apresenta essa comparação entre as combinações ($L1, L2, L1$), ($L2$, *Média*, $Huber$), (*LogCosh*, *Média*, $L1$), e ($Huber, L1, L2$). Nela é possível notar que as combinações ($L1, L2, L1$) e ($L2$, *Média*, $Huber$) possuem os valores de PSNR próximos, onde a combinação ($L2$, *Média*, $Huber$) preserva mais as

Tabela 8 – Combinação de funções utilizando a função *Huber* na primeira redução para a taxa de bits alvo $r_t = 0{,}1$ bpv. Resultados correspondentes a uma média de 3 testes, reportados como PSNR/bpv. Células em cinza destacam os melhores resultados em termos de PSNR para cada volume (quanto mais escura, melhor).

Combinação	Volumes				
	Kahu3D	Opunake3D	Penobscot3D	Poseidon3D	Waihapa3D
($Huber, L1, L2$)	31,49 / 0,08	37,36 / 0,04	35,13 / 0,06	24,40 / 0,12	30,07 / 0,10
($Huber, L1, LogCosh$)	33,15 / 0,06	35,43 / 0,03	32,61 / 0,04	24,51 / 0,09	30,15 / 0,08
($Huber, L1,$ *Média*)	33,83 / 0,07	34,22 / 0,04	32,87 / 0,06	24,34 / 0,10	30,23 / 0,10
($Huber, L2, L1$)	33,34 / 0,07	35,18 / 0,03	33,47 / 0,05	24,46 / 0,11	29,99 / 0,09
($Huber, L2, LogCosh$)	33,14 / 0,06	35,51 / 0,03	32,75 / 0,04	24,92 / 0,10	31,27 / 0,08
($Huber, L2,$ *Média*)	33,72 / 0,07	34,56 / 0,03	32,71 / 0,05	24,44 / 0,11	30,49 / 0,09
($Huber, LogCosh, L1$)	32,23 / 0,06	35,20 / 0,03	33,75 / 0,03	24,67 / 0,10	30,07 / 0,08
($Huber, LogCosh, L2$)	31,40 / 0,06	36,32 / 0,03	34,27 / 0,03	24,30 / 0,10	28,36 / 0,08
($Huber, LogCosh,$ *Média*)	32,84 / 0,06	34,63 / 0,03	32,69 / 0,03	24,49 / 0,10	30,43 / 0,08
($Huber,$ *Média*, $L1$)	33,67 / 0,07	35,91 / 0,04	33,79 / 0,05	24,34 / 0,10	30,33 / 0,09
($Huber,$ *Média*, $L2$)	32,81 / 0,07	35,75 / 0,04	32,98 / 0,05	24,11 / 0,11	29,56 / 0,09
($Huber,$ *Média*, $LogCosh$)	32,84 / 0,06	34,32 / 0,03	31,25 / 0,04	24,34 / 0,09	29,65 / 0,08

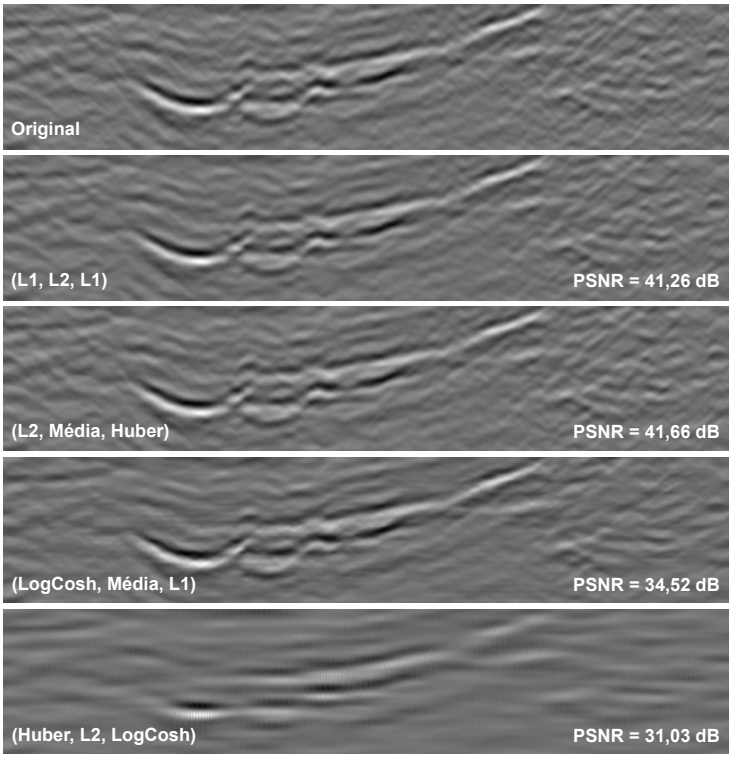
Fonte: Elaborada pelo autor (2022).

estruturas do dado original e a combinação ($L1, L2, L1$) mantém a variação de brilho. A combinação ($LogCosh$, *Média*, $L1$) distorce as estruturas presentes no dado original, resultando em um valor de PSNR relativamente baixo. A combinação ($Huber, L1, L2$), além de atingir o menor valor de PSNR em comparação com as demais, não preserva as informações presentes no dado original, o que resulta em uma reconstrução de baixíssima qualidade. Por conseguir um valor de PSNR um pouco maior e por conseguir manter mais as informações estruturais, a combinação ($L2$, *Média*, $Huber$) é fixada para os próximos experimentos.

4.3.3 Variação dos eixos na função de perda composta

Visto que realizar uma busca em grade, variando a ordem das funções e os eixos, seria algo muito custoso computacionalmente e demandaria vários dias de treinamento e teste, a variação da ordem dos eixos é feita de forma separada. O experimento a seguir consiste em analisar qual a ordem dos eixos preserva mais as informações dos volumes de teste utilizando a melhor combinação encontrada no experimento anterior, ($L2$, *Média*, $Huber$). A Tabela 9 apresenta os resultados de reconstrução variando a ordem dos eixos ao aplicar a combinação de funções ($L2$, *Média*, $Huber$). Os eixos x, y, e z, representam as direções, *inline*, *crossline*, e *time-depth*, respectivamente. O processo de treinamento/teste é aplicado 3 vezes para capturar a eficácia média de todos os modelos. Os melhores valores médios de PSNR para cada volume do conjunto de teste (coluna) são destacados em cinza,

Figura 19 – Comparação da reconstrução de uma fatia 2D (100×500 pixels) do volume Kahu3D comprimida com uma taxa de bits alvo de 0,1 bpv, aplicando as combinações com os melhores resultados presentes nas tabelas 5, 6, 7 e 8.



Fonte: Elaborada pelo autor (2022).

onde as células mais escuras indicam melhor resultado.

Tabela 9 – Variação da ordem dos eixos ao aplicar a combinação (*L2, Média, Huber*) como função de distorção para uma taxa de bits alvo de $r_t = 0{,}1$ bpv. Resultados correspondentes a uma média de 3 testes, sendo reportados como PSNR/bpv. Células em cinza destacam os melhores resultados em termos de PSNR para cada volume (quanto mais escura, melhor).

