

Sistema FIEB



**CENTRO UNIVERSITÁRIO SENAI CIMATEC
PROGRAMA DE PÓS-GRADUAÇÃO EM MODELAGEM
COMPUTACIONAL E TECNOLOGIA INDUSTRIAL**

Luan Rios Campos

**APLICAÇÃO E ANÁLISE DE DEEP LEARNING PARA A
GERAÇÃO DE MODELOS INICIAIS DE VELOCIDADE A
PARTIR DE SISMOGRAMAS**

Salvador
2020

Luan Rios Campos

Aplicação e Análise de Deep Learning para a Geração de Modelos Iniciais de Velocidade a partir de Sismogramas

Dissertação apresentada ao Programa de Pós-Graduação Stricto Sensu do Centro Universitário SENAI CIMATEC como requisito para a obtenção do título de Mestre em Modelagem Computacional e Tecnologia Industrial

Orientador: Prof. Dr. Erick Giovani Sperandio Nascimento

Salvador
2020

Ficha catalográfica elaborada pela Biblioteca do Centro Universitário SENAI CIMATEC

C198a Campos, Luan Rios

Aplicação e análise de deep learning para a geração de modelos iniciais de velocidade a partir de sismogramas / Luan Rios Campos. – Salvador, 2021.

84 f. : il. color.

Orientador: Prof. Dr. Erick Giovani Sperandio Nascimento.

Dissertação (Mestrado em Modelagem Computacional e Tecnologia Industrial) – Programa de Pós-Graduação, Centro Universitário SENAI CIMATEC, Salvador, 2021.

Inclui referências.

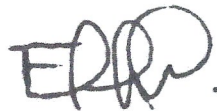
1. Aprendizado profundo. 2. Geofísica. 3. Estimação de modelo de velocidade. 4. Análise de dado sísmico. I. Centro Universitário SENAI CIMATEC. II. Nascimento, Erick Giovani Sperandio. III. Título.

CDD 620.00113

CENTRO UNIVERSITÁRIO SENAI CIMATEC

Mestrado Acadêmico em Modelagem Computacional e Tecnologia Industrial

A Banca Examinadora, constituída pelos professores abaixo listados, aprova a Defesa de Mestrado, intitulada “Aplicação e Análise de Deep Learning para a Geração de Modelos Iniciais de Velocidade a Partir de Sismogramas” apresentada no dia 9 de dezembro de 2020, como parte dos requisitos necessários para a obtenção do Título de Mestre em Modelagem Computacional e Tecnologia Industrial.



Assinado digitalmente por Erick Giovanni Sperandio Nascimento
DN: C=BR, S=Bahia, L=Salvador,
O=SENAI/DR/BA, CN=Erick Giovanni Sperandio Nascimento, E=erick.sperandio@fiqb.org.br
Razão: Eu sou o autor deste documento
Localização: sua localização de assinatura aqui
Data: 2021.01.27 15:37:42-03'00'
Foxit Reader Versão: 10.1.0

Orientador:

Prof. Dr. Erick Giovanni Sperandio Nascimento
SENAI CIMATEC



Membro Interno:

Prof. Dr. Davidson Martins Moreira
SENAI CIMATEC



Prof. Dr. Antônio José da Silva Neto
UERJ

Membro Externo:

Prof. Antônio J. Silva Neto
Professor Titular
Instituto Politécnico-UERJ
Matricula 36.536-1

Dedico este trabalho à clareza de pensamento, empatia social e avanço científico no Brasil mesmo em tempos sombrios como os vividos hoje.

Agradecimentos

Agradeço o apoio imenso que tive de meus pais, Ulemá e Mary, familiares e amigos durante essa jornada, especialmente Flávia, por sempre estar ao meu lado nos momentos de felicidade, mas também nos de dificuldade, sempre apoiando, aconselhando e guiando meu caminho. Agradeço também a Peterson e Rodrigo pela consultoria exclusiva, horas de resenha e aprendizado e total ajuda durante todo esse mestrado, já que sem eles a geofísica continuaria misteriosa e ainda mais complexa, e a meu orientador, Prof. Erick, pela paciência, direcionamento, esclarecimentos na área de *deep learning* e pelas oportunidades de realizar este trabalho, fazer os churras e exibir nossos talentos futebolísticos. Por fim, agradeço a meus *pets* que me deram todo carinho e conforto em toda a jornada: Ralf, Fubá, Nonó, Tuti, Zezo, Pipoca, Mini, Luci e Candinho.

Resumo

O estudo de subsuperfícies é um processo fundamental para a indústria de óleo e gás por fornecer um maior entendimento das estruturas presentes em uma determinada região. Isto pode auxiliar as empresas deste setor industrial em uma identificação mais precisa de regiões com presença de hidrocarbonetos, por exemplo. Entretanto, estudar estas áreas requer o processamento de uma grande quantidade de dados, obtidos durante o processo de aquisição sísmica. Tais dados possuem uma vasta quantidade de informações sobre uma subsuperfície. Uma forma de conduzir este estudo é através da análise dos parâmetros de velocidade, representados por meio do modelo de velocidade da subsuperfície, que estão contidos nos dados sísmicos. Gerar um modelo de velocidade satisfatório para representar uma subsuperfície é desafiante, já que este processo se caracteriza por ser não-linear e inverso, i.e., o modelo de velocidades é gerado a partir do dado sísmico. Neste sentido, aplica-se uma modelagem sísmica, que repete o processo de aquisição ao simular a propagação e captura da reflexão de ondas sísmicas em um meio, com o intuito de utilizar técnicas computacionais, como a inversão de forma de onda completa, que possam encontrar com mais exatidão os parâmetros de velocidade de determinada subsuperfície. Este trabalho tem como objetivo avaliar, primeiramente, a utilização de técnicas de *deep learning*, especialmente redes totalmente convolucionais, na geração de modelos de velocidades iniciais. O primeiro experimento avalia os modelos estimados por uma U-Net e os aplica ao método de inversão da forma de onda completa como forma de obter-se uma melhora em termos de resolução destes modelos. Em seguida, avalia-se os efeitos que alterações nos parâmetros da modelagem sísmica possuem nos resultados da U-Net. As últimas etapas deste trabalho envolvem a otimização de hiperparâmetros da U-Net, como função de ativação e otimizadores, aplicação de redução de dimensionalidades para treinamento da U-Net utilizando dados reduzidos e, por fim, utilização do modelo de *deep learning HyperColumn*. A análise dos modelos estimados é feita de forma qualitativa, por meio da comparação de um dos modelos estimados com seu modelo de referência, e quantitativa, ao comparar amplamente todos os modelos de velocidade por meio de métricas estatísticas, o que auxilia na tomada de decisão de possíveis pontos de melhoria. Os resultados obtidos foram satisfatórios tanto no sentido interdisciplinar, quanto no sentido científico, ao considerar-se o avanço obtido a cada experimento para a estimação de modelos de velocidade, apresentado diferentes e novas abordagens para utilização dos dados provenientes da modelagem sísmica e para resolução do problema de inversão sísmica.

Palavras-chaves:

Aprendizado profundo, Geofísica, Estimação de modelo de velocidade, Análise de dado sísmico, Redes totalmente convolucionais, U-Net, HyperColumn, Redução de Dimensionalidade.

Abstract

The process of studying a subsurface is fundamental for the oil and gas industry as it gives a better understanding over the underlying structures of a certain region. This can help this industry in identifying in a more precise way regions that contains hydrocarbons, for example. Nonetheless, to study these areas require processing a large amount of data, which is obtained during the seismic acquisition process. Such data have a vast quantity of information of a subsurface. One way of conducting this study is through the analysis of the velocity parameters, represented by the velocity model of a subsurface, that are within the seismic data. The generation of a satisfactory velocity model is a challenging task as it is a non-linear and inverse problem, i.e., the velocity model is created from the seismic data. In this sense, we can apply a seismic modeling, which repeats the process of acquisition by simulating the propagation and reflecting of the wave in a medium, with the objective of using computational tools, such as the full waveform inversion, to help us finding with a higher accuracy the velocity parameters of a given subsurface. This work has the goal to firstly evaluate the use of deep learning techniques, especially fully convolutional networks, to generate initial velocity models. The first experiment evaluate the velocity models generated by a U-Net and apply them to the full waveform inversion method in an attempt of improving the resolution of the models. In the next experiment we evaluate the effects that changes in the modeling parameters can cause to the results of the U-Net. The last steps of this work involves the U-Net's hyperparameter optimization, such as activation functions and optimizers, use of dimensionality reduction techniques to train the U-Net with reduced seismic data and, finally, the use of the HyperColumn deep learning model. The analysis of the estimated models are made qualitative, by comparing one of the estimated models with its respective ground-truth, and quantitatively, by comparing in a broad way all models using statistical metrics, which aids us in the decision taking of further improvement points. The results were satisfactory from both an interdisciplinary and scientific view when we consider the advances obtained in each experiment for estimating a velocity model, presenting different and novel approaches to utilize data generated by seismic modeling and for solving the seismic inversion problem.

Keywords:

Deep Learning, Geophysics, Velocity Model Estimation, Seismic Data Analysis, Fully Convolutional Networks, U-Net, HyperColumn, Dimensionality Reduction.

Lista de Figuras

Figura 1 – Processo de aquisição sísmica marítimo para uma única fonte. Fonte: (COMMITTEE, 2017)	16
Figura 2 – Exemplos de modelos de velocidades (a) de fundo/ <i>background</i> e (b) de alta resolução	17
Figura 3 – Modelo de neurônio onde o campo induzido (v_k) é gerado pela soma ponderada das entradas, com $x_0 = 1$ e w_0 representando o fator <i>bias</i> e a saída y_k sendo gerada ao aplicar uma função de ativação ϕ em v_k	24
Figura 4 – Modelo de um Perceptron Multicamadas com 4 camadas: 1 camada de entrada, 2 camadas escondidas e 1 camada de saída.	25
Figura 5 – ReLU Activation Function Plot	29
Figura 6 – Leaky ReLU Activation Function Plot	29
Figura 7 – ELU Activation Function Plot	30
Figura 8 – PReLU Activation Function Plot	31
Figura 9 – Ilustração do processo de otimização em busca do mínimo global	32
Figura 10 – Exemplo de uma imagem de segmentação semântica. Fonte: (CORDTS et al., 2016)	35
Figura 11 – Módulos de <i>encoder</i> e <i>decoder</i> de um Autoencoder (FRANCOIS, 2017)	38
Figura 12 – Representação de um variational autoencoder (FRANCOIS, 2017)	38
Figura 13 – Arranjo de fontes e receptores durante uma aquisição sísmica de poço.	41
Figura 14 – Fluxograma do processo de criação dos modelos de velocidades.	43
Figura 15 – Um modelo de velocidades sintético contendo 10 camadas, ondulações, inclinações e estruturas de falha	44
Figura 16 – Exemplo de dado sísmico sintético gerado por meio da modelagem sísmica de uma única fonte sísmica. Fonte: (SANTOS; PESTANA, 2016)	44
Figura 17 – Fluxograma do processo de criação dos dados sísmicos para cada modelo de velocidades.	45
Figura 18 – Esquema da U-Net utilizada nesta dissertação. n_s representa o número de canais na camada de entrada, formando assim uma entrada de $1500 \times 150_s$. Os blocos vermelhos além do primeiro representam as demais camadas e os valores abaixo deles representam o número de canais nestas camadas.	46
Figura 19 – Representação de um <i>hypercolumn</i> . O <i>hypercolumn</i> de um pixel é o vetor de ativações de todas as unidades acima deste pixel. Fonte: (HARIHARAN et al., 2015).	47

Figura 20 – Concatenação das camadas de <i>upsample</i> e geração saída do modelo HyperColumn. As camadas em vermelho representam operações de convolução. Ao contrário da U-Net, as camadas em vermelho não contribuem para a produção da saída da rede e as saídas das camadas de <i>upsample</i> não servem como entrada para as camadas posteriores.	48
Figura 21 – Monitoramento e comparação do decaimento da função de erro para a etapa de treinamento e validação a cada época	52
Figura 22 – Saída estimada durante as épocas de treinamento, respectivamente, a) 1, b) 75, c) 151 e d) 199	53
Figura 23 – Resultado da estimação de dois modelos de velocidades em comparação com suas versões verdadeiras existentes no conjunto de teste. a) O primeiro modelo contém camadas com ondulações e inclinações, b) o segundo modelo possui camadas onduladas e com falhas	53
Figura 24 – Comparação do (a) modelo verdadeiro, (b) modelo estimado, (c) modelo após uso do método FWI e (d) taxa de decrescimento do erro RMS a cada iteração do FWI	53
Figura 25 – Ilustração do decaimento da função de erro a cada época de treinamento e validação da U-Net para os dados sísmicos gerados com uma fonte central (a), 10 fontes (b), 25 fontes e 4Hz (c), 50 fontes (d), 25 fontes e 8Hz (e) e 25 fontes e 16Hz (f).	56
Figura 26 – Comparação qualitativa entre o modelo verdadeiro (a) e os modelos estimados por diferentes instâncias da U-Net, as quais possuem a mesma arquitetura, mas foram treinadas com dados sísmicos gerados a partir de diferentes configurações. Primeiro, foi fixado o valor de $f_{peak} = 4\text{Hz}$ e variou-se as fontes em fonte posicionada no centro do modelo (b), 10 fontes (c), 25 fontes (d) e 50 fontes (e). Em seguida o número de fontes foi fixado em 25 e f_{peak} foi variado em 8Hz (f) e 16Hz (g).	57
Figura 27 – Avaliação do tempo de treinamento (a) e avaliação estatística de cada experimento considerando as métricas MSE (b), MAE (c) e correlação de Pearson, coeficiente de determinação R^2 e fator de 2 (d).	58
Figura 28 – Comparação do decaimento da função de erro época a época durante treinamento e validação da U-Net -Net ao modificar o otimizador e a função ativação para RMSprop + ReLU (a), RMSprop + Leaky ReLU (b), RMSprop + ELU (c), RMSprop + PReLU (d), Adam + ReLU (e), Adam + Leaky ReLU (f), Adam + ELU (g), Adam + PReLU (h), Adamax + ReLU (i), Adamax + Leaky ReLU (j), Adamax + ELU (k) e Adamax + PReLU (l).	60

Figura 29 – Comparação do modelo verdadeiro (a) com os modelos de velocidades estimados pela U-Net ao modificar o otimizador e a função ativação para RMSprop + ReLU (b), RMSprop + Leaky ReLU (c), RMSprop + ELU (d), RMSprop + PReLU (e), Adam + ReLU (f), Adam + Leaky ReLU (g), Adam + ELU (h), Adam + PReLU (i), Adamax + ReLU (j), Adamax + Leaky ReLU (k), Adamax + ELU (l) e Adamax + PReLU (m).	61
Figura 30 – Avaliação estatística das combinações das função de ativações e otimizadores para o (a) MSE, (b) MAE, (c) correlação de Pearson, (d) coeficiente de determinação R^2 e (e) fator de 2	63
Figura 31 – Resultado da redução de dimensionalidade de um tiro de um dado sísmico com <i>autoencoder</i> (tiro à esquerda) e PCA (tiro à direita).	66
Figura 32 – Tiros originais e reconstruídos de um dado sísmico modelado com $f_{peak} = 8\text{Hz}$ e fontes posicionadas no (a) início, (b) centro e (c) final, (d) tiro de um dado sísmico modelado com $f_{peak} = 4\text{Hz}$ e fonte posicionada no centro e (e) tiro de um dado sísmico modelado com $f_{peak} = 16\text{Hz}$ e fonte posicionada no centro.	67
Figura 33 – Resultado dos modelos estimados pela U-Net utilizando os dados sísmicos reduzidos pelo <i>autoencoder</i> (a) e PCA (b)	68
Figura 34 – (a) Avaliação do tempo de treinamento e avaliação estatística de cada experimento do (b) MSE e MAE, (c) correlação de Pearson, coeficiente de determinação R^2 e fator de 2 utilizando dados reduzidos	68
Figura 35 – Representação das camadas utilizadas no modelo de HyperColumn.	69
Figura 36 – Monitoramento e comparação do decaimento da função de erro para a etapa de treinamento e validação a cada época do HyperColumn	70
Figura 37 – modelos de velocidades sintéticos estimados utilizando a rede HyperColumn	71
Figura 38 – Avanço do processo de treinamento da rede <i>HyperColumn</i> do (a) modelo de velocidades de referência, observando-se o resultado na (b) época 90, (c) época 150 e (d) última época.	73