Ordem Aplicada	Volumes				
	Kahu3D	Opunake3D	Penobscot3D	Poseidon3D	Waihapa3D
(x, y, z)	41,63 / 0,13	45,46 / 0,07	43,82 / 0,11	28,66 / 0,17	39,54 / 0,16
(x, z, y)	41,02 / 0,13	46,60 / 0,07	43,55 / 0,11	28,51 / 0,16	39,19 / 0,16
(y, x, z)	41,16 / 0,13	45,76 / 0,07	43,47 / 0,11	28,43 / 0,17	39,24 / 0,16
(y, z, x)	41,28 / 0,15	47,76 / 0,10	43,73 / 0,13	28,50 / 0,17	39,41 / 0,17
(z, x, y)	41,16 / 0,13	46,25 / 0,07	43,36 / 0,10	28,56 / 0,16	39,34 / 0,16
(z, y, x)	41,21 / 0,13	46,66 / 0,07	43,59 / 0,11	28,59 / 0,17	39,47 / 0,16

Fonte: Elaborada pelo autor (2022).

Os resultados obtidos para o volume Penobscot3D não são sensíveis à ordem dos eixos em que as funções *L2*, *Média* e *Huber* são aplicadas. Visto que esse volume tem

baixa complexidade para compressão, a rede tem facilidade em comprimir em todas as combinações. Isto também pode ser observado no volume Poseidon3D, que possui complexidade alta, onde a rede não retém mais informação desse volume ruidoso apenas trocando a ordem dos eixos. É importante notar que a ordem *inline*, *crossline* e *time-depth* é a única ordem de eixos que foi capaz de atingir os maiores valores de PSNR para grande parte dos volumes de teste. Com isso, a ordem (x, y, z) é fixada para os próximos experimentos.

4.3.4 Função de perda composta ponderada

As funções presentes na função composta $\mathbf{d_{composta}}(\mathbf{x}, \hat{\mathbf{x}})$ (Equação (3.5)) podem ser ponderadas separadamente para atingir uma melhorar reconstrução. Neste sentido, a Equação (3.5) pode ser reescrita na seguinte forma:

$$\mathbf{d_{composta}}(\mathbf{x}, \hat{\mathbf{x}}) = \gamma \cdot f_3(\theta \cdot f_2(\alpha \cdot f_1(\mathbf{E}(\mathbf{x}, \hat{\mathbf{x}})))), \tag{4.1}$$

onde os termos α, θ, $\gamma \in \mathbb{R} > 0$ ponderam as funções f_1, f_2, e f_3, respectivamente. Note que caso estes termos assumam o valor 1,0, a função $\mathbf{d_{composta}}(\mathbf{x}, \hat{\mathbf{x}})$ é equivalente à utilizada nos experimentos das Subseções **4.3.2** e **4.3.3**. Por outro lado, a contribuição de cada função pode ser maior/menor ao aumentar/diminuir o valor do seu respectivo termo.

A Tabela 10 mostra os resultados da compressão para todos os volumes do conjunto de teste obtidos ao variar α, θ, e γ. Cada termo é variado separadamente com os valores {0,6, 0,8, 1,2, 1,4}, sendo que ao assumir o valor 1,0, o termo é considerado como fixado. O processo de treinamento/teste é aplicado 3 vezes, os resultados são reportados como PSNR/bpv. Para realizar todas as combinações de pesos seriam necessários vários dias de treinamento e teste. Com objetivo de reduzir o custo computacional, os pesos foram variados em apenas uma função, enquanto os outros dois pesos foram fixados em 1,0.

É possível observar que as combinações de pesos em que os termos α e θ possuem os valores 0,6 e 0,8 resultam em valores de PSNR mais baixos. A combinação (1,4, 1,0, 1,0) consegue uma das melhores reconstruções em termos de PSNR para o volume Opunake3D, porém os valores obtidos para os demais volumes não se destacam. A variação realizada no termo γ resulta nos maiores valores de PSNR. A melhor combinação de pesos encontrada neste experimento é (1,0, 1,0, 1,2), obtendo um dos melhores resultados para todos os volumes, além de atingir o maior valor de PSNR para os volumes Kahu3D, Opunake3D, Penobscot3D e Waihapa3D. Com isso, o valor de γ é fixado em 1,2 nos próximos experimentos.

A Tabela 11 apresenta os resultados da compressão para todos os volumes do conjunto de teste obtidos ao fixar $\gamma = 1{,}2$ e variar os termos α e θ. Os termos são variados separadamente, assumindo os valores {0,6, 0,8, 1,2, 1,4}. O processo de treinamento/teste é aplicado 3 vezes e os pesos estão reportados como (α, θ, γ). Os melhores resultados foram

Tabela 10 – Variação dos pesos α, θ e γ para uma taxa de bits $r_t = 0{,}1$ bpv utilizando a combinação (*L2, Média, Huber*) para todos os volumes do conjunto de teste. Pesos reportados como (α, θ, γ) e resultados correspondentes a uma média de 3 testes, reportados como PSNR/bpv. Células em cinza destacam os melhores resultados em termos de PSNR para cada volume (quanto mais escura, melhor).

Pesos	Volumes				
	Kahu3D	Opunake3D	Penobscot3D	Poseidon3D	Waihapa3D
(1,0, 1,0, 1,0)	41,63 / 0,13	45,46 / 0,07	43,82 / 0,11	28,66 / 0,17	39,54 / 0,16
(1,0, 1,0, 0,6)	41,28 / 0,12	45,40 / 0,06	43,29 / 0,10	28,52 / 0,16	39,38 / 0,16
(1,0, 1,0, 0,8)	41,44 / 0,13	46,60 / 0,07	43,30 / 0,10	28,64 / 0,16	39,59 / 0,16
(1,0, 1,0, 1,2)	41,65 / 0,15	47,88 / 0,10	44,23 / 0,13	28,63 / 0,17	39,64 / 0,17
(1,0, 1,0, 1,4)	41,48 / 0,14	46,40 / 0,08	43,98 / 0,12	28,56 / 0,17	39,51 / 0,16
(1,0, 0,6, 1,0)	40,70 / 0,12	45,63 / 0,06	42,62 / 0,09	28,37 / 0,17	39,07 / 0,16
(1,0, 0,8, 1,0)	41,11 / 0,13	45,74 / 0,07	43,42 / 0,11	28,48 / 0,17	39,30 / 0,16
(1,0, 1,2, 1,0)	41,28 / 0,14	46,43 / 0,08	43,93 / 0,12	28,56 / 0,17	39,50 / 0,16
(1,0, 1,4, 1,0)	41,39 / 0,14	46,52 / 0,08	43,90 / 0,12	28,57 / 0,17	39,46 / 0,16
(0,6, 1,0, 1,0)	40,75 / 0,12	44,81 / 0,06	42,71 / 0,09	28,45 / 0,16	39,08 / 0,15
(0,8, 1,0, 1,0)	40,90 / 0,15	46,02 / 0,12	43,07 / 0,14	28,40 / 0,18	39,03 / 0,17
(1,2, 1,0, 1,0)	41,34 / 0,14	46,60 / 0,08	43,90 / 0,12	28,60 / 0,17	39,45 / 0,17
(1,4, 1,0, 1,0)	41,05 / 0,13	46,86 / 0,08	43,63 / 0,11	28,46 / 0,16	39,10 / 0,15

Fonte: Elaborada pelo autor (2022).

obtidos com as combinações (1,0, 1,0, 1,2), (1,0, 1,2, 1,2), e (1,4, 1,0, 1,2). Elas proveem os melhores resultados em praticamente todos volumes do conjunto de teste. A melhor combinação é obtida pelos pesos (1,0, 1,2, 1,2), atingindo o maior valor de PSNR para todos volumes, exceto para o Poseidon3D com uma pequena margem. Consequentemente, o valor de $\theta = 1{,}2$ é fixado para a próxima busca.