Lista de Tabelas

Tabela 1 – Métricas de avaliação para a U-Net	52
Tabela 2 – Métricas de avaliação para a HyperColumn e duração do treinamento em horas em comparação às demais técnicas	72

Lista de abreviaturas e siglas

Adam	<i>Adaptive Moment Estimation</i>
AE	<i>Autoencoder</i>
AG	<i>Algoritmo Genético</i>
CIG	<i>Common-Image Gather</i>
CNN	<i>Convolutional Neural Network</i>
CRF	<i>Conditional Random Field</i>
ELU	<i>Exponential Linear Unit</i>
FCN	<i>Fully Convolutional Network</i>
FWI	<i>Full Waveform Inversion</i>
GPU	<i>Graphical Processing Unit</i>
Leaky ReLU	<i>Leaky Rectified Linear Unit</i>
MAE	<i>Mean Absolute Error</i>
MSE	<i>Mean Squared Error</i>
PCA	<i>Principal Component Analysis</i>
PReLU	<i>Parametric Rectified Linear Unit</i>
PSO	<i>Particle Swarm Optimization</i>
RAM	<i>Random Access Memory</i>
ReLU	<i>Rectified Linear Unit</i>
RTM	<i>Reverse-Time Migration</i>
SGD	<i>Stochastic Gradient Descent</i>
VAE	<i>Variational Autoencoder</i>

Lista de símbolos

d_{est}	Dado sísmico estimado por um método de inversão sísmica
d_{obs}	Dado sísmico obtido durante a aquisição sísmica
fac_2	Fator de 2
f_{peak}	Frequência de pico
f_{max}	Frequência máxima
$J(\theta)$	Função objetivo
m_i	Média móvel do gradiente utilizada no otimizador Adam
m_v	Modelo de velocidade
R^2	Coefficiente de determinação
r	Coefficiente de correlação de Pearson
v_i	Média móvel do gradiente ao quadrado utilizada no otimizador Adam
α	Constante utilizada em funções de ativação como <i>Leaky</i> ReLU, ELU e <i>Parametric</i> ReLU
β	Taxa de decaimento dos otimizadores Adam e Adamax
δ	Valor de erro da rede neural
ϵ	Valor que evita divisões por zero
η	Taxa de aprendizado
θ	Parâmetros da rede neural
ϕ	Função de ativação

Sumário

1	INTRODUÇÃO	15
1.1	Objetivos	19
1.1.1	Objetivo Geral	19
1.1.2	Objetivos Específicos	19
1.2	Organização	19
2	REVISÃO DE LITERATURA	21
2.1	Sísmica	21
2.1.1	Modelagem Sísmica Acústica	21
2.1.2	Método de Inversão de Forma de Onda	22
2.2	Redes Neurais Artificiais	23
2.2.1	Modelo do Neurônio	23
2.2.2	Redes Multicamadas	24
2.2.3	Funções de Ativação	27
2.2.3.1	ReLU	28
2.2.3.2	Leaky ReLU	29
2.2.3.3	ELU	30
2.2.3.4	PReLU	30
2.2.4	Otimizadores	31
2.2.4.1	RMSprop	32
2.2.4.2	Adam e Adamax	33
2.2.5	Redes Convolucionais	34
2.2.5.1	Redes totalmente convolucionais	35
2.3	Redução de Dimensionalidade	36
2.3.1	Análise de Componentes Principais (PCA)	36
2.3.2	Autoencoder (AE)	37
2.3.3	Variational Autoencoder (VAE)	37

2.4	Estado da Arte	38
3	METODOLOGIA	42
3.1	Modelos de velocidades e Sismograma	42
3.2	Modelos <i>Deep Learning</i>	45
3.3	Formas de Análise	47
4	RESULTADOS E DISCUSSÕES	51
4.1	Variação nos Parâmetros de Modelagem Sísmica	54
4.2	Otimização de Hiperparâmetros	59
4.3	Redução de Dimensionalidade	63
4.4	Utilização de HyperColumn	68
5	CONSIDERAÇÕES FINAIS	75
	REFERÊNCIAS	78
	Glossário	83

1 Introdução

A exploração de subsuperfícies da terra é um processo que envolve altos custos para a indústria de óleo e gás, os quais vão desde o processo de estudo de uma área, até o processo de perfuração de um poço.

O primeiro passo para o estudo da subsuperfície é adquirir informações sísmicas da área. Esta é uma etapa essencial para a exploração de hidrocarbonetos e, portanto, é importante saber como os dados sísmicos são adquiridos. A aquisição envolve levantamentos sísmicos que podem ser realizados ao longo de linhas para produzir um perfil vertical (levantamento 2D) ou sobre uma área a ser explorada para gerar um volume de dados da subsuperfície (levantamento 3D).

Em um experimento de aquisição de dados sísmicos, a energia acústica é normalmente gerada por fontes controladas, como cargas explosivas ou por grandes caminhões vibroseis, no caso da sísmica terrestre, e por cilindros de ar comprimido, conhecidos como *air gun*, no caso da sísmica marinha. Dessa forma, a energia se espalha pela subsuperfície como uma frente de onda esférica que se propaga em todas as direções. As interfaces entre os diferentes tipos de rochas refletem e transmitem essa frente de onda. Os sinais refletidos retornam à superfície onde são detectados pelos receptores implantados ao longo de uma geometria predefinida. Estes receptores são chamados geofones para aquisições terrestres e hidrofones para aquisições marinhas. Eles podem gravar um ou todos os três componentes do vetor velocidade de partículas ou campo de pressão no caso de um hidrofone. Nesta dissertação, no entanto, trataremos apenas com o campo de pressão da ondas compressionais (ondas P).

O processo de aquisição sísmica, ou geração do dado sísmico, em subsuperfícies marítimas é ilustrado na Figura 1. Neste processo, uma série de dispositivos receptores (hidrofones) são distribuídos em linha e associados a uma embarcação. A embarcação é munida de cilindros de ar comprimido (*air gun*) que provocam perturbações em forma de onda no meio através do disparo de ar. Estas ondas se propagam pelas estruturas da subsuperfície e, quando refletidas, têm sua assinatura capturada pelos hidrofones. Assim, o sismograma detém inúmeras informações sobre as estruturas que residem abaixo de onde o processo de aquisição ocorreu.

Nesta dissertação, são consideradas configurações simples de aquisição sísmica 2D. No levantamento sísmico de reflexão 2D, tanto as fontes quanto os receptores são movidos ao longo de uma linha reta, enquanto na aquisição sísmica 3D, uma matriz densa de receptores se move em função da fonte.

O bom entendimento dos dados sísmicos (sismogramas) de uma subsuperfície pode

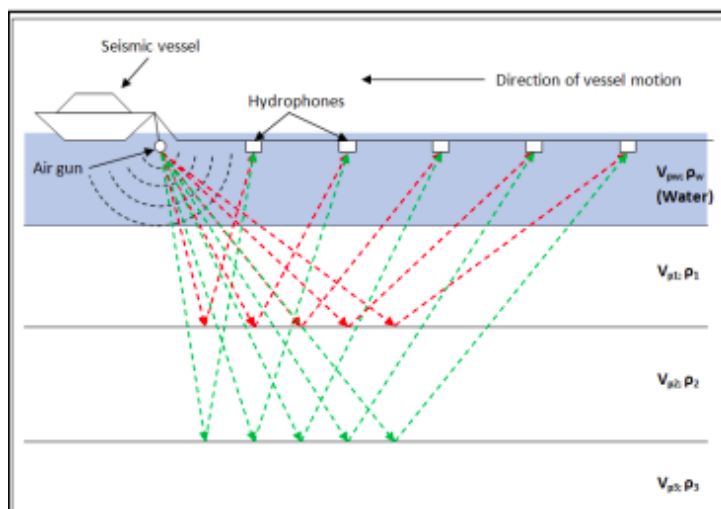


Figura 1 – Processo de aquisição sísmica marítimo para uma única fonte. Fonte: (COMMITTEE, 2017)

auxiliar as empresas de óleo e gás a definirem com maior acurácia os locais de perfuração que podem conter, por exemplo, petróleo, diminuindo o risco de erro e, conseqüentemente, o custo deste processo.

Dois principais problemas podem ser elencados ao tratar-se do entendimento de dados sísmicos: 1) estes dados, em sua forma natural, apesar de possuírem inúmeras informações, não as possuem de forma detalhada e direta; 2) eles são, considerando-se situações do mundo real, muitos grandes, no sentido da quantidade de informações que possuem, complexos para serem analisados manualmente por humanos.

Neste contexto, o processo de modelagem computacional sísmica tenta replicar o processo de aquisição dos dados sísmicos ao simular as fontes de propagação de onda e seus receptores. O intuito desta modelagem é conseguir analisar o dado por meio de imageamento sísmico, aplicando-se técnicas como *Reverse Time Migration* (RTM) ou *Full-Waveform Inversion* (FWI), de forma a reduzir os problemas mencionados anteriormente. A técnica de RTM não é apresentada de forma completa nesta dissertação, apenas citada em algumas situações. Em contrapartida, maiores detalhes acerca do FWI poderão ser vistos na Seção 2.1.2 - *Método de Inversão de Forma de Onda*. Tais métodos necessitam de um modelo de velocidades para que possam operar.

Um modelo de velocidades consiste no valor da velocidade com que a onda se propaga nas rochas da subsuperfície. De posse dos valores de velocidade nas rochas, obtém-se uma representação das estruturas existentes no substrato rochoso. Isto ocorre porque a velocidade das ondas sísmicas varia à medida que o material rochoso em subsuperfície muda, i.e., a velocidade da onda muda ao propagar-se na água, folhelhos, arenitos ou corpos de sais, sendo possível, assim, representar a subsuperfície.

No caso da RTM, necessita-se de um modelo de alta resolução, isto é, com detalhes das estruturas bem definidas, já o intuito da FWI é, a partir de um modelo inicial, com menor resolução, iterativamente refinar e produzir um modelo de alta resolução, de modo a resolver um problema não-linear de inversão (VIRIEUX; OPERTO, 2009). Os modelos de velocidades podem ser caracterizados em modelos de fundo (*background*) ou de alta resolução. Modelos de fundo (Figura 2(a)) carecem de detalhes com relação às estruturas presentes em uma subsuperfície, mesmo que forneça uma suposição inicial de suas velocidades. Já os de alta resolução (Figura 2(b)) são muito mais relacionados ao modelo de velocidades verdadeiro por permitir a identificação clara das estruturas presentes na subsuperfície e do seu formato através dos valores de velocidade.

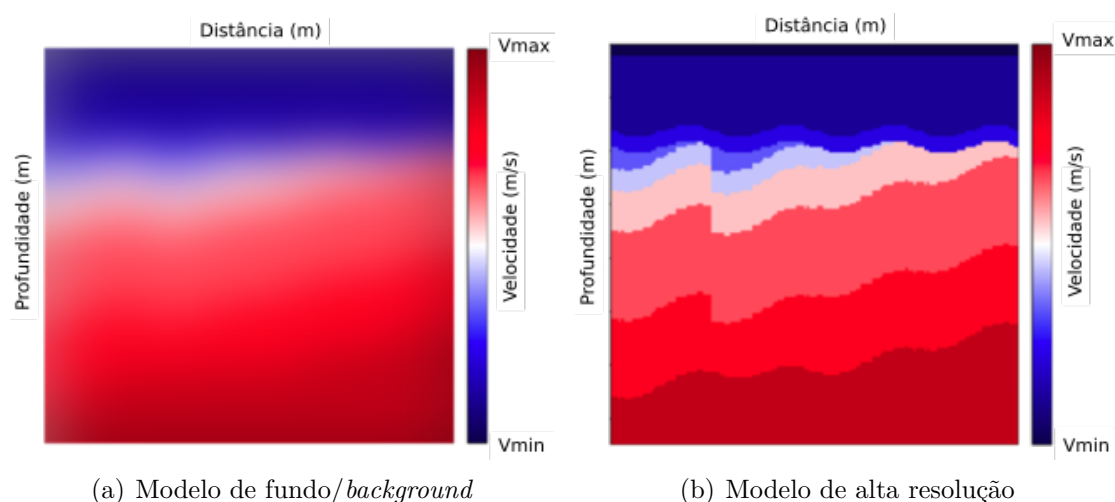


Figura 2 – Exemplos de modelos de velocidades (a) de fundo/*background* e (b) de alta resolução

Produzir modelos de velocidades de alta resolução é um desafio na comunidade geofísica, e, por conta disso, diversas técnicas têm sido desenvolvidas por pesquisadores da área. Diferentes abordagens têm surgido na literatura geofísica para lidar, primeiramente, com o problema de geração de modelos de velocidades iniciais.

Tal problema é de extrema importância, já que a qualidade do modelo inicial é fundamental para técnicas de inversão da forma de onda, na qual é necessário um modelo de partida para que se obtenha uma boa convergência e, conseqüentemente, uma boa imagem sísmica final. Oferecer um ótimo modelo inicial de velocidades para a FWI, por exemplo, pode reduzir o poder computacional necessário para aplicar o método, já que ele poderá ser finalizado em menos iterações, e aumentar seu sucesso por evitar mínimos locais, garantindo que um modelo de velocidades de alta resolução seja produzido.

Logo, o modelo de velocidades de uma subsuperfície põe-se como uma das ferramentas cruciais para a análise de sismograma e tomada de decisão, já que uma boa versão inicial dele pode melhorar os resultados de técnicas já conhecidas e aplicadas na

indústria. Recentemente, pesquisadores de geofísica têm, além das técnicas convencionais, experimentado o uso de técnicas de inteligência artificial para resolver o problema da estimação de modelos de velocidades.

A Inteligência Artificial, especialmente seu ramo de aprendizado profundo (*deep learning*), tem ganhado cada vez mais espaço com o crescimento da computação de alto desempenho, principalmente ao crescimento do poder de processamento de *graphics processing units* (GPUs).

Assim, o uso de aprendizado profundo pode contribuir para a indústria de óleo e gás ao apresentar possíveis alternativas aos métodos convencionais para inversão sísmica. O estudo da aplicação de *deep learning* para resolver este problema em específico da geofísica pode trazer ganhos como redução do custo de simulação, armazenamento e processamento de dados sísmicos. Além disso, uma vez que o modelo esteja treinado, o processo de inversão sísmica a partir de dados cujo modelos não são conhecidos pode ser acelerado. Por fim, o uso destas técnicas pode, ainda, trazer aos geofísicos um ponto de partida mais refinado, quando comparado com as técnicas comumente utilizadas.

A questão que norteia esta pesquisa é: como a aplicação de técnicas de *deep learning* pode auxiliar no desenvolvimento de modelos iniciais de velocidade de uma subsuperfície? Em complemento, questiona-se também se: 1) é possível estimar modelos iniciais para serem utilizados em outras técnicas como, por exemplo, o FWI?; 2) qual a influência dos parâmetros de modelagem, como o número de fontes, no *aprendizado* do modelo de *deep learning* em comparação a outros métodos de inversão sísmica?