A Tabela 12 mostra os resultados com os pesos θ e γ fixados em 1,2 e com o termo α assumindo os valores {0,6, 0,8, 1,2, 1,4}. O processo de treinamento/teste é aplicado 3 vezes e os pesos estão reportados como (α, θ, γ). Observando os valores de PSNR obtidos, é possível notar que os maiores valores de PSNR são obtidos com $\alpha \geq 1{,}0$ para a maior parte dos volumes de teste. Vale ressaltar que a combinação (1,2, 1,2, 1,2) possui valores de PSNR inferior em todos os volumes em comparação com a combinação padrão (1,0, 1,0, 1,0) (Tabela 10). Porém, a combinação (1,2, 1,2, 1,2) consegue um melhor valor de PSNR para o volume Opunake3D, mesmo ambas combinações possuindo os mesmos valores de peso para todas as funções. A combinação com o melhor resultado em termos de PSNR continua sendo a (1,0, 1,2, 1,2) para todos os volumes de teste. Também é importante ressaltar que após otimizar a combinação (1,0, 1,0, 1,0) (Tabela 10), os resultados obtidos com a combinação (1,0, 1,2, 1,2) atingiu um valor de PSNR superior em todos os volumes, exceto no Poseidon3D, no qual obteve um valor igual. Esse resultado indica que a ponderação da função composta pode auxiliar o método 3DSC-GAN

Tabela 11 – Variação dos pesos α e θ, com $\gamma = 1{,}2$ para uma taxa de bits $r_t = 0{,}1$ bpv utilizando a combinação (*L2, Média, Huber*) para todos os volumes do conjunto de teste. Pesos reportados como (α, θ, γ) e resultados correspondentes a uma média de 3 testes, reportados como PSNR/bpv. Células em cinza destacam os melhores resultados em termos de PSNR para cada volume (quanto mais escura, melhor).

Pesos	Volumes				
	Kahu3D	Opunake3D	Penobscot3D	Poseidon3D	Waihapa3D
(1,0, 1,0, 1,2)	41,65 / 0,15	47,88 / 0,10	44,23 / 0,13	28,63 / 0,17	39,64 / 0,17
(1,0, 0,6, 1,2)	41,24 / 0,13	45,94 / 0,07	43,22 / 0,10	28,54 / 0,17	39,35 / 0,16
(1,0, 0,8, 1,2)	41,35 / 0,13	46,49 / 0,07	43,54 / 0,11	28,55 / 0,16	39,26 / 0,16
(1,0, 1,2, 1,2)	41,79 / 0,15	48,03 / 0,10	44,24 / 0,13	28,66 / 0,18	39,74 / 0,17
(1,0, 1,4, 1,2)	41,21 / 0,14	46,74 / 0,08	43,88 / 0,12	28,46 / 0,17	39,36 / 0,16
(0,6, 1,0, 1,2)	41,40 / 0,12	45,14 / 0,06	43,38 / 0,09	28,58 / 0,16	39,52 / 0,15
(0,8, 1,0, 1,2)	41,22 / 0,12	46,17 / 0,07	43,32 / 0,10	28,62 / 0,16	39,48 / 0,15
(1,2, 1,0, 1,2)	41,11 / 0,14	47,13 / 0,09	43,57 / 0,12	28,50 / 0,17	39,33 / 0,17
(1,4, 1,0, 1,2)	41,51 / 0,14	46,82 / 0,09	44,12 / 0,12	28,67 / 0,17	39,60 / 0,16

Fonte: Elaborada pelo autor (2022).

a alcançar valores de PSNR superiores.

Tabela 12 – Variação dos pesos de α, com $\theta = \gamma = 1{,}2$ para uma taxa de bits $r_t = 0{,}1$ bpv utilizando a combinação (*L2, Média, Huber*) para todos os volumes do conjunto de teste. Pesos reportados como (α, θ, γ) e resultados correspondentes a uma média de 3 testes, reportados como PSNR/bpv. Células em cinza destacam os melhores resultados em termos de PSNR para cada volume (quanto mais escura, melhor).

Pesos	Volumes				
	Kahu3D	Opunake3D	Penobscot3D	Poseidon3D	Waihapa3D
(0,6, 1,2, 1,2)	41,34 / 0,14	46,63 / 0,07	43,47 / 0,11	28,51 / 0,17	39,32 / 0,17
(0,8, 1,2, 1,2)	41,17 / 0,13	46,87 / 0,08	43,65 / 0,11	28,49 / 0,17	39,28 / 0,16
(1,0, 1,2, 1,2)	41,79 / 0,15	48,03 / 0,10	44,24 / 0,13	28,66 / 0,18	39,74 / 0,17
(1,2, 1,2, 1,2)	41,24 / 0,14	47,04 / 0,09	43,73 / 0,12	28,53 / 0,16	39,43 / 0,16
(1,4, 1,2, 1,2)	41,25 / 0,14	46,53 / 0,08	43,79 / 0,12	28,45 / 0,17	39,32 / 0,16

Fonte: Elaborada pelo autor (2022).

4.3.5 Comparação entre os métodos e discussão

A comparação entre os métodos 3DSC-GAN, 3DSC-GAN com funções compostas, 3DSC e BPG, é apresentada na Tabela 13 para todos os volumes do conjunto de teste com uma taxa de bits $r_t = 0{,}1$ bpv. Os métodos baseados em aprendizado profundo foram treinados conforme os melhores parâmetros reportados. Assim como nos experimentos presentes na Seção 4.2, o BPG foi selecionado por atingir resultados competitivos em comparação com os demais métodos em termos de PSNR. Vale ressaltar que mesmo a

métrica PSNR não sendo a melhor opção para validar a qualidade de reconstrução de dados sísmicos do tipo pós-pilha, ela ainda é uma métrica genérica amplamente utilizada para comparação de campos escalares, logo, ela é utilizada como comparador quantitativo.

Tabela 13 – Comparação dos resultados da reconstrução para todos os volumes do conjunto de teste com uma taxa de bits $r_t = 0{,}1$ bpv. Resultados reportados como PSNR/bpv (diferença percentual com o melhor resultado). Células em cinza destacam o maior valor de PSNR para cada volume, e a maior diferença percentual está destacada em negrito.

Volumes	Métodos			
	3DSC	3DSC-GAN (PSNR)	3DSC-GAN (*L2*, *Média*, *Huber*)	BPG
Kahu3D	39,39 / 0,14 (**-5,7%**)	40,45 / 0,13 (-3,2%)	41,79 / 0,14	40,37 / 0,11 (-3,4%)
Opunake3D	44,26 / 0,11 (-8,6%)	45,22 / 0,07 (-6,7%)	48,45 / 0,10	44,23 / 0,10 (**-8,7%**)
Penobscot3D	41,93 / 0,13 (-5,3%)	42,75 / 0,09 (-3,4%)	44,27 / 0,12	40,37 / 0,09 (**-8,8%**)
Poseidon3D	27,80 / 0,14 (-4,7%)	27,58 / 0,13 (**-5,5%**)	28,69 / 0,17 (-1,6%)	29,17 / 0,11
Waihapa3D	35,20 / 0,15 (**-11,4%**)	37,41 / 0,14 (-5,9%)	39,74 / 0,17	36,32 / 0,11 (-8,6%)

Fonte: Elaborada pelo autor (2022).

Após observar os resultados presentes na Tabela 13, é possível notar que o método 3DSC-GAN com funções compostas atinge o maior valor de PSNR em todos os volumes de teste, exceto para o volume Poseidon3D. A nova função de perda composta trouxe uma melhora significativa no valor de PSNR em comparação com os métodos base, sendo superior em todos os volumes. As diferenças mais expressivas para estes métodos são encontradas nos volumes Opunake3D e Waihapa3D, chegando a 11,4%. Em comparação com o BPG, as diferenças são superiores a 8,5% para os volumes Opunake3D, Penobscot3D e Waihapa3D. Considerando o volume Poseidon3D, por ser ruidoso, temos que o BPG mantém um PSNR maior do que os métodos baseados em aprendizado profundo. Entretanto, o método 3DSC-GAN com funções compostas obteve uma diferença de apenas 1,6%, enquanto os métodos base têm PSNR mais do que 4,6% inferiores, indicando que ao utilizar uma função objetivo mais apropriada, é possível aprimorar a compressão com relação ao critério PSNR.

Recentemente, no trabalho de Zhang et al. (2018), foi proposta uma nova métrica perceptual que utiliza características profundas obtidas por diferentes arquiteturas de redes neurais. A métrica é chamada similaridade perceptual aprendida de trecho de imagem (*Learned Perceptual Image Patch Similarity* – LPIPS), que calcula a distância entre as características profundas das imagens de entrada. Quanto mais alta a LPIPS, mais diferentes são as imagens e quanto menor, mais semelhantes. Zhang et al. (2018) argumenta que a métrica LPIPS é próxima ao julgamento perceptual humano. Dessa forma, a métrica LPIPS foi utilizada para comparar as mesmas reconstruções apresentadas na Tabela 13.

As Tabelas 14 e 15 apresentam uma comparação entre os métodos de cálculo do LPIPS que usam a arquitetura AlexNet (KRIZHEVSKY; SUTSKEVER; HINTON, 2012)

e a VGG (SIMONYAN; ZISSERMAN, 2014), respectivamente. A versão 0.1.4 do LPIPS disponibilizada pelos próprios autores é aplicada para calcular as distâncias. Assim como na comparação anterior, os métodos 3DSC, 3DSC-GAN com função de distorção PSNR, 3DSC-GAN com funções compostas, e BPG são comparados. Todos os métodos são avaliados nos volumes do conjunto de teste com a taxa de bits $r_t = 0{,}1$ bpv.

Tabela 14 – Comparação dos resultados da reconstrução para todos os volumes do conjunto de teste com uma taxa de bits $r_t = 0{,}1$ bpv. Resultados utilizando a métrica LPIPS com a arquitetura AlexNet. Células em cinza destacam a menor distância encontrada em cada volume.

Volumes	Métodos			
	3DSC	3DSC-GAN (PSNR)	3DSC-GAN ($L2$, *Média*, $Huber$)	BPG
Kahu3D	0,02342	0,0108	0,00921	0,01633
Opunake3D	0,00408	0,00327	0,00129	0,00431
Penobscot3D	0,01284	0,0089	0,0046	0,01671
Poseidon3D	0,12249	0,12208	0,0846	0,08413
Waihapa3D	0,05546	0,02165	0,01205	0,05472

Fonte: Elaborada pelo autor (2022).

Tabela 15 – Comparação dos resultados da reconstrução para todos os volumes do conjunto de teste com uma taxa de bits $r_t = 0{,}1$ bpv. Resultados utilizando a métrica LPIPS com a arquitetura VGG. Células em cinza destacam a menor distância encontrada em cada volume.

Volumes	Métodos			
	3DSC	3DSC-GAN (PSNR)	3DSC-GAN ($L2$, *Média*, $Huber$)	BPG
Kahu3D	0,08982	0,05182	0,04879	0,10407
Opunake3D	0,04231	0,04226	0,03446	0,12337
Penobscot3D	0,0617	0,05604	0,03317	0,12415
Poseidon3D	0,20408	0,19399	0,15374	0,19111
Waihapa3D	0,10764	0,07225	0,05172	0,13431

Fonte: Elaborada pelo autor (2022).

Observando a Tabela 14 é possível notar que os resultados da LPIPS com AlexNet assemelha-se aos resultados utilizando o PSNR como métrica de distorção (Tabela 13). Com isso, o BPG consegue a menor distância entre as características obtidas apenas no volume Poseidon3D, e para os demais volumes, a menor distância é obtida pelo método 3DSC-GAN com funções compostas.

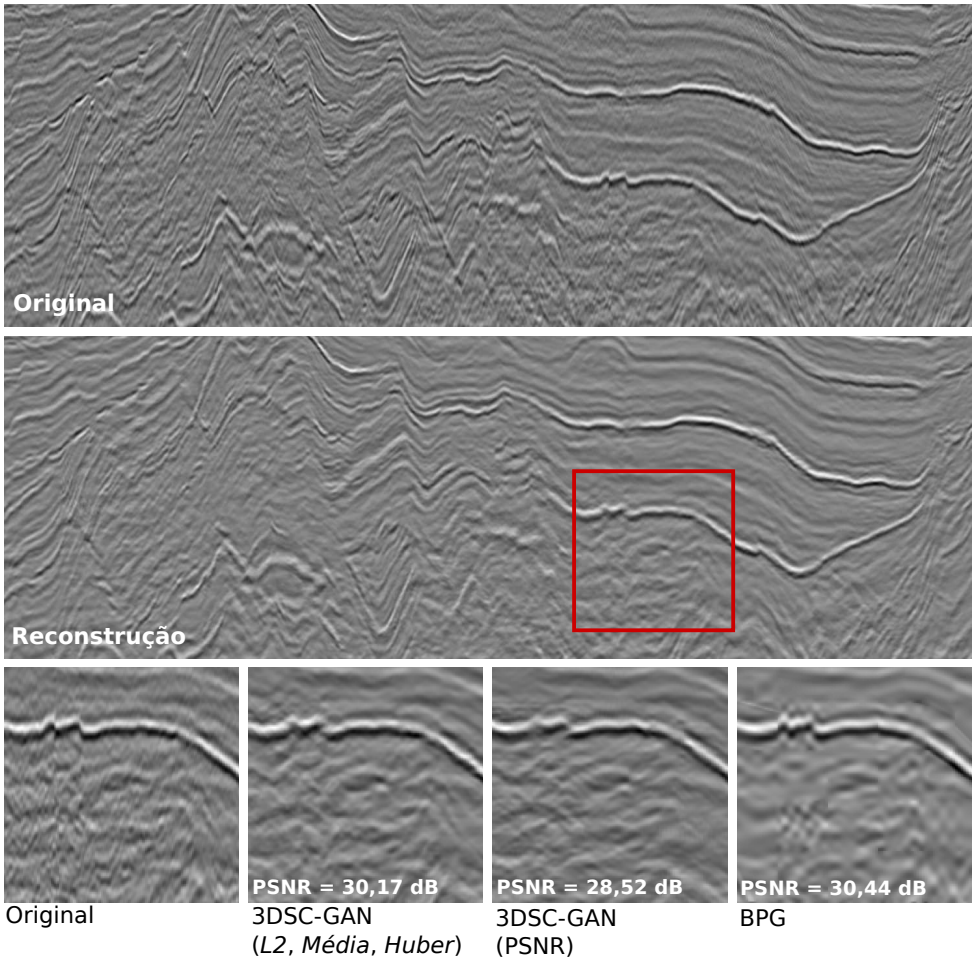
A Tabela 15 apresenta os resultados obtidos pela LPIPS utilizando a rede VGG e os resultados presentes nela diferem dos utilizando o PSNR e a LPIPS com a AlexNet (Tabelas 13 e 14). O método 3DSC-GAN com funções compostas obtém a menor distância