A partir daí, surgem duas hipóteses:

1. os métodos de *deep learning* conseguem estimar modelos iniciais de velocidade satisfatórios que podem, inclusive, ser utilizados sem ônus no método FWI;
2. um aumento no número de fontes utilizados durante a modelagem não significa melhoria na estimação do modelo pela técnica de *deep learning*.

Em linhas gerais, o intuito deste trabalho é analisar o uso da U-Net, um tipo de rede totalmente convolucional (*fully convolutional network* - FCN), para a geração de modelos iniciais de velocidade sob a motivação de poder promover um maior entendimento sobre o uso de *deep learning* em dados sísmicos. Espera-se que com esta pesquisa se possa, futuramente, guiar pesquisadores no desenvolvimento de técnicas robustas para a exploração de subsuperfícies, auxiliando tanto geofísicos, quanto empresas neste processo.

As limitações desta pesquisa se dão, principalmente, devido à demanda de recursos computacionais dos dados sísmicos e dos métodos de *deep learning*. No primeiro caso, há

uma grande demanda em termos de memória de acesso aleatório, do inglês *random access memory* (RAM), para o armazenamento dos dados sísmicos necessário para treinamento do modelo de *deep learning*. Conforme o tamanho da subsuperfície na qual a modelagem está sendo feita aumenta, tende-se, também, a aumentar o tamanho computacional do dado sísmico resultante, i.e., a quantidade de *bytes* necessária para armazená-lo em um disco.

No que diz respeito às técnicas de *deep learning*, podemos citar dois pontos limitantes: 1) a complexidade da rede, i.e., o número de parâmetros necessários para treiná-la e 2) a dependência que existe quanto ao tamanho do dado de entrada.

1.1 Objetivos

1.1.1 Objetivo Geral

O objetivo geral deste trabalho é avaliar a viabilidade de utilização de uma rede totalmente convolucional U-Net na estimação de modelos iniciais de velocidade de uma subsuperfície marítima.

1.1.2 Objetivos Específicos

Em termos de objetivos específicos, elencam-se:

1. reduzir o custo computacional necessário para geração de modelos iniciais de velocidade;
2. avaliar os modelos desenvolvidos em técnicas de geofísica, como o FWI;
3. avaliar como os parâmetros de modelagem afetam o processo de treinamento do modelo de *deep learning*;
4. otimizar o modelo de *deep learning* para geração de modelos de velocidades cada vez melhores.

1.2 Organização

Este trabalho está disposto de acordo com os seguintes capítulos:

- **Capítulo 1 - Introdução:** Apresenta considerações iniciais acerca do problema, sua definição, objetivos e importância. Além de indicar a motivação, limites, questões e hipóteses deste trabalho e como ele está estruturado, contextualizando, assim, o âmbito geral da pesquisa.

- **Capítulo 2 - Revisão de Literatura:** Colabora com a elucidação de alguns conceitos de geofísica, como aquisição sísmica e o método de inversão de forma de onda, e de redes neurais artificiais, como a concepção de um neurônio artificial, funções de ativação, otimizadores, o algoritmo de *backpropagation*, redes convolucionais e suas variações, aplicadas a este trabalho. Além disso, apresenta algumas técnicas que compõem o estado da arte para solucionar o problema de estimação de modelos de velocidades.
- **Capítulo 3 - Metodologia:** Apresenta os aspectos metodológicos utilizados nesta dissertação, como: estratégia para geração de modelos de velocidades sintéticos, abordagem utilizada para treinamento do modelo de *deep learning* e formas de avaliação dos experimentos.
- **Capítulo 4 - Resultados e Discussões:** Apresenta o experimento e resultados iniciais obtidos com a aplicação de *deep learning* para a estimação de modelos iniciais de velocidade.
- **Capítulo 5 - Considerações Finais:** Apresenta as conclusões, contribuições e sugestões para atividades de pesquisa a serem desenvolvidas no futuro.

2 Revisão de Literatura

2.1 Sísmica

2.1.1 Modelagem Sísmica Acústica

A modelagem sísmica simula o processo de propagação das ondas em uma subsuperfície conforme ocorre durante a aquisição sísmica real. Esta simulação é realizada para que os pesquisadores possam obter avanços nos processos que auxiliam o entendimento de tais áreas. Esta subseção é dedicada a apresentar algumas das equações consideradas para realizar uma modelagem sísmica por diferenças finitas a partir da equação da onda acústica.

$$\begin{cases} \frac{1}{v(x)^2} \frac{\partial^2 P_s(x,t)}{\partial t^2} - \nabla^2 P_s(x,t) = s(x,t) \\ P_s(., t=0) = 0 \\ \left. \frac{\partial P_s(x,t)}{\partial t} \right|_{t=0} = 0 \end{cases} \quad (2.1)$$

$$\nabla^2 = \frac{\partial^2}{\partial d_x^2} + \frac{\partial^2}{\partial d_z^2} \quad (2.2)$$

A equação da onda acústica (ALFORD; KELLY; BOORE, 1974) (BAYSAL; KOSLOFF; SHERWOOD, 1983) e suas condições iniciais são apresentadas na Equação 2.1, onde $x = (d_x, d_z)$ é a posição na subsuperfície considerando um modelo 2D, $v(x)$ é a velocidade na posição considerada, $P_s(x,t)$ é o campo de onda da fonte e $s(x,t)$ denota a fonte sísmica da onda acústica. O operador Laplaciano (Equação 2.2), representa as derivadas espaciais de segunda ordem para o caso bi-dimensional.

Uma forma de realizar a modelagem sísmica é por meio do método de diferenças finitas (BAYSAL; KOSLOFF; SHERWOOD, 1983), o qual oferece uma "implementação simples e fácil" (SANTOS, 2013) através da série de Taylor, consequentemente ocasionando em uma discretização das equações. Tanto a Equação 2.1, quanto a Equação 2.2 podem ser expandidas pela série de Taylor. Entretanto, algumas condições devem ser respeitadas para que a dispersão e instabilidade numérica que podem surgir durante o processo de discretização de uma Equação de tempo contínuo seja evitada (SANTOS, 2013).

$$\Delta t \leq \frac{1}{\max(v) \times \sqrt{\frac{1}{\Delta d_x^2} + \frac{1}{\Delta d_z^2}}} \quad (2.3)$$

A Equação 2.3 (SANTOS, 2013) denota as condições necessárias para que se evite a instabilidade numérica em um modelo 2D, no qual Δt é o intervalo de tempo

de amostragem, $max(v)$ é a velocidade máxima do modelo, Δd_x e Δd_z são os intervalos espaciais de amostragem respectivamente nos eixos x e z .

$$f_{max} \leq \frac{1}{F} \frac{min(v)}{max(\Delta d_x, \Delta d_z)} \quad (2.4)$$

Já a Equação 2.4 (SANTOS, 2013) exhibe as condições necessárias para que o problema da dispersão numérica seja evitado em um modelo bi-dimensional. Nesta Equação, f_{max} é o valor máximo de frequência permitido para que não haja dispersão considerando algumas das características do modelo em questão, i.e., o valor máximo entre seus intervalos de amostragem [$max(\Delta d_x, \Delta d_z)$] e seu valor mínimo de velocidade [$min(v)$]. O parâmetro F é um valor constante e é definido de acordo com a ordem aplicada na série de Taylor. Quanto maior a ordem, menor o valor de F .

$$f_{peak} = \frac{f_{max}}{2,3} \quad (2.5)$$

Outro parâmetro importante definido na modelagem sísmica é a frequência de pico (f_{peak}). Este parâmetro é definido na Equação 2.5 como sendo, aproximadamente, metade da frequência máxima e representa o ponto de maior amplitude no espectro de frequência.

Além de dispersão e instabilidade numérica, a discretização da onda acústica pode ocasionar em problemas nas regiões de contorno da malha numérica. Alguns métodos podem ser aplicados para aliviar tais problemas, são eles: a condição de contorno não reflexiva (CERJAN et al., 1985), por absorção (SOCHACKI et al., 1987; HIGDON, 1991) ou por aplicação de uma camada perfeitamente combinada (*perfectly matched layer* - PML) para aproximação da condição de borda (PAN; ABUBAKAR; HABASHY, 2012).

2.1.2 Método de Inversão de Forma de Onda

Segundo Santos (2013), o objetivo do método de inversão de forma de onda, *full waveform inversion* (FWI) (TARANTOLA, 1984), é conseguir gerar um modelo de velocidades de alta resolução por meio da equação completa da onda de modo que a propagação das ondas, simulada durante a modelagem sísmica, gere um dado sintético bastante próximo do dado obtido durante o processo de aquisição sísmica, incluindo a amplitude e fase do sinal sísmico e seus tempos de trânsito.

$$E(m_v) = \frac{1}{2} \|d_{est} - d_{obs}\|^2 \quad (2.6)$$

Calcular o operador de inversão em sua total exatidão é uma tarefa bastante complexa e, muitas vezes, impossível de ser feita. Assim, a FWI procura uma aproximação

para tal operador ao otimizar uma função de mínimos quadrados (Equação 2.6) (SANTOS, 2013) (SANTOS; PESTANA, 2016). Nesta equação, d_{obs} representa o dado observado durante o processo de aquisição, d_{est} o dado sintético produzido, $E(m_v)$ é a medida de erro do modelo de velocidades (m_v) a ser encontrado e $\|\cdot\|^2$ representa a norma l_2 . Desta forma, todas as informações presentes no sismograma são consideradas ao parametrizar-se d_{est} .

A FWI procura resolver um problema não-linear (VIRIEUX; OPERTO, 2009) tendo como principal característica a utilização de um número reduzido de aproximações. Entretanto, Santos (2013) chama a atenção para dois pontos importantes nestes tipos de problema:

1. a possibilidade de encontrar mínimos locais na proposição clássica da FWI;
2. o alto custo computacional requerido para realizar a inversão de forma de onda.

Um dos pontos relacionados ao custo computacional é o domínio no qual a FWI realiza seu processo de inversão. A execução da FWI pode ser realizada no domínio do tempo (BUNKS et al., 1995), mas é no domínio da frequência que se torna possível implementar com maior facilidade sua abordagem multiescala (PRATT, 1999). Neste último caso, consideram-se diferentes valores de frequência para inversão e isto contribui para redução do custo computacional do método.

2.2 Redes Neurais Artificiais

2.2.1 Modelo do Neurônio

O primeiro modelo de um neurônio artificial foi proposto por McCulloch e Pitts (1943). Em linhas gerais, o modelo de neurônio pode ser descrito como uma soma ponderada de entradas, acrescida de um fator *bias*.

$$v_k = \sum_{i=1}^n x_i w_i + b \quad (2.7)$$

A Equação 2.7 representa este modelo, onde v_k é o campo local induzido do neurônio k , b é o fator de *bias*, n é o número de entradas, x_i representa uma entrada i e w_i representa o peso associado à entrada i .

Em outras palavras, Haykin (2009) define este modelo como uma composição de três elementos básicos:

1. um conjunto de sinapses, ou elos de conexão, em que cada sinapse é caracterizada por um fator próprio de força (os pesos). Este peso sináptico pode conter valores negativos ou positivos;
2. um somador que combina linearmente as entradas ponderadas do neurônio;
3. uma função de ativação que limita a amplitude de atuação do neurônio.

Os elementos que compõem este modelo de neurônio estão ilustrados na Figura 3.

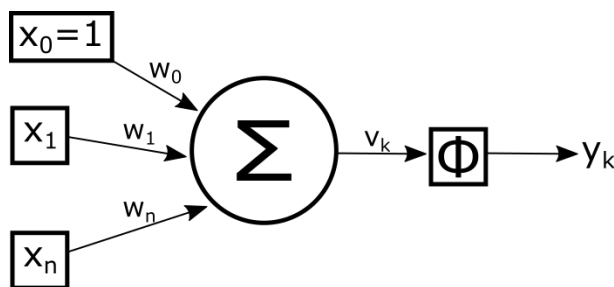


Figura 3 – Modelo de neurônio onde o campo induzido (v_k) é gerado pela soma ponderada das entradas, com $x_0 = 1$ e w_0 representando o fator *bias* e a saída y_k sendo gerada ao aplicar uma função de ativação ϕ em v_k

$$v_k = \sum_{i=0}^n x_i w_i \quad (2.8)$$

Além disso, Haykin (2009) indica que o fator de *bias* do modelo serve para ajustar a saída do combinador linear, aumentando ou diminuindo seu valor antes de se tornar uma entrada da função de ativação. Desta forma, pode-se ajustar a Equação 2.7 para a Equação 2.8, onde o fator *bias* passa a ser uma entrada constante ao modelo ($x_0 = 1$), ajustável com base em seu valor de peso w_0 .

$$y_k = \phi(v_k). \quad (2.9)$$

A saída y de um neurônio k é, portanto, determinada pela aplicação de uma função de ativação ϕ sobre o campo local induzido (v_k), ou potencial de ativação, deste neurônio (HAYKIN, 2009) (Equação 2.9).

2.2.2 Redes Multicamadas

Redes neurais de multicamadas são aquelas que possuem, ao menos, uma camada escondida em sua arquitetura, i.e., uma camada que reside entre os nós de entrada e saída da rede. Estas redes são conhecidas também como totalmente conectadas (*fully*

connected), no sentido de que tanto as suas entradas, quanto os neurônios de uma camada estão conectados a todos os neurônios da camada seguinte (Figura 4), mas podem assumir também outras formas, como redes recorrentes ou convolucionais. A adição de mais camadas à arquitetura da rede a possibilita extrair estatística de alta ordem das entradas, i.e., os conjuntos extras de sinapses permitem obter uma perspectiva global das características das amostras mesmo em regiões locais da rede (BISHOP, 2006).

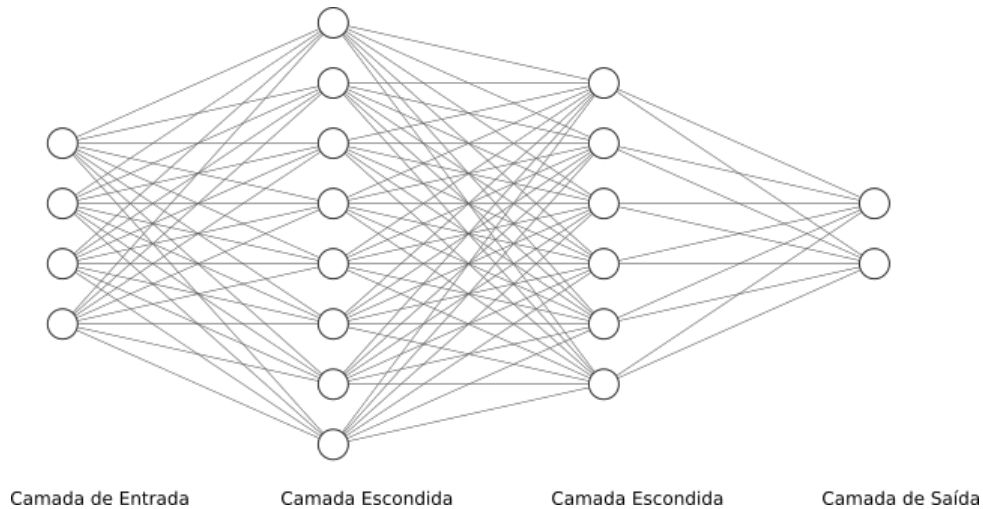


Figura 4 – Modelo de um Perceptron Multicamadas com 4 camadas: 1 camada de entrada, 2 camadas escondidas e 1 camada de saída.

$$E_n = \frac{1}{2} \sum_i (y_{nk} - \hat{y}_{nk})^2 \quad (2.10)$$

$$\frac{\partial E_n}{\partial w_{ki}} = (y_{nk} - \hat{y}_{nk})x_{nk} \quad (2.11)$$

$$\frac{\partial E_n}{\partial w_{ki}} = \frac{\partial E_n}{\partial v_k} \frac{\partial v_k}{\partial w_{ki}} \quad (2.12)$$

O algoritmo de retropropagação de erro, ou *backpropagation*, faz parte do processo de treinamento de uma rede neural *feedforward* multicamadas. Entretanto, seu uso não é exclusivo desta rede, podendo ser aplicado em diversos outros tipos de redes neurais, como recorrentes ou convolucionais. Este algoritmo tem como objetivo realizar de forma eficiente a atualização de pesos com base na avaliação do gradiente da função de erro associada a uma rede neural *feedforward*. A explicação do processo de retropropagação nesta dissertação utiliza a Equação 2.10 como função de erro de um padrão de dados n , onde \hat{y}_{nk} é saída de um neurônio k para o padrão de dados n e y_{nk} a saída esperada. Assim, o gradiente desta função de erro com relação a um peso w_{ki} é dado pela Equação

2.11, onde x_{nk} é o padrão de entrada n no neurônio k . Aplicando-se a regra da cadeia, redefine-se a Equação 2.11 na Equação 2.12.