para todos os volumes do conjunto de teste. O BPG possui a maior dissimilaridade para os todos os volumes, exceto para o Poseidon3D, indicando que a LPIPS com a rede VGG distingue melhor as características qualitativas de um dado sísmico do tipo pós-pilha. É interessante salientar que o 3DSC-GAN com função de distorção PSNR não obteve um valor de PSNR superior ao 3DSC para a taxa de bits $r_t = 0{,}1$ bpv (Figura 14d), mesmo com um resultado qualitativo superior visualmente (Figura 17). Entretanto, na Tabela 15 é possível notar a melhora quantitativa do método 3DSC-GAN com função de distorção PSNR comparado com o método base.

A Figura 20 mostra uma reconstrução gerada pelo método 3DSC-GAN com funções compostas para o volume Poseidon3D com uma taxa de bits $r_t = 0{,}1$ bpv. Uma visão mais detalhada da região marcada pelo quadrado vermelho é selecionada para comparação com os métodos 3DSC-GAN com PSNR, e o BPG. O método 3DSC-GAN com a função composta ($L2$, *Média*, $Huber$) preserva melhor a variação de estruturas do dado em comparação com o 3DSC-GAN com PSNR, e atinge um valor de PSNR superior (30,17 dB). Mesmo com um valor maior de PSNR (30,44 dB), a qualidade da reconstrução do método BPG continua sendo inferior aos demais métodos. Vários artefatos podem ser encontrados na reconstrução do BPG, como detalhes perdidos e estruturas muito suavizadas. Neste exemplo, o 3DSC-GAN com funções compostas melhor preservou os detalhes estruturais do dado original.

Assim como na Figura 18, os perfis de linha e magnitude dos espectros são avaliados. A Figura 21 apresenta a média das linhas horizontais (esquerda) da região interna do quadrado vermelho da Figura 20, e as médias dos espectros das linhas horizontais (direita). Analisando os perfis de linha do gráfico da esquerda, é notório que o método 3DSC-GAN com funções compostas (azul) é mais aderente ao original (vermelho), preservando as oscilações médias do sinal. O 3DSC-GAN utilizando PSNR (ciano) possui regiões com ganho médio de intensidade próximo às colunas 30, 70, 100 e 140. Por outro lado, o BPG (verde) reduz a intensidade média em todo o sinal, e mais acentuadamente entre as colunas 80 e 100. Além disso, o BPG suaviza demasiadamente a imagem a partir da coluna 120. Observando o gráfico da direita, o espectro médio original é subtraído das médias dos espectros para uma melhor percepção das diferenças entre os métodos. O 3DSC-GAN com funções compostas (azul) preserva as baixas e médias frequências, sendo mais próximo do sinal médio original (vermelho). O BPG (verde) e o 3DSC-GAN com PSNR (ciano) possuem uma eficácia similar nas baixas e médias frequências. Os três métodos possuem uma perda semelhante nas altas frequências, mas o BPG consegue preservá-las ligeiramente mais do que os métodos baseados em aprendizado profundo. Os três métodos se comportam de forma similar em outras regiões de referência na fatia exemplo da Figura 20. Os resultados sugerem que a nova função de perda composta para dados multidimensionais é promissora, melhorando a eficácia do método 3DSC-GAN, além de auxiliá-lo a conseguir uma reconstrução mais próxima do dado sísmico original.

Figura 20 – Comparação da reconstrução de uma fatia 2D (300 × 900 pixels) do volume Poseidon3D comprimida com uma taxa de bits alvo extremamente baixa (0,1 bpv).



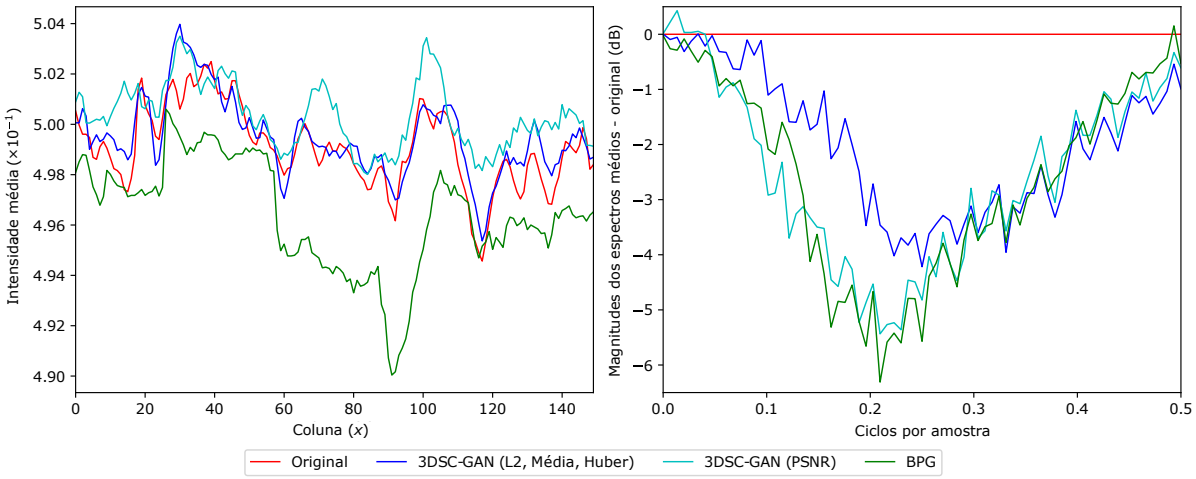
Fonte: Elaborada pelo autor (2022).

4.3.6 Método de ajuste fino: treinamento da rede com o volume a ser comprimido

Visando explorar a capacidade das redes neurais de serem treinadas e testadas no mesmo conjunto de dados, a melhor rede 3DSC-GAN com função composta encontrada nos experimentos anteriores adicionalmente pode ser treinada com ajuste fino. O objetivo é reforçar a rede para comprimir e reconstruir o volume sísmico de interesse. Para isso, propõe-se que cada volume sísmico utilizado nesta dissertação seja particionado em 90% para o conjunto de treinamento e 10% para o de validação. O conjunto de teste é composto pelo volume completo. O procedimento de treinamento é aplicado na rede pré-treinada por 25 épocas. A hipótese é que a rede 3DSC-GAN com função composta conserve mais informações salientes na reconstrução.

A Tabela 16 mostra os resultados da compressão da rede 3DSC-GAN com função composta ajustada para cada volume com uma taxa de bits $r_t = 0{,}1$ bpv. Ela também

Figura 21 – Média de todas as linhas horizontais do interior do quadrado vermelho da Figura 20 para o dado original e suas reconstruções para os métodos BPG, 3DSC-GAN com PSNR como função de distorção, e 3DSC-GAN com funções compostas: perfis de linha médios (esquerda) e espectro médio subtraído do original (direita).



Fonte: Elaborada pelo autor (2022).

apresenta uma comparação quantitativa com a melhor rede 3DSC-GAN com função composta encontrada nos experimentos anteriores e o compressor BPG. Observando os resultados, é notória a evolução do valor de PSNR que a rede obteve ao ser ajustada para o volume de interesse. A rede com ajuste obtém um ganho em praticamente todos os volumes, com exceção dos volumes Penobscot3D e Waihapa3D, onde obteve um valor ligeiramente menor. Considerando os volumes ruidosos, Kerry3D e Poseidon3D, a rede ajustada supera o BPG em termos de PSNR. Esses resultados indicam que o ajuste fino com o próprio volume a ser comprimido tende a beneficiar a compressão de volumes com presença de médias e altas frequências.

A Figura 22 apresenta uma comparação qualitativa entre os métodos 3DSC-GAN com função composta, com e sem ajuste, e BPG. A região reconstruída é a mesma delimitada pelo quadrado vermelho presente na Figura 20, sobre o volume Poseidon3D. A rede com ajuste alcança um valor de PSNR superior aos demais métodos (33,66 dB). Além disso, ela entrega uma reconstrução mais próxima da original, mantendo a variação de estruturas do dado, e aperfeiçoando a reconstrução obtida pelo método 3DSC-GAN com função composta.

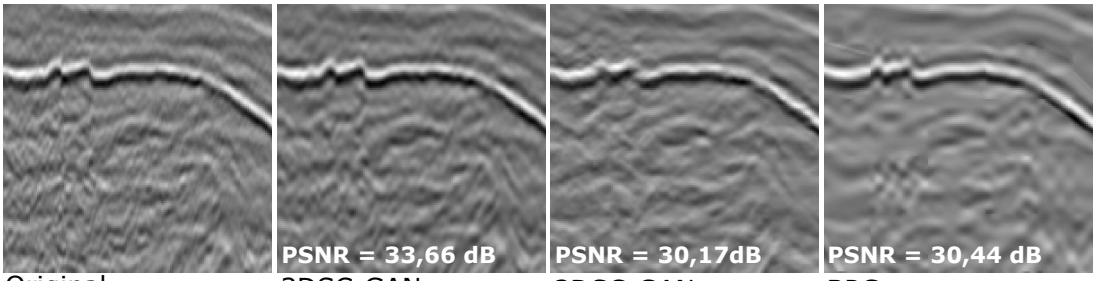
Esse resultado aponta um modo de aplicar o método 3DSC-GAN com função composta na prática. O ajuste de uma rede pré-treinada com um conjunto de dados amplo para comprimir um volume de interesse beneficia a qualidade de reconstrução de forma quantitativa (Tabela 16) e qualitativa (Figura 22). O procedimento de treinamento para o volume Poseidon3D levou 78 minutos, e após a compressão extrema feita pela rede, ele

Tabela 16 – Comparação dos resultados da reconstrução para todos os volumes com uma taxa de bits $r_t = 0{,}1$ bpv. Resultados reportados como PSNR/bpv. Células em cinza destacam o maior valor de PSNR para cada volume

Volumes	Métodos		
	3DSC-GAN (*L2*, *Média*, *Huber*) Com Ajuste	3DSC-GAN (*L2*, *Média*, *Huber*)	BPG
Kahu3D	44,05 / 0,16	41,79 / 0,14	40,37 / 0,11
Kerry3D	33,32 / 0,15	31,19 / 0,19	31,56 / 0,11
Netherlands F3-Block	36,54 / 0,16	36,33 / 0,15	34,37 / 0,10
Opunake3D	49,74 / 0,11	48,45 / 0,10	44,23 / 0,10
Parihaka3D	45,04 / 0,12	44,66 / 0,13	40,15 / 0,10
Penobscot3D	43,26 / 0,14	44,27 / 0,12	40,37 / 0,09
Poseidon3D	30,51 / 0,15	28,69 / 0,17	29,17 / 0,11
Waihapa3D	38,09 / 0,14	39,74 / 0,17	36,32 / 0,11

Fonte: Elaborada pelo autor (2022).

Figura 22 – Comparação da reconstrução de uma fatia 2D (150×150 pixels) do volume Poseidon3D comprimida com uma taxa de bits alvo de 0,1 bpv.



Fonte: Elaborada pelo autor (2022).

passa a necessitar de apenas 6,24 MB para ser armazenado, ao invés dos 1372,37 MB que ele demanda originalmente. Vale ressaltar que os pesos da rede também devem ser salvos para descomprimir o volume posteriormente, e isso pode não ser praticável em volumes pequenos, como os utilizados nesta dissertação. Se adicionamos os pesos da rede aos coeficientes do volume Poseidon3D comprimido, o dado total passa a precisar de 395,73 MB para ser transmitido ou armazenado. Portanto, conclui-se que a abordagem com ajuste fino é indicada para volumes de grande dimensão, situações em que a preservação das estruturas seja crucial, a taxa de bits não precise ser extrema, e que o aumento no número de bits com os pesos da rede seja tolerável.

5 CONCLUSÃO

Neste trabalho são apresentados um método baseado em aprendizado profundo para compressão de dados sísmicos pós-pilha 3D e uma nova função de perda composta para dados multidimensionais. O método proposto, chamado 3DSC-GAN, utiliza GANs para aperfeiçoar a qualidade de reconstrução por meio do treinamento adversário. A integração da rede 3DSC com a GAN foi realizada ao inserir um novo módulo discriminador e utilizar o módulo decodificador como gerador. Além disso, a nova função de perda composta proposta para dados multidimensionais, considera que o dado possui informações distintas e que cada eixo pode ser reduzido de forma a capturar suas diferenças. Foram realizados experimentos e otimizações para cada método de modo a encontrar a melhor combinação de hiperparâmetros para taxas de bits extremas ($r_t < 0{,}5$ bpv).

Resultados experimentais mostraram que o 3DSC foi aperfeiçoado com o treinamento adversário. O método 3DSC-GAN superou as abordagens anteriores baseadas em aprendizado profundo e, comparado com o BPG e JPEG2000, conseguiu atingir valores PSNR superiores para taxas de bits $r_t \leq 0{,}25$ bpv. A análise dos perfis horizontais médios indica que o 3DSC-GAN entrega uma qualidade de reconstrução superior aos demais métodos, mesmo com um valor de PSNR inferior ao obtido pelos compressores de imagens BPG e JPEG2000. Estes resultados indicam que um alto valor de PSNR não reflete uma melhor compressão para dados sísmicos.

Ao substituir a função de distorção PSNR por uma função de distorção mais adequada é notória a evolução alcançada na reconstrução pelo método 3DSC-GAN. Uma busca extensiva foi realizada para encontrar a melhor combinação de funções para compor a função de distorção proposta. As funções $Huber$, $L1$, $L2$, $LogCosh$ e *Média* foram utilizadas. As funções $Huber$ e $LogCosh$ mostraram-se que não são adequadas quando utilizadas na primeira redução, pois não capturam as informações salientes do dado. A combinação com a melhor eficácia foi obtida com ($L2$, *Média, Huber*). O método 3DSC-GAN tem uma melhora quantitativa e qualitativa com a nova função de distorção, se comparado com os obtidos com a função de distorção PSNR. A função de perda multidimensional composta auxilia o método a manter as estruturas do dado sísmico, além de preservar regiões com mais altas frequências, como observado no volume Poseidon3D.