Bishop (2006) compreende este algoritmo em duas partes: 1) realiza-se a retropropagação do erro e avalia-se as **derivadas da função de erro com relação aos pesos** da cada unidade da rede; 2) ajusta-se os valores dos pesos conforme as derivadas calculadas por meio de otimizadores. A segunda parte deste algoritmo será discutida mais na seção que aborda os otimizadores de redes neurais. Já a primeira etapa será apresentada, de forma resumida, nesta seção.

O Algoritmo 1 provê uma visão geral dos passos necessários para a execução do algoritmo de *backpropagation*. A etapa 1 deste algoritmo refere-se às instruções da linha 5 até 9 do pseudo-código.

Algoritmo 1 Backpropagation

Entrada: $dados = (x, y)^n$, épocas

Saída: pesos atualizados

```

1: programa BACKPROPAGATION( $dados$ , épocas)
2:   inicialize os pesos aleatoriamente;
3:   repita
4:     para todo  $(x_i, y_i) \in dados$  faça
5:       aplique um vetor de entrada  $x_i$  à rede e faça uma propagação direta dele
       através da rede;
6:       encontre a ativação de todas as unidades escondidas e de saída;
7:       avalie  $\delta_j$  para todas as unidades de saída utilizando a Equação 2.13;
8:       realize a retropropagação dos valores de erro de forma a obter  $\delta_k$  para cada
       unidade escondida na rede.
9:       avalie as derivadas;
10:      realize os ajustes dos pesos.
11:    fim para
12:  até fim do número de épocas
13: fim programa

```

$$\delta_j = \hat{y}_j - y_j \quad (2.13)$$

O cálculo do erro nas unidades de saída pode ser definida pela Equação 2.13 para o caso de a função de erro ser um somatório de quadrados (BISHOP, 2006). Neste caso, δ_j é o valor de erro de uma unidade de saída j da rede, \hat{y}_j a saída estimada na unidade j e y_j a saída esperada (verdadeira) da unidade em questão.

$$\delta_k = \frac{\partial E_n}{\partial v_k} \quad (2.14)$$

$$\delta_k = \frac{\partial E_n}{\partial v_k} = \sum_j \frac{\partial E_n}{\partial v_j} \frac{\partial v_j}{\partial v_k} \quad (2.15)$$

$$\delta_k = \phi'(v_k) \sum_j w_{jk} \delta_j \quad (2.16)$$

Por outro lado, o cálculo do erro δ_k nas demais unidades escondidas da rede *feedforward* é dada pela Equações 2.14. Neste caso, os valores de δ_k são os próprios gradientes e, portanto, são utilizados na etapa de otimização para atualização dos pesos da rede. A Equação 2.14 demonstra que as variações na função de erro dependem do campo induzido v_k . Aplicando-se a regra da cadeia, obtém-se a Equação 2.15, onde variações no campo induzido das unidades de saída (v_j) produzem variações na função de erro através de variações do campo induzido de uma unidade da rede (v_k). Por fim, ao substituir a Equação 2.14 na Equação 2.15 e utilizar as Equações 2.8 e 2.9 obtém-se a Equação 2.16, a qual indica que o valor de δ para cada unidade escondida k pode ser obtido ao retropropagar o valor de δ em unidades subseqüentes.

É importante frisar que δ_j contém o valor calculado pela Equação 2.13 para a camada imediatamente anterior à camada de saída. Entretanto, δ_j passa a assumir o valor calculado pela Equação 2.16 quando as unidades estão duas camadas ou mais distantes da camada de saída.

Por exemplo, considerando-se o modelo de 4 camadas ilustrado na Figura 4, tem-se que o valor de δ_j é calculado conforme fórmula da Equação 2.13 para a segunda camada escondida, já que ela está imediatamente anterior às unidades da camada em questão. Já no caso da primeira camada escondida, situada após a camada de entrada, o valor de δ_j para o cálculo do erro em suas unidades assume o valor do δ_k calculado na segunda camada escondida. Esta lógica amplia-se caso a rede neural possua um número maior de camadas.

2.2.3 Funções de Ativação

Conforme apresentado anteriormente, uma função de ativação é formalmente denotada por $\phi(v_k)$ e define a saída baseada em um campo local induzido (v_k). Sendo assim, uma função de ativação limita a amplitude de saída de um neurônio que pertence à rede neural, assim, determinando o quão influente é este neurônio na saída final, i.e., se ele será ativado ou não.

$$\phi(v_k) = \begin{cases} 1, & \text{se } v_k \geq 0 \\ 0, & \text{c.c} \end{cases} \quad (2.17)$$

$$\phi(v_k) = \frac{1}{1 + \exp(-v_k)} \quad (2.18)$$

Haykin (2009) aponta que existem dois tipos básicos de função de ativação: a função de limite (*threshold function*) (Equação 2.17), também conhecida como função degrau, e a função sigmoide (Equação 2.18), que é uma das mais comuns.

Enquanto a função degrau entrega duas opções de saída, 1 quando o campo local induzido (v_k) possui um valor nulo ou positivo e 0 quando negativo), a função sigmoide calcula a exponencial do negativo do campo local para definir os valores de saída. Isto faz com que esta função de ativação, apesar de possuir os mesmos intervalos vistos na Equação 2.17, ofereça como possibilidades de saída uma quantidade muito mais diversa de valores no intervalo de 0 até 1.

Existem outras variações de funções de ativação que são mais adequadas ao construir certos tipos de redes neurais, como redes convolucionais. Algumas destas funções são a: unidade linear retificada, do inglês *rectified linear unit* (ReLU), unidade linear retificada com “vazamento” - *leaky rectified linear unit* (Leaky ReLU) -, unidade linear exponencial - *exponential linear unit* (ELU) - e unidade linear retificada paramétrica - *parametric rectified linear unit* (PReLU).

2.2.3.1 ReLU

A unidade linear retificada, ou *Rectified Linear Unit* (ReLU) (NAIR; HINTON, 2010), varia de 0 a infinito para valores positivos, em contraste com as funções anteriores que eram limitadas a uma variação entre 0 e 1.

$$\phi(v_k) = \begin{cases} v_k, & \text{se } v_k \geq 0 \\ 0, & \text{c.c.} \end{cases} \quad (2.19)$$

A Equação 2.19 descreve a função de ativação ReLU. Apesar de possuir uma componente linear quando v_k é positivo, a ReLU é uma função não-linear e sua combinação com outras funções é, portanto, também, não-linear, o que, em conjunto com o fato de ser diferenciável, a torna possível de ser utilizada no algoritmo de *backpropagation*, conforme discutiremos em seções seguintes.

Entretanto, tal função possui um problema de que a saída de um neurônio é sempre zerada quando o valor do campo local induzido (v_k) for negativo (Figura 5), causando um fenômeno conhecido como “*Dying ReLU*” ou problema de vazamento (*leakage problem*). Quando isto acontece, um neurônio é completamente desativado e a capacidade de uma rede neural de aprender é comprometida porque o gradiente que guia o ajuste de pesos

para um neurônio específico é multiplicado por zero, conforme será visto na seção sobre otimizadores.

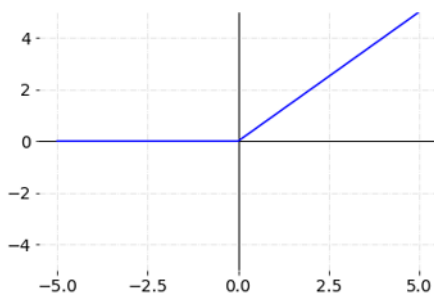


Figura 5 – ReLU Activation Function Plot

2.2.3.2 Leaky ReLU

A unidade linear retificada com “vazamento”, ou *Leaky Rectified Linear Unit* (Leaky ReLU) (MAAS; HANNUN; NG, 2013), é uma variação da função ReLU que tenta superar o problema do “*Dying ReLU*”.

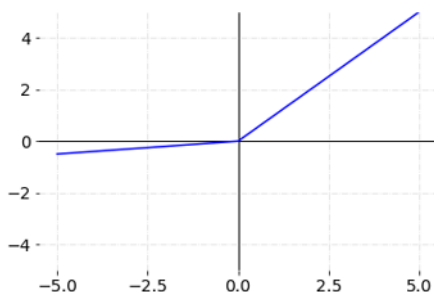


Figura 6 – Leaky ReLU Activation Function Plot

$$\phi(v_k) = \begin{cases} v_k, & \text{se } v_k \geq 0 \\ \alpha v_k, & \text{c.c.} \end{cases} \quad (2.20)$$

Pode-se visualizar na Equação 2.20 e Figura 6 que esta função agora varia entre $-\infty$ e ∞ e, diferentemente do ReLU, os valores negativos de v_k são multiplicados por uma constante alfa (α) ao invés de serem zerados.

Uma dificuldade, porém, pode existir ao definir o valor da constante α , já que diferentes valores atuam diferentemente nas saídas das camadas de neurônios que compõem a rede neural. Além disso, por ser constante, o valor escolhido de α pode atuar melhor em certas camadas do que em outras, o que dificulta mais sua escolha.

2.2.3.3 ELU

A unidade linear exponencial, ou *Exponential Linear Unit* (ELU) (DJORK-ARNÉ; UNTERTHINER; HOCHREITER, 2016), é outra alternativa ao problema de “*Dying ReLU*”. No lugar de multiplicar uma constante ao campo local (v_k) como o *Leaky ReLU*, esta função de ativação aplica uma exponencial que pode ser escalada por um valor de α (Equação 2.21), resultando em uma curva mais suave e não-linear para os valores negativos (Figura 7).

$$\phi(v_k) = \begin{cases} v_k, & \text{se } v_k \geq 0 \\ \alpha(\exp^{v_k} - 1), & \text{c.c.} \end{cases} \quad (2.21)$$

No caso da ELU, o valor de escala (α) não tem tanta influência na saída da função como acontece no caso do *Leaky ReLU*, já que ela é determinada, de fato, pela componente exponencial, tornando sua escolha um aspecto não tão importante.

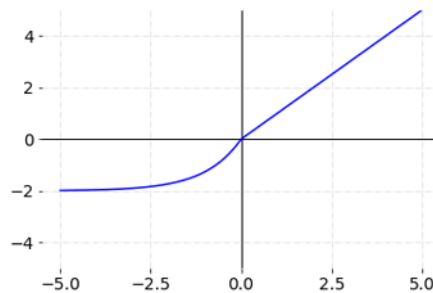


Figura 7 – ELU Activation Function Plot

2.2.3.4 PReLU

A unidade linear retificada paramétrica, ou *Parametric Rectified Linear Unit* (PReLU) (HE et al., 2015), é como uma forma generaliza do *Leaky ReLU*, já que, em vez de fixar um único valor de α para ser aplicado durante toda a execução da função de ativação, o PReLU calcula um valor α para cada k no campo local induzido (v_k) (Equação 2.22). Pode-se notar na Figura 8, os valores abaixo de zero podem variar muito, já que os valores de α em cada ponto podem ser bastante diferentes uns dos outros.

$$\phi(v_k) = \begin{cases} v_k, & \text{se } v_k \geq 0 \\ \alpha_k v_k, & \text{c.c.} \end{cases} \quad (2.22)$$

Como a função PReLU calcula automaticamente os valores de α , ela adiciona mais parâmetros ao treinamento da rede neural. Isto proporciona um pequeno acréscimo ao tempo de treinamento, mas proporciona melhoria significativa nos resultados (HE et al., 2015).

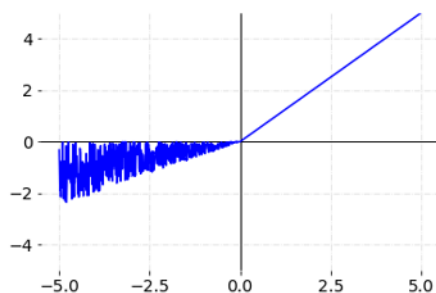


Figura 8 – PReLU Activation Function Plot

2.2.4 Otimizadores

O otimizador é uma parte crucial no processo de aprendizado de uma rede neural e é responsável por guiar sua convergência ao ótimo global do espaço de busca. Algoritmos de aprendizado supervisionados otimizam uma função objetivo, usualmente sua função de erro, também conhecida como função de perda, ou *loss function*, para descida de gradiente, que é o método numérico mais simples para realizar otimização (GORI, 2017).

$$\theta_{i+1} = \theta_i - \eta \nabla_{\theta_i} J(\theta_i) \quad (2.23)$$

A otimização da função de erro/*loss* ocorre durante o algoritmo de propagação reversa (*backpropagation*) (GORI, 2017) no caso de redes *feedforward* e pode ser realizada em lotes (*batches*) de dados (Equação 2.23). De forma geral, a função objetivo $J(\theta)$ é minimizada quando atualizações nos parâmetros (θ) do modelo são realizadas na direção oposta ao gradiente da função objetivo ($\nabla_{\theta} J(\theta)$). Esta otimização também depende de um parâmetro chamado taxa de aprendizado (η), ou *learning rate*, que define o tamanho do passo que o processo de otimização caminhará em direção ao mínimo global (Figura 9). Neste caso, calcula-se o valor de gradiente para o conjunto de dados como um todo a cada etapa de atualização, causando redundância nos cálculos e, logo, aumento do custo computacional quando usado em grandes conjuntos (RUDER, 2017).

Um tipo amplamente utilizado de algoritmo para otimização é o de descida do gradiente estocástica, ou *stochastic gradient descent* (SGD). O SGD é um método de otimização mais rápido quando comparado com abordagens clássicas de descida de gradiente

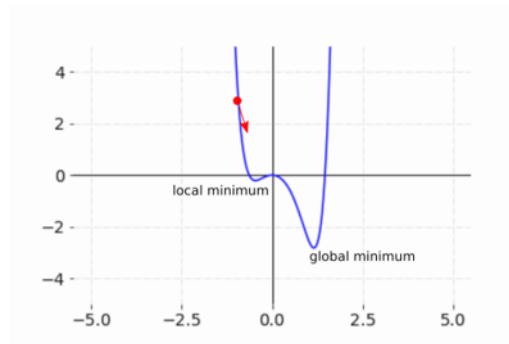


Figura 9 – Ilustração do processo de otimização em busca do mínimo global

em lote, já que evita operações de atualização redundantes nos gradientes ao calculá-los uma vez a cada atualização (RUDER, 2017). O SGD também permite o encontro de melhores resultados de mínimos locais, mas a convergência de mínimos exatos torna-se mais difícil por conta da flutuação de locais no espaço de busca inerentes ao método (RUDER, 2017). Ao tratar-se do SGD, é mais comum considerar sua versão para pequenos lotes (*mini-batch*), na qual a atualização é realizada para um subconjunto do conjunto de dados, em vez de uma amostra por vez (RUDER, 2017).

$$\theta_{i+1} = \theta_i - \eta \nabla_{\theta_i} J(\theta_i; x^{(k:k+m)}; y^{(k:k+m)}) \quad (2.24)$$

A atualização de parâmetros através do algoritmo SGD com *mini-batch* ocorre conforme a Equação 2.24, na qual θ são os parâmetros obtidos a partir de amostras do conjunto de treinamento, $x^{(k:k+m)}$ e $y^{(k:k+m)}$ são, respectivamente, entrada e o alvo de um lote que possui m amostras consideradas a partir da amostra k do conjunto de treinamento, η é a taxa de aprendizado e ∇_{θ_i} é o gradiente da função de erro definida por $J(\theta_i; x^{(k:k+m)}; y^{(k:k+m)})$, a qual foi obtida considerando-se os parâmetros θ_i e os pequenos lotes.