A métrica LPIPS foi utilizada para comparar os métodos, por ser projetada para considerar o julgamento perceptual humano. Entretanto, a LPIPS com a rede AlexNet provê um resultado similar ao obtido pela métrica PSNR. Por outro lado, os resultados utilizando a LPIPS com a rede VGG mostram que a nova função de distorção aperfeiçoa o método 3DSC-GAN e supera o BPG para todos os volumes do conjunto de teste com uma taxa de bits alvo de 0,1 bpv. Vale ressaltar que mesmo não sendo otimizada pelo PSNR, o método 3DSC-GAN com funções compostas obtém um valor PSNR superior em quase todos

os volumes de teste, exceto para o Poseidon3D. Por fim, uma comparação de qualidade de reconstrução foi realizada entre os métodos no volume Posiedon3D, por ser o mais ruidoso e difícil para compressão. O 3DSC-GAN com funções compostas conseguiu entregar uma reconstrução visualmente superior em comparação com a obtida pelos compressores BPG, além de preservar mais as estruturas do dado sísmico, se comparado com o 3DSC-GAN com função de distorção PSNR. Isso indica que funções de distorção mais apropriadas para a natureza do volume de entrada auxiliam a rede a alcançar melhores resultados.

Como trabalhos futuros, pretende-se explorar novas arquiteturas, bem como novas funções de distorção. Outras abordagens generativas podem ser interessantes para o problema para compressão de dados sísmicos, como VAEs. Diferentes funções podem ser incorporadas à função de perda para dados multidimensionais. Assim, funções mais adequadas podem explorar as informações mais relevantes do eixo e auxiliar no aprendizado da rede. Mais uma opção de função de distorção é a própria LPIPS, sendo necessário treiná-la especificamente para dados sísmicos. Outras linhas a serem seguidas estão na utilização de dados sísmicos pré-pilha, visto que também há necessidade de sua compressão, e na aplicação do método 3DSC-GAN no problema de compressão de imagens. Alternativas para aprimorar a estabilidade do 3DSC-GAN em relação à taxa de bits são importantes objetos de pesquisa na área de redes neurais. Neste caso, o objetivo é prover arquiteturas capazes de comprimir o volume de dados sísmicos de forma estável em relação à taxa de bits alvo.

REFERÊNCIAS

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. **Advances in neural information processing systems**, v. 30, 2017.

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

BALLÉ, Johannes; LAPARRA, Valero; SIMONCELLI, Eero P. End-to-end optimized image compression. **arXiv preprint arXiv:1611.01704**, 2016.

BALLÉ, Johannes; MINNEN, David; SINGH, Saurabh; HWANG, Sung Jin; JOHNSTON, Nick. Variational image compression with a scale hyperprior. **arXiv preprint arXiv:1802.01436**, 2018.

BELLARD, Fabrice. **BPG Image format**. 2014. <https://bellard.org/bpg/>. Acessado: 2022-10-07.

BESTAGINI, Paolo; LIPARI, Vincenzo; TUBARO, Stefano. A machine learning approach to facies classification using well logs. In: **Seg technical program expanded abstracts 2017**. [S.l.]: Society of Exploration Geophysicists, 2017. p. 2137–2142.

BRANKOVIC, Milan; GILDIN, Eduardo; GIBSON, Richard L; EVERETT, Mark E. A machine learning-based seismic data compression and interpretation using a novel shifted-matrix decomposition algorithm. **Applied Sciences**, MDPI, v. 11, n. 11, p. 4874, 2021.

BROWN, Alistair R. **Interpretation of three-dimensional seismic data**. [S.l.]: Society of Exploration Geophysicists and American Association of Petroleum, 2011.

CHOWDARY, K Sai Himaja; KALAIYARASI, M; SARAVANAN, S. Gated recurrent unit rnn based non-negative tucker decomposition for satellite image compression. In: IEEE. **2022 Trends in Electrical, Electronics, Computer Engineering Conference (TEECCON)**. [S.l.], 2022. p. 93–96.

DENG, Li. The mnist database of handwritten digit images for machine learning research. **IEEE Signal Processing Magazine**, IEEE, v. 29, n. 6, p. 141–142, 2012.

DONDURUR, Derman. **Acquisition and processing of marine seismic data**. [S.l.]: Elsevier, 2018.

DONG, Chao; LOY, Chen Change; HE, Kaiming; TANG, Xiaoou. Learning a deep convolutional network for image super-resolution. In: SPRINGER. **European Conference on Computer Vision**. [S.l.], 2014. p. 184–199.

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

FELZENSZWALB, Pedro F; GIRSHICK, Ross B; MCALLESTER, David; RAMANAN, Deva. Object detection with discriminatively trained part-based models. **IEEE transactions on pattern analysis and machine intelligence**, IEEE, v. 32, n. 9, p. 1627–1645, 2009.

GOODFELLOW, Ian; POUGET-ABADIE, Jean; MIRZA, Mehdi; XU, Bing; WARDE-FARLEY, David; OZAIR, Sherjil; COURVILLE, Aaron; BENGIO, Yoshua. Generative adversarial nets. **Advances in neural information processing systems**, v. 27, 2014.

GUO, Dandan; LU, Ruiying; CHEN, Bo; ZENG, Zequn; ZHOU, Mingyuan. Matching visual features to hierarchical semantic topics for image paragraph captioning. **International Journal of Computer Vision**, Springer, p. 1–18, 2022.

HUANG, Junxuan; YUAN, Junsong; QIAO, Chunming. Generation for unsupervised domain adaptation: A gan-based approach for object classification with 3d point cloud data. In: IEEE. **ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)**. [S.l.], 2022. p. 3753–3757.

HUBER, Peter J. Robust estimation of a location parameter. In: **Breakthroughs in statistics**. [S.l.]: Springer, 1992. p. 492–518.

JAYASANKAR, Uthayakumar; THIRUMAL, Vengattaraman; PONNURANGAM, Dhavachelvan. A survey on data compression techniques: From the perspective of data quality, coding schemes, data type and applications. **Journal of King Saud University-Computer and Information Sciences**, Elsevier, v. 33, n. 2, p. 119–140, 2021.

JOHNSON, Justin; ALAHI, Alexandre; FEI-FEI, Li. Perceptual losses for real-time style transfer and super-resolution. In: SPRINGER. **European conference on computer vision**. [S.l.], 2016. p. 694–711.

KIELY, Aaron; XU, Mingsen; SONG, Wen-Zhan; HUANG, Renjie; SHIRAZI, Behrooz. Adaptive linear filtering compression on realtime sensor networks. **The Computer Journal**, OUP, v. 53, n. 10, p. 1606–1620, 2010.

KRIZHEVSKY, Alex; SUTSKEVER, Ilya; HINTON, Geoffrey E. Imagenet classification with deep convolutional neural networks. **Advances in neural information processing systems**, v. 25, 2012.

KUZNETSOVA, Alina; ROM, Hassan; ALLDRIN, Neil; UIJLINGS, Jasper; KRASIN, Ivan; PONT-TUSET, Jordi; KAMALI, Shahab; POPOV, Stefan; MALLOCI, Matteo; KOLESNIKOV, Alexander et al. The open images dataset v4. **International Journal of Computer Vision**, Springer, v. 128, n. 7, p. 1956–1981, 2020.

LEE, Meemong; YARLAGADDA, R. Reversible seismic data compression. In: IEEE. **ICASSP'82. IEEE International Conference on Acoustics, Speech, and Signal Processing**. [S.l.], 1982. v. 7, p. 1870–1873.

LELEWER, Debra A; HIRSCHBERG, Daniel S. Data compression. **ACM Computing Surveys (CSUR)**, ACM New York, NY, USA, v. 19, n. 3, p. 261–296, 1987.

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.

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

MAGOTRA, Neeraj; MCCOY, Wes; LIVINGSTON, Frank; STEARNS, S. Lossless data compression using adaptive filters. In: IEEE. **1995 International Conference on Acoustics, Speech, and Signal Processing**. [S.l.], 1995. v. 2, p. 1217–1220.

MENTZER, Fabian; AGUSTSSON, Eirikur; TSCHANNEN, Michael; TIMOFTE, Radu; GOOL, Luc Van. Conditional probability models for deep image compression. In: **Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition**. [S.l.: s.n.], 2018. p. 4394–4402.

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

NIJIM, Yousef W; STEARNS, Samuel D; MIKHAEL, Wasfy B. Lossless compression of seismic signals using differentiation. **IEEE transactions on geoscience and remote sensing**, IEEE, v. 34, n. 1, p. 52–56, 1996.

OORD, Aaron Van; KALCHBRENNER, Nal; KAVUKCUOGLU, Koray. Pixel recurrent neural networks. In: PMLR. **International conference on machine learning**. [S.l.], 2016. p. 1747–1756.

OORD, Aaron Van den; KALCHBRENNER, Nal; ESPEHOLT, Lasse; VINYALS, Oriol; GRAVES, Alex et al. Conditional image generation with pixelcnn decoders. **Advances in neural information processing systems**, v. 29, 2016.

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

RIBEIRO, Kevyn Swhants Santos; SCHIAVON, Ana Paula; NAVARRO, João Paulo; VIEIRA, Marcelo Bernardes; VILLELA, Saulo Moraes; SILVA, Pedro Mário Cruz. Poststack seismic data compression using a generative adversarial network. **IEEE Geoscience and Remote Sensing Letters**, v. 19, p. 1–5, 2022.

SCHIAVON, Ana Paula. **Post-stack seismic data compression with multidimensional deep autoencoders**. Dissertação (Mestrado) — Programa de Pós Graduação em Ciência da Computação, Universidade Federal de Juiz de Fora (UFJF), Juiz de Fora, Brazil, 2020.

SCHIAVON, Ana Paula; NAVARRO, João Paulo; VIEIRA, Marcelo Bernardes; SILVA, Pedro Mário Cruz. 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.

SCHIAVON, Ana Paula; RIBEIRO, Kevyn; NAVARRO, João Paulo; VIEIRA, Marcelo Bernardes; SILVA, Pedro Mário Cruz. 3-d poststack seismic data compression with a deep autoencoder. **IEEE Geoscience and Remote Sensing Letters**, IEEE, 2020.

SEG. **Open data**. 2022. <https://wiki.seg.org/wiki/Open_data>. Acessado: 2022-10-07.

SIMONYAN, Karen; ZISSERMAN, Andrew. Very deep convolutional networks for large-scale image recognition. **arXiv preprint arXiv:1409.1556**, 2014.

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.

STEARNS, Samuel D; TAN, L-Z; MAGOTRA, Neeraj. Lossless compression of waveform data for efficient storage and transmission. **IEEE Transactions on Geoscience and Remote Sensing**, IEEE, v. 31, n. 3, p. 645–654, 1993.

TAO, Ming; TANG, Hao; WU, Fei; JING, Xiao-Yuan; BAO, Bing-Kun; XU, Changsheng. Df-gan: A simple and effective baseline for text-to-image synthesis. In: **Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition**. [S.l.: s.n.], 2022. p. 16515–16525.

TAUBMAN, David S; MARCELLIN, Michael W. Jpeg2000: Standard for interactive imaging. **Proceedings of the IEEE**, IEEE, v. 90, n. 8, p. 1336–1357, 2002.

THEIS, Lucas; SHI, Wenzhe; CUNNINGHAM, Andrew; HUSZÁR, Ferenc. Lossy image compression with compressive autoencoders. **arXiv preprint arXiv:1703.00395**, 2017.

TODERICI, George; O'MALLEY, Sean M; HWANG, Sung Jin; VINCENT, Damien; MINNEN, David; BALUJA, Shumeet; COVELL, Michele; SUKTHANKAR, Rahul. Variable rate image compression with recurrent neural networks. **arXiv preprint arXiv:1511.06085**, 2015.

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

VANKDOTHU, Ramdas; HAMEED, Mohd Abdul. Image compression of brain mri images using an autoencoder and restricted boltzmann machine. **Neuroscience Informatics**, Elsevier, p. 100084, 2022.

VEEKEN, P.C.H.; SILVA, M. Seismic inversion methods and some of their constraints. **First break**, European Association of Geoscientists & Engineers, v. 22, n. 6, 2004.

VERMEER, Peter; BRAGSTAD, Helge; ORR, Caroline. Aspects of seismic data compression. In: **SEG Technical Program Expanded Abstracts 1996**. [S.l.]: Society of Exploration Geophysicists, 1996. p. 2031–2034.

WANG, Ting-Chun; LIU, Ming-Yu; ZHU, Jun-Yan; TAO, Andrew; KAUTZ, Jan; CATANZARO, Bryan. High-resolution image synthesis and semantic manipulation with conditional gans. In: **Proceedings of the IEEE conference on computer vision and pattern recognition**. [S.l.: s.n.], 2018. p. 8798–8807.

WU, Xinming; LIANG, Luming; SHI, Yunzhi; FOMEL, Sergey. Faultseg3d: Using synthetic data sets to train an end-to-end convolutional neural network for 3d seismic fault segmentation. **Geophysics**, Society of Exploration Geophysicists, v. 84, n. 3, p. IM35–IM45, 2019.

WU, Xing; ZHANG, Yafei; LI, Qing; QI, Yangyang; WANG, Jianjia; GUO, Yike. Face aging with pixel-level alignment gan. **Applied Intelligence**, Springer, p. 1–14, 2022.

YANG, Xiaofei; YE, Yunming; LI, Xutao; LAU, Raymond YK; ZHANG, Xiaofeng; HUANG, Xiaohui. Hyperspectral image classification with deep learning models. **IEEE Transactions on Geoscience and Remote Sensing**, IEEE, v. 56, n. 9, p. 5408–5423, 2018.

YILMAZ, Özdoğan. **Seismic data analysis**. [S.l.]: Society of exploration geophysicists Tulsa, 2001. v. 1.

ZHANG, Richard; ISOLA, Phillip; EFROS, Alexei A; SHECHTMAN, Eli; WANG, Oliver. The unreasonable effectiveness of deep features as a perceptual metric. In: **CVPR**. [S.l.: s.n.], 2018.

ZHANG, Zhen-Dong; ALKHALIFAH, Tariq. Regularized elastic full-waveform inversion using deep learning. **Geophysics**, Society of Exploration Geophysicists, v. 84, n. 5, p. R741–R751, 2019.

ZHONG, Sheng-hua; LIU, Yan; LIU, Yang. Bilinear deep learning for image classification. In: **Proceedings of the 19th ACM international conference on Multimedia**. [S.l.: s.n.], 2011. p. 343–352.

ZIZI, Mohammed Outhmane Faouzi; TURQUAIS, Pierre. A dictionary learning method for seismic data compression. **Geophysics**, Society of Exploration Geophysicists, v. 87, n. 2, p. V101–V116, 2022.