Conforme diminui-se o valor da taxa de aprendizado, o processo de otimização tende a convergir para o mínimo global do problema, mas isto pode aumentar consideravelmente o tempo de treinamento da rede neural (RUDER, 2017). Existem variações do SGD, como RMSprop, Adam ou Adamax, que tentam superar algumas de suas limitações.

2.2.4.1 RMSprop

O RMSprop é uma variação do Adagrad (DUCHI; HAZAN; SINGER, 2011), que por sua vez é uma variação do SGD. O maior problema encontrado no uso do Adagrad, e que o RMSprop propõe resolver, é sua tendência de encolher as taxas de aprendizado devido à acumulação do quadrado dos gradientes (RUDER, 2017). A atualização do

gradiente com o RMSprop ocorre de acordo com as seguintes equações:

$$g_i = \nabla_{\theta_i} J(\theta_i; x; y) \quad (2.25)$$

$$\theta_{i+1} = \theta_i - \frac{\eta}{\sqrt{E[g^2]_i + \epsilon}} g_i \quad (2.26)$$

$$E[g^2]_i = 0,9E[g^2]_{i-1} + 0,1g_i^2 \quad (2.27)$$

O gradiente em uma etapa i é definida pela Equação 2.25. Os parâmetros (Equação 2.26) são, então, atualizadas considerando $\sqrt{E[g^2]}$, que é a média do gradiente ao quadrado em uma dada etapa. $\sqrt{E[g^2]}$ é definida pela raiz quadrada da Equação 2.27 e considera a média da iteração anterior ($E[g^2]_{i-1}$) e o valor atual do gradiente ao quadrado (g_i^2) para determinar a média atual.

2.2.4.2 Adam e Adamax

Adamax é uma variação da estimação de momento adaptável, ou *Adaptive Moment Estimation* (Adam), sendo que o segundo também é uma variação do SGD. Ambos otimizadores computam taxas de aprendizado adaptáveis para cada parâmetro em vez de utilizar um único valor η durante todo o processo de treinamento. Isto é feito no Adam mantendo-se um histórico da queda exponencial média de ambos os quadrados dos gradientes (Equação 2.28) e gradientes passados (Equação 2.29), nos quais β_1 e β_2 são as taxas de decaimento de m_i e v_i respectivamente.

$$v_i = \beta_2 v_{i-1} + (1 - \beta_2) g_i^2 \quad (2.28)$$

$$m_i = \beta_1 m_{i-1} + (1 - \beta_1) g_i \quad (2.29)$$

$$\theta_{i+1} = \theta_i - \frac{\eta}{\sqrt{\hat{v}_i + \epsilon}} \hat{v}_i \quad (2.30)$$

Os parâmetros utilizando o Adam são atualizados conforme Equação 2.30, nos quais \hat{v}_i e \hat{m}_i , respectivamente, corrigem o enviesamento (*bias*) que v_i e m_i podem produzir e ϵ é um número de baixo valor utilizado para evitar divisões por zero. As diferenças entre Adam e Adamax reside no cálculo de v_i .

$$\theta_{i+1} = \theta_i - \frac{\eta}{\sqrt{u_i}} \hat{m}_i \quad (2.31)$$

Ao passo que o Adam atualiza v_i ao aplicar uma norma l_2 , Adamax realiza suas atualizações considerando uma norma l_∞ . A Equação 2.31 exibe como Adamax altera os parâmetros do modelo, no qual $u_i = \max(\beta_2 v_i, |g_i|)$ denota a norma l_∞ de v_i . Mais detalhes com relação a ambos otimizadores são melhor discutidos no trabalho de Kingma e Ba (2015).

2.2.5 Redes Convolucionais

Redes neurais convolucionais, ou *Convolutional Neural Networks* (CNNs), têm sua inspiração advinda do córtex visual do cérebro (MIN; LEE; YOON, 2017), tornando-as bastante adequadas para uso como modelos de *deep learning* em de imageamento. Apesar de a primeira implementação de CNN ter sido proposta por LeCun et al. (1989), a partir da competição da ImageNet em 2012, onde a AlexNet, implementação proposta por Krizhevsky, Sutskever e Hinton (2012), alcançou um resultado de classificação de imagens melhores até mesmo que classificadores humanos, o uso e pesquisa de CNNs e suas variações aumentou significativamente.

A ideia por detrás das redes convolucionais segue o princípio do banco de filtros de Gabor para a extração de atributos, que propõe a criação de um banco de N filtros que é, filtro por filtro, convoluído com a imagem de entrada, produzindo N diferentes imagens com diferentes características da imagem original (AGHDAM; HERAVI, 2017). No caso das redes convolucionais, diversas operações de convolução e *pooling* são realizadas em cascata para a extração de características.

As operações de convolução são realizadas pelas camadas de convolução de uma rede convolucional. Estas camadas, diferentemente de camadas totalmente conectadas, não possuem todos seus neurônios conectados entre si. Desta forma, é possível relacionar cada neurônio da camada de convolução a uma região da entrada, já que os atributos que estão próximos uns dos outros possuem mais correlação do que atributos posicionados mais distante. Assim, a operação de convolução percorre a amostra de entrada com base em um determinado tamanho de passo (*kernel size*) e ajustam os valores de pesos de cada neurônio de forma a construir os filtros que melhor extraem as características daquela região (*feature maps*).

As camadas de *pooling*, por sua vez, são utilizadas para reduzir a resolução do *feature map* resultante de uma operação de convolução, i.e., aplicar uma operação de *downsampling* obedecendo um determinado fator de escala. Ao diminuir a resolução, a operação reduz a complexidade do *feature map* para operações de convolução subsequentes, além de manter as características mais marcantes identificadas até o momento, permitindo que estas características sejam cada vez melhor identificadas.

O oposto da operação de *pooling* chama-se *upsampling*. Esta operação aumenta a resolução dos *feature maps*, permitindo que as características identificadas anteriormente possam ser localizadas no mapa de características.

2.2.5.1 Redes totalmente convolucionais

Redes totalmente convolucionais, ou *Fully Convolutional Networks* (FCNs), tiveram seu uso inicialmente sugerido para o trato de problemas de segmentação semântica (LONG; SHELHAMER; DARRELL, 2015), os quais têm como objetivo tentar classificar diferentes seções de uma imagem em uma determinada classe e determinar uma região (máscara) que delimitam os *pixels* que representam estas seções. A Figura 10 é um exemplo de uma segmentação semântica. Neste caso, algumas das seções que foram classificadas com as cores de suas respectivas máscaras são pessoas (vermelho), semáforos (amarelo), carros (azul), rua (rosa) e vegetação (verde).

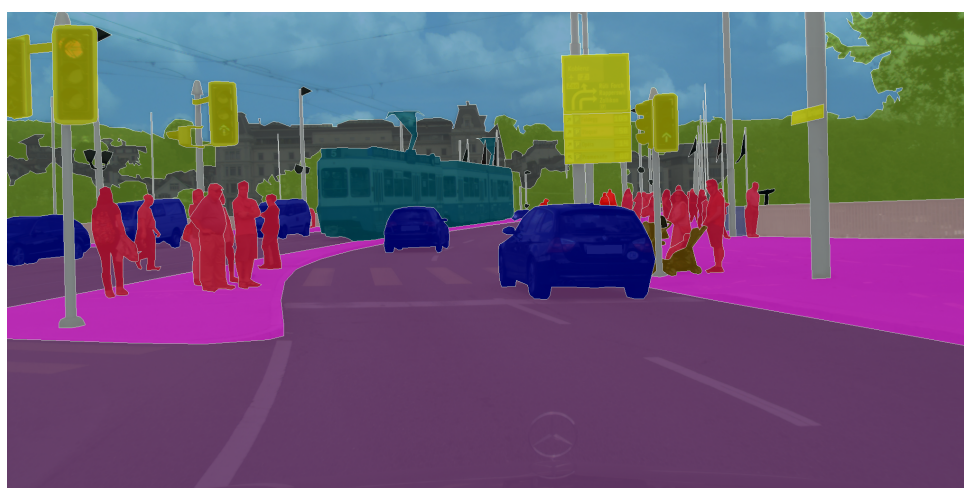


Figura 10 – Exemplo de uma imagem de segmentação semântica. Fonte: (CORDTS et al., 2016)

FCNs são redes convolucionais construídas exclusivamente com camadas de convolução, *pooling*, *upsampling* ou *funções de ativação*. Portanto, como não são utilizadas camadas totalmente conectadas (ou densas), como em redes convolucionais comuns ou perceptron multicamadas, estas redes produzem filtros profundos não lineares bem adequados para trabalhar com imagens usando menos parâmetros e tempo computacional, além de serem menos dependentes a variações no tamanho da imagem (LONG; SHELHAMER; DARRELL, 2015).

$$y_{ij} = f_{ks}(\{x_{si+\Delta i, sj+\Delta j}\}_{0 \leq \Delta i, \Delta j \leq k}) \quad (2.32)$$

A Equação 2.32 é utilizada para definir uma FCN, onde uma determinada saída

y_{ij} depende do tipo de camada f_{ks} utilizada com um *kernel* de tamanho k e um fator de subamostragem, ou *stride*, de tamanho s operando sobre uma entrada $x_{si,sj}$.

2.3 Redução de Dimensionalidade

Em um conjunto de dados uma amostra (instância) pode ser representada por uma série de atributos (*features*). Essa quantidade de atributos de um conjunto de dados determina sua dimensionalidade. Assim, um dado representado por 2 atributos é dito ser bi-dimensional, por três tri-dimensional e por n multi-dimensional. É evidente que dados até tri-dimensionais são de fácil visualização, mas, conforme a dimensionalidade aumenta, a visualização de tais conjuntos torna-se impraticável, além de acarretar em uma maior complexidade em termos de processamento e análise dos dados.

Neste contexto, a redução de dimensionalidade é o processo de mapeamento de uma instância de n dimensões para uma dimensão k menor que a original sem que haja uma perda considerável na estrutura original dos dados. Este processo reduz o tamanho para representar e armazenar um conjunto de dados, além de promover uma visualização das amostras no conjunto (VLACHOS, 2010) e reduzir a complexidade para processamento e análise de dados.

A redução de dimensionalidade de um conjunto de dados pode ser alcançada aplicando-se diversas técnicas diferentes. Dentre todas as variedades de técnicas, as subseções seguintes apresentam três métodos: análise de componentes principais, do inglês *principal component analysis* (PCA), *autoencoders* (AE) e *variational autoencoders* (VAEs).

2.3.1 Análise de Componentes Principais (PCA)

A análise de componentes principais é uma técnica de análise de dados multivariados cujo objetivo é extrair as informações importantes presentes em um conjunto de dados ($X_{n,p}$) que possui (n) amostras descritas por inúmeros atributos (p) dependentes e, em geral inter-correlacionados, de forma a expressá-las em variáveis ortogonais (chamadas de componentes principais) com base em sua variância (ABDI; WILLIAMS, 2010).

$$C(S, Q) = \frac{1}{n-1} \sum_{i=1}^n (S_i - \bar{S})(Q_i - \bar{Q})^T = \frac{1}{n-1} SQ^T \quad (2.33)$$

$$C(S, Q) = \begin{pmatrix} \sigma(S, S) & \sigma(S, Q) \\ \sigma(Q, S) & \sigma(Q, Q) \end{pmatrix}, \text{ onde } \sigma(S, Q) = \sigma(Q, S) \quad (2.34)$$

As componentes do PCA podem ser obtidas por meio da matriz de covariância do conjunto de dados. O primeiro passo é calcular a média de cada atributo p existente em

X e subtraí-los de seus respectivos atributos para que sejam centralizados em torno da origem da distribuição e para calcular-se a matriz de covariância dos dados. Considerando um caso bi-dimensional ($p = 2$), a matriz de covariância pode ser calculada conforme a Equação 2.33, onde S_i representa o valor da amostra i da dimensão S com \bar{S} sendo sua média e Q_i representa o valor da amostra i da dimensão Q com \bar{Q} sendo sua média. Ao final, a matriz de covariância assume a forma descrita na Equação 2.34.

$$C = \sum_{i=1}^r \lambda_i v_i v_i^T \quad (2.35)$$

A variância capturada durante o PCA é representada pelos autovalores da matriz de covariância. Ao aplicar-se a decomposição dos autovalores da Equação 2.33, encontra-se a representação dada pela Equação 2.35, onde r é o número total de componentes, λ_i é a variância (ou autovalor) do componente i e v_i representa o i -ésimo componente principal.

2.3.2 Autoencoder (AE)

Autoencoders (AE) são modelos de *deep learning* conhecidos por aprenderem de uma maneira auto-supervisionada e que podem ser vistos como uma generalização do PCA. Isto significa que estes modelos possuem saídas (*targets*) assim como os supervisionados, logo não podem ser rotulados totalmente como modelos não supervisionados, tampouco podem ser definidos como totalmente supervisionados, já que os dados de suas saídas são os mesmos dados de entrada. De acordo com Francois (2017), modelos de aprendizado auto-supervisionados, como *autoencoders*, podem ser vistos tanto como modelos supervisionados ou não-supervisionados, dependendo se a atenção é direcionada ao mecanismo de aprendizado ou ao contexto da aplicação.

O objetivo geral de um autoencoder (Figura 11) é aprender a reconstruir os atributos originais dos dados de entrada utilizando um paradigma de *encoder-decoder* no modelo. O módulo de codificação (*encoder*) interpreta os atributos dos dados de entrada e o mapeia para uma representação latente contendo informações comprimidas (de menor dimensionalidade) do dado original. Como a saída do *autoencoder* é a própria entrada, o módulo de decodificação (*decoder*) encarrega-se de reconstruir a representação latente do dado de entrada em uma saída com as mesmas dimensões e informações presentes no dado original.

2.3.3 Variational Autoencoder (VAE)

Os *variational autoencoders* (VAEs), como o nome sugere, são modelos que partem do mesmo princípio dos *autoencoders* para codificação e decodificação de dados, mas

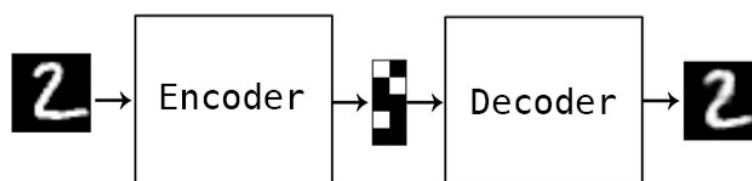


Figura 11 – Módulos de *encoder* e *decoder* de um Autoencoder (FRANCOIS, 2017)

misturando-se o conceito de inferência Bayesiana¹, fazendo com que VAEs aprendam representações latentes dos dados contínuas e altamente estruturadas (FRANCOIS, 2017) (Figura 12).

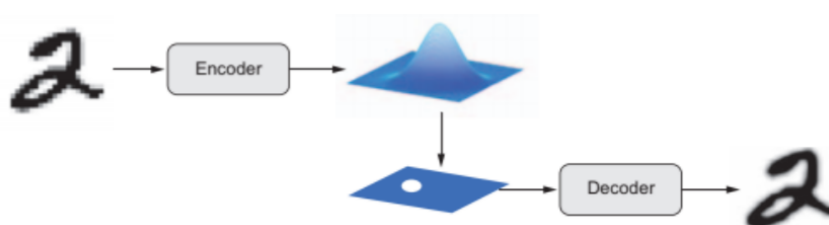


Figura 12 – Representação de um variational autoencoder (FRANCOIS, 2017)

Assim, os dados de dimensão reduzidas representam os parâmetros de média e variância de uma distribuição estatística, permitindo assumir que os atributos dos dados de entrada foram gerados a partir de um processo estatístico do qual a aleatoriedade deve ser considerada no processo de codificação e decodificação do dado. Isto aumenta a robustez do processo de redução de dimensionalidade e faz com que o espaço latente possua representações significativas em toda a sua extensão, i.e, cada ponto da distribuição pode ser decodificado em uma saída válida. Devido a isto, VAEs são utilizados como método generativos além de métodos de redução de dimensionalidade, ou seja, são capazes de gerar amostras totalmente novas além das existentes no conjunto original.

2.4 Estado da Arte

Conforme dito anteriormente, o modelo de velocidades buscar representar uma subsuperfície com base na velocidade que as ondas geradas pelas fontes durante a aquisição sísmica se propagam em diferentes meios. Esta análise de velocidades se caracteriza como um problema inverso, já que as velocidades de propagação são estimadas com base em

¹ Método estatístico que atualiza o conhecimento sobre parâmetros desconhecidos com base na teoria Bayesiana (BOX; TIAO, 2011) conforme mais dados são disponibilizados

medidas indiretas obtidas do meio (SANTOS, 2013).

$$F(m_v) = d_{obs} \quad (2.36)$$

O dado sísmico observado durante o processo de aquisição (d_{obs}) é representado como o resultado da aplicação de um operador F sobre o modelo m_v que possui como parâmetro as velocidades de propagação de onda, conforme visto na Equação 2.36.

$$m_v = F^{-1}(d_{obs}) \quad (2.37)$$

Em um caso realístico, o modelo m_v é o objeto a ser encontrado e d_{obs} é a informação, obtida durante o processo de aquisição sísmica, que se possui sobre a subsuperfície. Sendo assim, obtém-se m_v quando um operador inverso F^{-1} é aplicado para a modelagem dos dados sísmicos da subsuperfície (d_{obs}), conforme a Equação 2.37.

Santos (2013) aponta que aproximações são necessárias para que se obtenha o operador inverso de modelagem, já que a propagação de ondas sísmicas é bastante complexa. Assim, alguns métodos podem ser empregados para a geração de um modelo aproximado, ou inicial, que contenha os parâmetros de velocidade de uma subsuperfície.

Um deles é o método de geração de modelos por meio da reflexão por tomografia (*tomography reflection*) (CARRION, 1995). Este método é realizado em duas etapas:

1. identificar horizontes acústicos e determinar o tempo de percurso relativo às várias combinações de fontes e receptores;
2. estimar iterativamente os modelos de velocidades locais.

A exatidão na escolha dos horizontes, porém, pode ser limitada por fatores como ruídos, interferência com outros eventos, rotação de fase, entre outros.

Por sua vez, a análise de velocidade por migração (*migration velocity analysis*) (LIU; BLEISTEIN, 1995) baseia-se na ideia de que valores errados de velocidade podem distorcer as imagens nos dados migrados. Por conseguinte, a mudança residual no eixo de profundidade dessas imagens pode ser avaliado por meio desse erro no valor de velocidade.

Este método estima diretamente, e de forma iterativa, o intervalo de velocidades em uma análise de *common-image gathers* (CIGs), assumindo que as velocidades são constantes em cada camada do modelo de velocidades e separadas por uma interface (refletor) suavizada. De forma geral, os traços sísmicos são ordenados em CIGs após a migração de pré-empilhamento e calculam-se, então, os pesos de interpolação com

base na mudança residual da área migrada para atualização dos valores de velocidades. Entretanto, [Liu e Bleistein \(1995\)](#) indicam que a execução repetida do processo de migração é inevitável para a atualização das velocidades em estruturas complexas e que sua eficiência depende do algoritmo de migração utilizado, o que denota, em linhas gerais, um alto custo computacional para esta técnica.

Uma alternativa tanto para a técnica de reflexão por tomografia, quanto para a de análise de velocidade por migração é o uso de métodos de busca global ([SEN; STOFFA, 2013](#)), como algoritmos genéticos (AG) ([HOLLAND et al., 1992](#)) ou otimização por enxame de partículas (*particle swarm optimization* - PSO) ([KENNEDY; EBERHART, 1995](#); [SHI; EBERHART, 1998](#)). Uma utilização de algoritmos genéticos, por exemplo, é discutida por [Sajeva et al. \(2016\)](#). Neste trabalho, aplicam-se os conceitos de algoritmos genéticos para gerar modelos de velocidades 2D com base em um campo de busca bem definido a partir de modelos iniciais aleatórios.

[Shaw e Srivastava \(2007\)](#) e [Mojica e Kukreja \(2019\)](#), por outro lado, demonstram a aplicabilidade do processo de inversão utilizando otimização por enxame de partículas para modelos, respectivamente, unidimensionais e bidimensionais. Por fim, no trabalho de [Mojica et al. \(2019\)](#) vê-se a comparação da aplicação de implementações com base em elitismo do AG e do PSO para geração de um modelo de velocidades inicial baseado no modelo sintético Marmousi ([VERSTEEG, 1994](#)).

Apesar de promissores, o uso de métodos globais para geração de modelos de velocidades tem duas principais limitações: 1) seu alto custo computacional e 2) inversão de um único tipo de modelo de velocidades por vez.

O custo computacional para execução de tais modelos aumenta conforme aumentam a complexidade e o tamanho do modelo, já que será necessário ampliar o espaço de busca. Uma forma de se fazer isto é por meio da inserção de mais modelos iniciais aleatórios, o que, conseqüentemente, aumenta o número de modelagens necessárias a cada iteração. Além disso, para cada novo dado observado do qual deseja-se gerar um modelo de velocidades, tanto o AG, quanto o PSO devem ser executados por completo, já que o espaço de busca depende dos modelos aleatórios iniciais, do dado observado utilizado e, indiretamente, do modelo de velocidades resultante.

Assim, alguns trabalhos mais recentes têm proposto o uso de técnicas de *deep learning* para a análise dos parâmetros de velocidade e geração de modelos iniciais. [Röth e Tarantola \(1994\)](#) discorrem sobre a possibilidade de aplicação de redes neurais para inversão sísmica e demonstram sua aplicabilidade em problemas inversos não triviais. [Lewis e Vigh \(2017\)](#) aplicam um modelo de *deep learning* baseado em redes convolucionais para gerar um mapa probabilístico que indica as regiões que possuem e as que não possuem

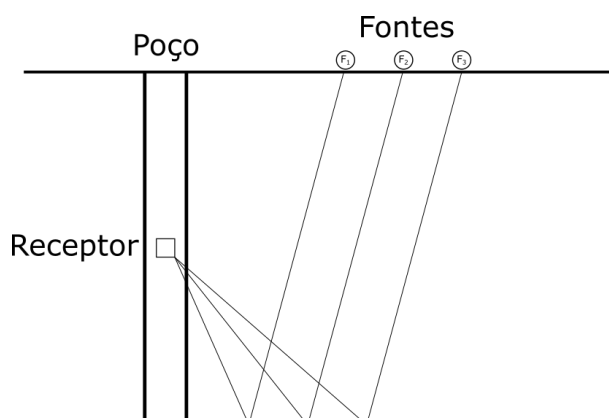


Figura 13 – Arranjo de fontes e receptores durante uma aquisição sísmica de poço.

estruturas em imagens sísmicas migradas.

Araya-Polo et al. (2018) aplicam um modelo de *deep learning* multicamadas totalmente conectado diretamente nos dados sísmicos, tendo como alvo modelos de velocidades sintéticos de duas dimensões. Wu, Lin e Zhou (2018) utilizam redes convolucionais para propor a InversionNet, um modelo com estrutura de *encoder-decoder* possuindo um campo condicional aleatório (*conditional random field* - CRF) ao final do *decoder* que gera modelos de velocidades com base nos dados sísmicos.

Por fim, Wang, Yang e Ma (2018) aplicam uma U-Net para a estimação de modelos de velocidades representados por um *grid* de $x, z = 150, 80$ diretamente dos dados sísmicos. Neste caso, porém, os dados sísmicos são gerados a partir de uma modelagem de poço, na qual a(s) fonte(s) é(ão) posicionada(s) no topo da subsuperfície e o(s) receptor(es) posicionado(s) a uma certa profundidade do topo (Figura 13), facilitando o processo de inversão. Além disso, os autores fazem uma comparação dos modelos estimados utilizando uma modelagem com apenas 1 fonte e outra com 10 fontes, onde o segundo caso obteve melhores resultados. Entretanto, outros experimentos com mais fontes e diferentes configurações de modelagem não foram realizados.

3 Metodologia

A metodologia adotada nesta dissertação consiste em utilizar uma abordagem de aprendizado supervisionado do modelo de *deep learning*, i.e., para cada uma das amostras de entradas utilizadas para o treinamento do modelo, conhece-se sua saída esperada. Desta forma, propõe-se abordar o problema de geração do modelo inicial de velocidades como um problema de regressão: a partir de um dado sísmico de entrada, gera-se-á seu modelo de velocidades correspondente. Os modelos de *deep learning* desta dissertação foram implementados utilizando as bibliotecas Tensorflow (ABADI et al., 2016) e Keras (CHOLLET et al., 2015). As demais ferramentas, como PCA, pré-processamento e visualização dos dados foram feitas utilizando, respectivamente, as bibliotecas Scikit-Learn (PEDREGOSA et al., 2011), Numpy (OLIPHANT, 2006) e Matplotlib (HUNTER, 2007). Discorre-se, a seguir, separadamente sobre os modelos de velocidades, o modelo de *deep learning* e as formas utilizadas para analisar os modelos estimados.

3.1 Modelos de velocidades e Sismograma

O primeiro ponto considerado para a realização do experimento descrito nesta dissertação foi a criação de modelos de velocidades que possuíssem algum significado geofísico, i.e.: as estruturas representadas no modelo estão divididas em mais de uma camada, onde cada camada possui variações verticais ou horizontais de velocidade, possuem ou não regiões com ondulações e possuem estruturas com falhas ou sem falhas.

Como forma de manter simples a representação geofísica dos modelos de velocidades e para facilitar seu processo de criação, optou-se pela limitação do seu número total de camadas entre 8 e 12 camadas, existência apenas de variação horizontal, estrutura de falha em apenas uma região e a velocidade mínima e máxima foi limitada entre, respectivamente, 1500 e 3500 m/s, onde 1500 m/s representa a lâmina d'água na subsuperfície marinha. O valor de 3500 m/s foi escolhido como limite a fim de se manter uma representação geológica não complexa, sem, por exemplo, corpos de sal. Adicionou-se também a possibilidade de as camadas nos modelos estarem inclinadas ou possuírem ondulações. A velocidade muda de uma camada para outra (variação horizontal) conforme a seguinte taxa $VEL_{incr} = \frac{(VEL_{max} - VEL_{min})}{num_camadas}$, onde VEL_{max} é a velocidade máxima do modelo, VEL_{min} a velocidade mínima, $num_camadas$ é o número de camadas total em um dado modelo e VEL_{incr} a quantidade de velocidade a ser incrementada de uma camada para outra. Neste experimento, considerou-se modelos de velocidades de subsuperfícies marinhas, por conseguinte, a primeira camada dos modelos representa a lâmina d'água.

A Figura 14 detalha em forma de fluxograma o processo para criação de um modelo

de velocidades. Um total de 1020 modelos de velocidades sintéticos bi-dimensionais foram gerados de forma aleatória conforme as características descritas no parágrafo anterior para o experimento). Um exemplo de tal modelo pode ser visto na Figura 15.

Após a criação dos modelos, aplicou-se, a cada um deles separadamente, uma modelagem sísmica baseada em diferenças finitas da equação de onda acústica utilizando uma Série de Taylor de ordem 32, gerando um dado sísmico similar ao ilustrado na Figura 16. Durante a modelagem sísmica, ambos receptores e fontes estão posicionados no eixo x . Esta configuração de posicionamento para fontes e receptores foi escolhida porque ela é mais provável de ocorrer durante o processo de aquisição de dados reais na exploração de subsuperfícies. A Figura 17 ilustra o processo de geração de dados sísmicos.

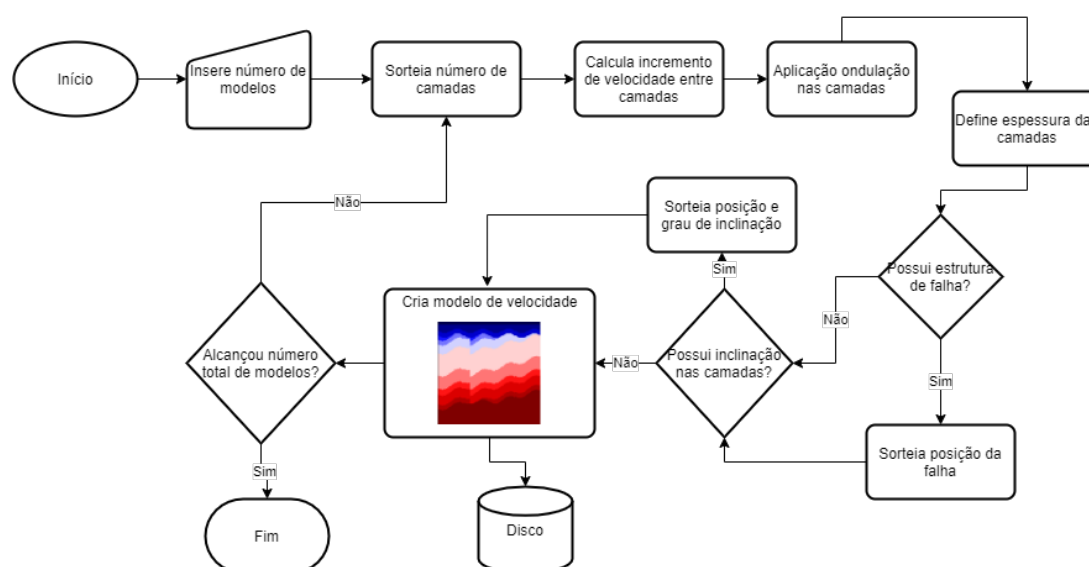


Figura 14 – Fluxograma do processo de criação dos modelos de velocidades.

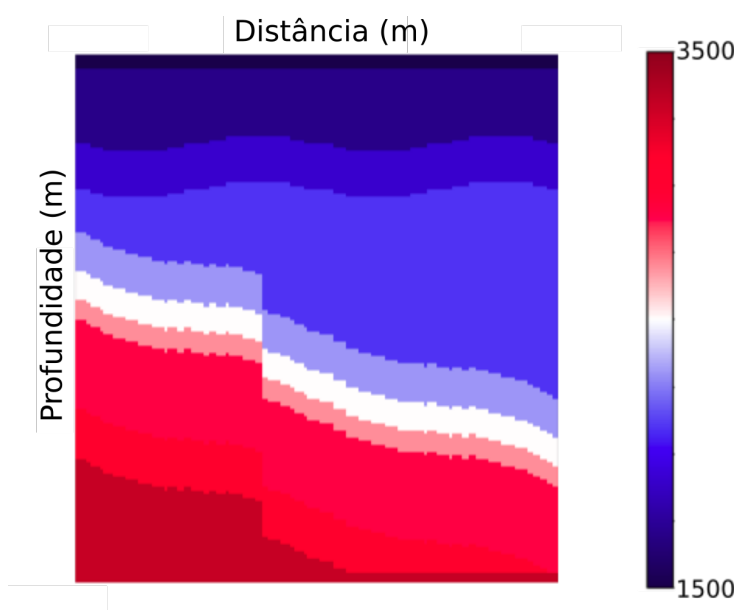


Figura 15 – Um modelo de velocidades sintético contendo 10 camadas, ondulações, inclinações e estruturas de falha

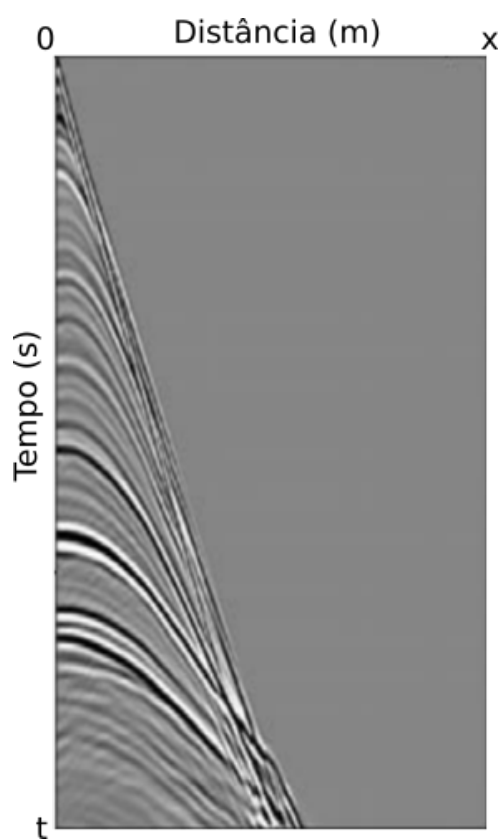


Figura 16 – Exemplo de dado sísmico sintético gerado por meio da modelagem sísmica de uma única fonte sísmica. Fonte: (SANTOS; PESTANA, 2016)

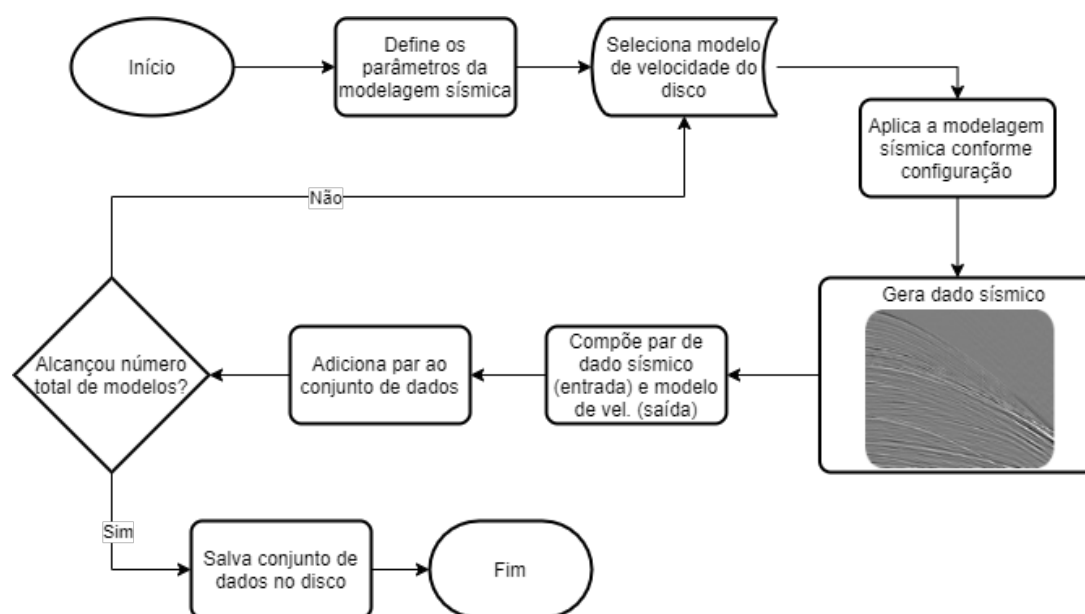


Figura 17 – Fluxograma do processo de criação dos dados sísmicos para cada modelo de velocidades.

3.2 Modelos *Deep Learning*

Nesta dissertação, os sismogramas são utilizados como o parâmetro de entrada do modelo de *deep learning* e os modelos de velocidades são usados como sua saída (alvo). Assim, pode-se entender esta aplicação como um problema de regressão ou, no caso específico da geofísica, de inversão sísmica onde, dado um sismograma, tentar-se-á obter o modelo de velocidades correspondente. Esta inversão é guiada pela redução do erro quadrático médio, medida de norma- l_2 que representa a função de erro do modelo de *deep learning*, entre o modelo estimado e sua contrapartida verdadeira.

O modelo de *deep learning* utilizado é uma rede totalmente convolucional, do inglês *Fully Convolutional Network* (FCN), construída especificamente seguindo a arquitetura U-Net (RONNEBERGER; FISCHER; BROX, 2015), cujos resultados provam-se satisfatórios quando aplicados em diversas áreas do conhecimento (WANG; YANG; MA, 2018) (LONG; SELHAMER; DARRELL, 2015). De forma resumida, a U-Net possui duas partes:

1. **Encoder:** composto por camadas convolucionais e de *pooling*, responsáveis por realizar a extração das características dos sismogramas, i.e., identificar as regiões do sismograma que melhor representam o modelo de velocidades. A saída do *encoder* é, portanto, uma versão comprimida do sismograma. Entretanto, as informações de localização destas características no espaço de entrada são perdidas, já que a resolução do sismograma é reduzida a cada operação de *pooling* realizada;

2. **Decoder:** composto por camadas de *upsampling* e convolucionais, realizando, assim, um aumento de resolução na forma reduzida resultante do *encoder*. Este aumento de resolução permite a localização das regiões que foram identificadas como as mais relevantes pelo *encoder*.

Além disso, uma U-Net toma proveito de um processo chamado de *skip connection*, o qual concatena informações locais obtidas em camadas anteriores ao *pooling* no *encoder* às informações globais captadas nas camadas posteriores ao *upsampling* no *decoder*. Isto é feito como uma tentativa de melhoria de localização das características. Ao final do *decoder*, aplica-se mais uma camada de convolução em uma porção do resultado do *decoder* a fim de realizar-se a inversão para o modelo de velocidades. A Figura 18 ilustra a estrutura geral da U-Net utilizada nesta dissertação, onde n_s é o número de fontes utilizada na modelagem dos dados sísmicos e representa o número de canais na camada de entrada.

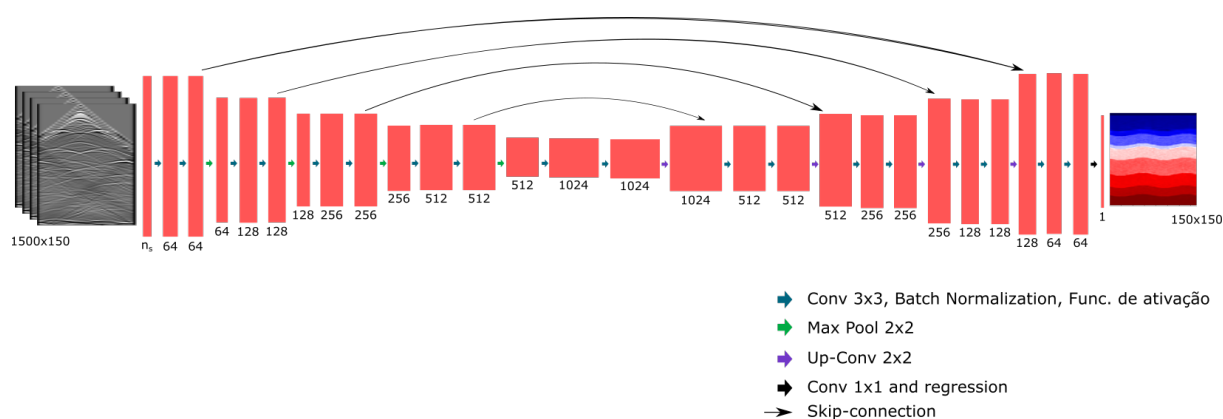


Figura 18 – Esquema da U-Net utilizada nesta dissertação. n_s representa o número de canais na camada de entrada, formando assim uma entrada de $1500 \times 150_s$. Os blocos vermelhos além do primeiro representam as demais camadas e os valores abaixo deles representam o número de canais nestas camadas.

Em adição à U-Net, foi aplicada também uma outra FCN conhecida como Hyper-Column (HARIHARAN et al., 2015). O nome da rede é, segundo os autores, derivado da definição de *hypercolumn* na neurociência, o qual descreve um conjunto de neurônios V1 organizados em formato de coluna sensíveis à detecção de bordas em diversas orientações. Os autores indicam que isto é apenas uma notação geral, já que o modelo deles não só detecta bordas, mas também unidades semânticas no dado de entrada. A noção de *hypercolumn* de uma camada, portanto, está definida como as saídas de todas as unidades acima desta camada empilhadas em um único vetor (Figura 19).

Este modelo de rede totalmente convolucional difere da U-Net principalmente em dois pontos. O primeiro ponto diz respeito à topologia, onde não se utiliza uma composição

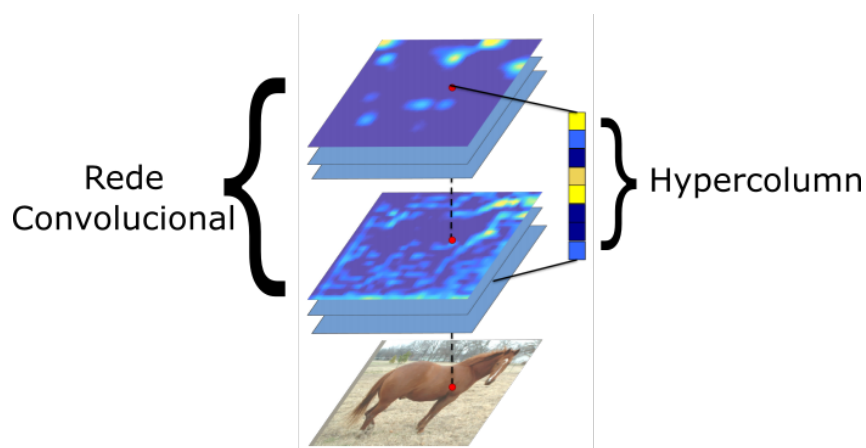


Figura 19 – Representação de um *hypercolumn*. O *hypercolumn* de um pixel é o vetor de ativações de todas as unidades acima deste pixel. Fonte: (HARIHARAN et al., 2015).

de camadas *max-pooling* e *upsampling* em estruturas de *encoder-decoder*. No caso da rede HyperColumn, após cada operação, seja ela uma convolução ou *pooling*, aplica-se uma operação de convolução seguida de um redimensionamento do dado por meio de interpolação (*upsample*) para o tamanho do dado de entrada. Estas operações ocorrem em paralelo ao fluxo “natural” da rede convolucional, i.e., a saída da camada de *upsample* é independente e não constitui uma entrada para a camada posterior, como ocorre na U-Net.

O segundo ponto trata justamente da geração da saída na HyperColumn. Ao passo que a U-Net utiliza a última camada do *decoder* para produzir a saída da rede, o HyperColumn concatena cada camada de *upsample* e processa o resultado desta concatenação para gerar o resultado final. A Figura 20 apresenta uma rede HyperColumn utilizada por Hariharan et al. (2015) para classificação de imagens.

No processo de aprendizagem dos dois tipos de modelo (U-Net e HyperColumn), o conjunto de dados é composto pelos dados sísmicos como entrada e os modelos de velocidades sintéticos como saída e é dividido em três sub conjuntos: 1) de treinamento, que é apresentado ao modelo de *deep learning* durante sua etapa de aprendizado e sobre o qual a atualização de pesos é realizada; 2) de validação, que é apresentado à FCN ao final de cada época de treinamento a fim de calcular-se as métricas de avaliação do modelo a cada época, as quais são consideradas para a otimização dos hiperparâmetros do modelo de *deep learning*; 3) de teste, cujas composições contêm amostras que são apresentadas ao modelo de *deep learning* **somente ao final** do processo de aprendizagem.

3.3 Formas de Análise

Avalia-se e analisa-se o modelo de *deep learning* após seu treinamento com base apenas no conjunto de dados separado para teste, garantindo-se, assim, que o modelo

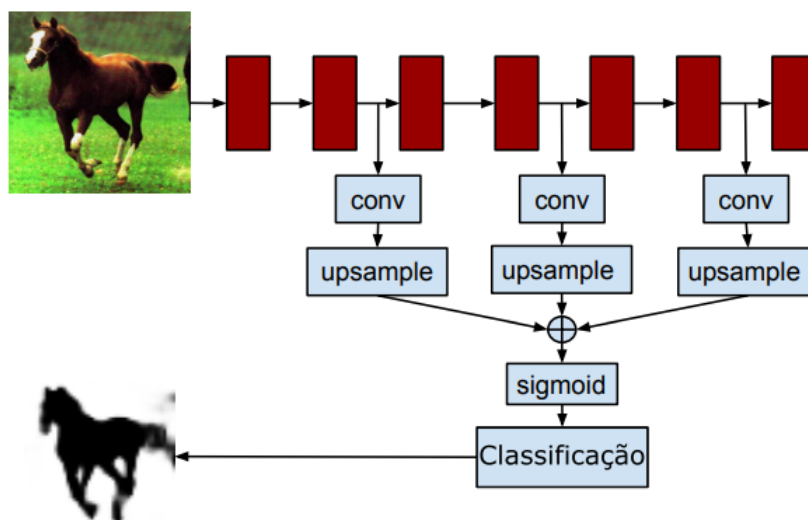


Figura 20 – Concatenação das camadas de *upsample* e geração saída do modelo HyperColumn. As camadas em vermelho representam operações de convolução. Ao contrário da U-Net, as camadas em vermelho não contribuem para a produção da saída da rede e as saídas das camadas de *upsample* não servem como entrada para as camadas posteriores.

generalize bem e não esteja enviesado ao conjunto de dados. Há dois tipos de avaliação aplicadas ao modelo: 1) qualitativo, aplicado tanto durante o processo de aprendizagem, quanto após finalizado o treinamento, no qual apresentam-se e analisam-se visualmente alguns modelos de velocidades que a FCN foi capaz de gerar a partir de dados sísmicos desconhecidos por ele; 2) quantitativo, cuja aplicação e análise mais detalhada é feita após o processamento de treinamento utilizando-se as seguintes métricas de regressão:

1. erro quadrático médio - em inglês *mean squared error* (MSE) - que é, também, a função a ser minimizada pelo modelo de *deep learning*
2. erro absoluto médio - em inglês *mean absolute error* (MAE)
3. coeficiente de correlação de Pearson (r)
4. coeficiente de determinação (R^2)
5. fator de dois (fac_2)

$$MSE = \frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2 \quad [m^2/s^2] \quad (3.1)$$

O erro quadrático médio (MSE), medido em m^2/s^2 , descrito na Equação 3.1 mede o quão longe um modelo de velocidades estimado está de seu valor verdadeiro. Esta métrica é também a função a ser minimizada pela FCN durante o processo de atualização dos

pesos. Assim, quanto maior a diferença entre uma saída e seu respectivo alvo, maior é a penalização ao modelo e, conseqüentemente, o erro associado às suas estimações. Na Equação 3.1 e em todas as equações seguintes, os parâmetros y indicam a saída verdadeira, \bar{y} o valor médio da saída verdadeira, \hat{y} a saída estimada e $\bar{\hat{y}}$ o valor médio da saída estimada.

$$MAE = \frac{1}{n} \sum_{i=1}^n |y_i - \hat{y}_i| \quad [m/s] \quad (3.2)$$

O erro absoluto médio descrito na Equação 3.2, medido em m/s , possui valores menores quando comparados aos do MSE e indica o quão diferente são os valores de velocidades entre um modelo estimado e a sua versão de referência, i.e., caso o MAE resulte em, por exemplo, 100, significa que a saída possui um erro absoluto médio de 100 m/s quando comparado com o modelo referência.

$$r = \frac{\sum_{i=1}^n (y_i - \bar{y})(\hat{y}_i - \bar{\hat{y}})}{\sqrt{\sum_{i=1}^n (y_i - \bar{y})^2} \sqrt{\sum_{i=1}^n (\hat{y}_i - \bar{\hat{y}})^2}} \quad (3.3)$$

O coeficiente de correlação de Pearson (r) descrito na Equação 3.3 quantifica a relação linear entre um modelo de velocidades estimado e sua contrapartida verdadeira, no qual o valor de -1 significa correlações opostas, 0 indica não correlação e 1 indica total correlação.

$$R^2 = \frac{[\sum_{i=1}^n (y_i - \bar{y})(\hat{y}_i - \bar{\hat{y}})]^2}{\sum_{i=1}^n (y_i - \bar{y})^2 \sum_{i=1}^n (\hat{y}_i - \bar{\hat{y}})^2} \quad (3.4)$$

O coeficiente de determinação (R^2) é descrito na Equação 3.4 e indica o quão bom um modelo de velocidades estimado é ao o compararmos a um modelo base, já que o valor deste coeficiente indica o quanto o modelo de *deep learning* se ajusta às amostras que lhes foram apresentadas.

$$fac_{2i} = \begin{cases} 1, & \text{se } 0,5 \leq \frac{\hat{y}_i}{y_i} \leq 2 \\ 0, & \text{c.c} \end{cases} \quad (3.5)$$

$$fac_2 = \frac{1}{n} \sum_{i=1}^n fac_{2i} \quad (3.6)$$

O fator de dois (fac_2) descreve quanto dos valores presentes na estimacão realizada pela FCN pode ser considerado como um valor fora do padrão esperado de acordo com o modelo verdadeiro. Isto é feito contando-se quantos elementos da divisão $\frac{\hat{y}_i}{y_i}$ estão dentro do intervalo $[0,5; 2]$ conforme as Equações 3.5 e 3.6.

O objetivo é conseguir aproximar as métricas MSE e MAE o máximo possível de 0 ao passo que as métricas R^2 , r e fac_2 devem estar o mais próximo possível de 1.

4 Resultados e Discussões

Os modelos de velocidades gerados neste experimento representam uma seção de subsuperfícies de 3 km de profundidade (eixo z) por 9,2km de comprimento (eixo x). Utilizou-se uma representação em *grid* de 150 células tanto em z (n_z), quanto em x (n_x), fazendo, assim, com que os intervalos entre as amostras sejam respectivamente $\Delta d_z = 20$ e $\Delta d_x = 61,5$ metros.

Os sismogramas foram gerados utilizando uma composição de 50 fontes (n_s) igualmente espaçadas 184,5 metros umas das outras no eixo x e 150 receptores espaçados 61,5 metros entre si e posicionados no mesmo eixo das fontes ($z = 0$). O número de amostras no domínio do tempo é 1500 (nt) com um intervalo de amostragem de 2ms (Δdt), totalizando 3 segundos de amostragem.

Dos 1020 modelos de velocidades utilizados, 1000 foram utilizados para compor a etapa de treinamento e os 20 restantes foram guardados para utilização na etapa de teste de modelo. A U-Net foi implementada seguindo o esquema indicado na Figura 18 utilizando a função de ativação ReLU e treinada durante 200 épocas utilizando um *batch* de 5 amostras em um nó computacional contendo duas GPUs NVIDIA Tesla P100-SXM2 com 16 GB de memória RAM cada. Dos modelos utilizados durante o treinamento, 20% deles foram reservados para a etapa de validação, que ocorre ao final de cada época de treinamento, e análise de *overfitting* e *underfitting* do modelo.

Monitorou-se o processo de treinamento da U-Net a cada duas épocas por meio da estimação de um dos modelos pertencentes ao conjunto de dados separados para teste. Este monitoramento restringe-se a um acompanhamento visual de como o modelo de velocidades passa a ser formado conforme as épocas avançam, não influenciando, portanto, a forma como a rede aprende a realizar a inversão.

Assim, nota-se na Figura 22 que, apesar de o resultado de inversão da primeira época aparentar conter apenas a lâmina d'água, o modelo da U-net é capaz de, gradualmente, identificar, não apenas a posição inicial e final de mais camadas, mas também suas inclinações e ondulações. Não obstante, dois problemas podem ser prematuramente observados: ao final do treinamento as estruturas de falhas, tampouco o início e final de algumas camadas, puderam ser corretamente identificadas.

A U-Net demorou um total de 18 horas para treinar todas as 200 épocas, mas a estimação dos 20 modelos de velocidades a partir de seus sismogramas demorou, aproximadamente, 16 segundos. Isto significa que, uma vez que o dado sísmico for adquirido, um modelo de velocidades inicial pode ser obtido em menos de um segundo. A Figura 21 mostra como a função de erro decai a cada época de treinamento. Além disso, é possível

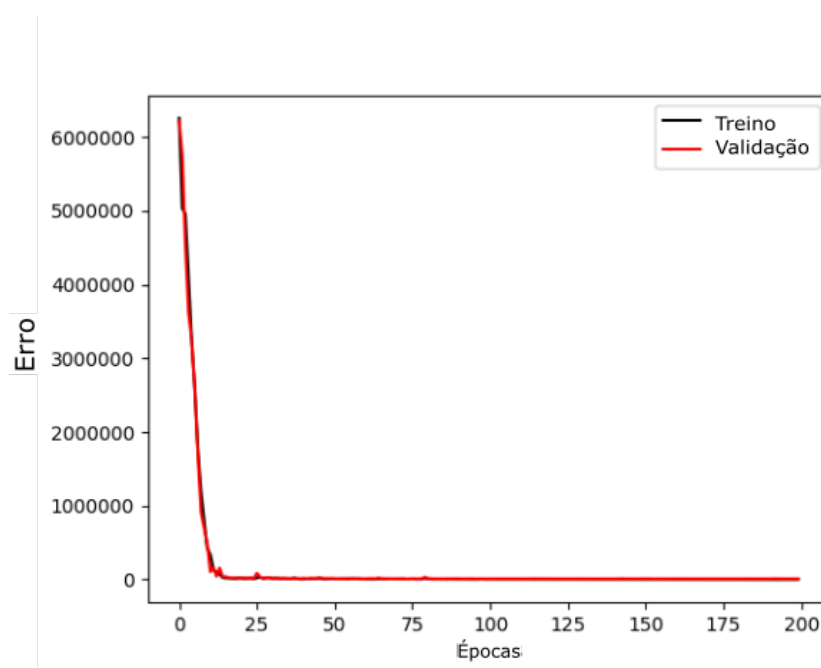


Figura 21 – Monitoramento e comparação do decaimento da função de erro para a etapa de treinamento e validação a cada época

também visualizar que o aprendizado da U-Net não é afetado por *overfitting*, já que a curva validação está praticamente pareada com a curva de treinamento. Na Figura 23 são mostrados dois exemplos de modelos estimados que podem auxiliar a análise dos problemas apontados anteriormente. Além disso, a Tabela 1 indica as métricas de avaliação da U-Net totalmente treinada nos 20 modelos separados para teste.

Percebe-se que o modelo obteve um valor relativamente baixo de 65,5 para o MAE, significando que os modelos estimados diferem, em média, em 65,5 m/s de suas versões verdadeiras. Ademais, valores satisfatórios de R^2 e correlação de Pearson (r) e Fator de 2 foram obtidos, já que os dois primeiros estão próximos a 1 e o último é exatamente 1. Todavia, o alto valor de MSE justifica a falta de precisão ao determinar o início e o fim de camadas inclinadas (conforme visto nos modelos da Figura 23(b) e 23(a)) e ao identificar estruturas de falhas (modelos na Figura 23(d) e 23(c)).

Métrica	Valor
MSE	10188,0654
MAE	65,5954
r	0,9840
R^2	0,9683
Fator de 2	1,0000

Tabela 1 – Métricas de avaliação para a U-Net

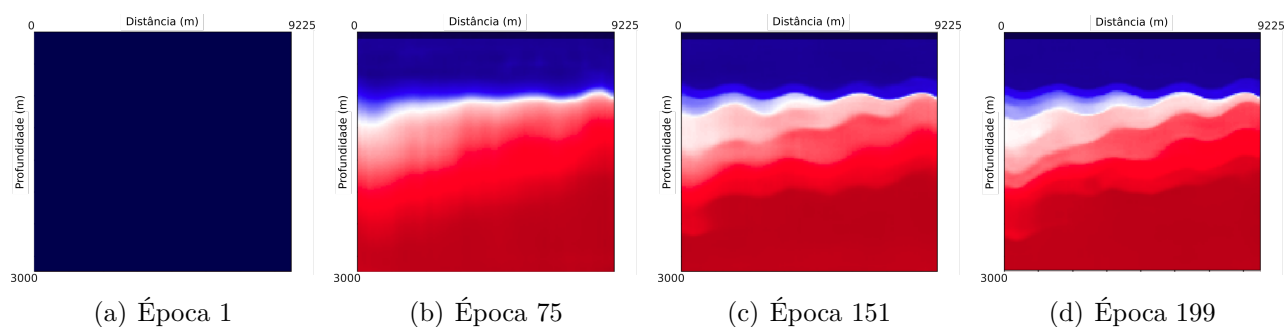


Figura 22 – Saída estimada durante as épocas de treinamento, respectivamente, a) 1, b) 75, c) 151 e d) 199

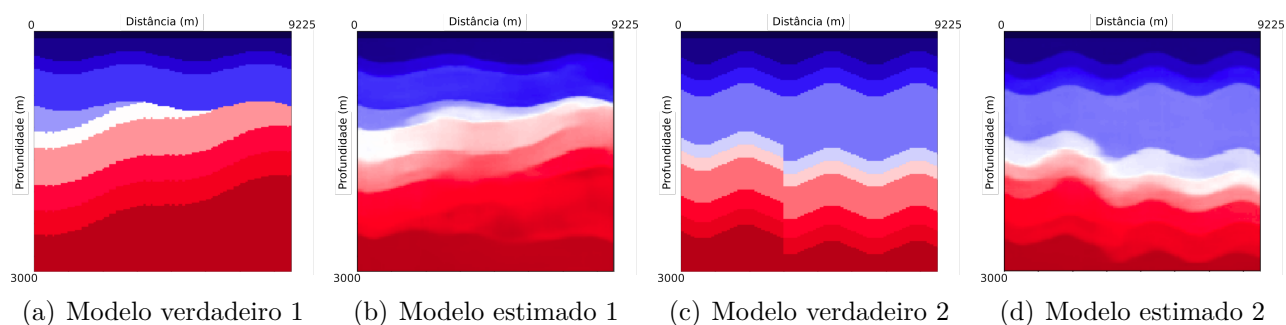


Figura 23 – Resultado da estimação de dois modelos de velocidades em comparação com suas versões verdadeiras existentes no conjunto de teste. a) O primeiro modelo contém camadas com ondulações e inclinações, b) o segundo modelo possui camadas onduladas e com falhas

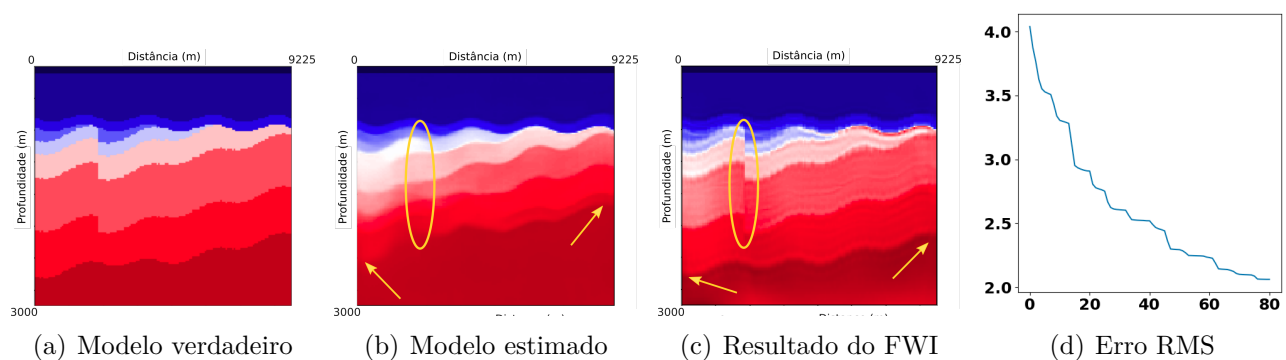


Figura 24 – Comparação do (a) modelo verdadeiro, (b) modelo estimado, (c) modelo após uso do método FWI e (d) taxa de decrescimento do erro RMS a cada iteração do FWI

A etapa final deste experimento consiste em utilizar um dos modelos estimados como entrada para uma implementação da FWI multiescala em frequência. A avaliação do FWI é feita calculando-se o erro RMS entre dado sísmico resultante do modelo de velocidades gerado a cada iteração do algoritmo e o dado sísmico observado. O mesmo modelo usado durante o monitoramento do treinamento da FCN foi utilizado. Conforme notou-se anteriormente, o modelo estimado não contém as estruturas de falhas e algumas

camadas estão posicionadas incorretamente. Ambos os problemas são identificados na Figura 24(b) por uma elipse e duas setas, respectivamente. No entanto, espera-se que estes problemas ocorram, já que propõe-se a estimação de modelos iniciais de velocidade pela FCN, ao passo que almeja-se, com o uso do FWI, corrigir estes problemas e gerar um modelo final da subsuperfície.

Vê-se na Figura 24(d) que, de fato, o modelo inicial estimado é satisfatório devido ao baixo erro RMS calculado no início do processo do FWI. Outrossim, o FWI conseguiu melhorar ainda mais o modelo de velocidades ao identificar as estruturas de falhas que antes estavam ausentes e ao ajustar o posicionamento de algumas camadas. Identifica-se estas melhorias por meio de uma elipse e setas na Figura 24(c) e pelo decréscimo do erro RMS na Figura 24(d).

4.1 Variação nos Parâmetros de Modelagem Sísmica

Os experimentos descritos nesta seção consideram uma única configuração de modelo de velocidades, mas diferentes arranjos para a modelagem sísmica. Diferentemente do experimento anterior, o modelo de velocidades foi modificado para representar uma subsuperfície de 3km de profundidade por 3km de comprimento, mas a representação de um grid de $(x,z) = (150,150)$ foi mantida. Ainda assim, foi necessário ajustar o espaçamento entre as células do *grid*, $\Delta d_x = 20\text{m}$ e $\Delta d_z = 20\text{m}$, para se adequaram ao novo tamanho da subsuperfície. Com relação à modelagem sísmica, houve variação em dois parâmetros: número de fontes e valor da frequência de pico. Os experimentos consideram 3 valores de frequência de pico (4, 8 e 16Hz) que se adéquam à Equação 2.4, e os seguintes arranjos de fontes:

1. 1 fonte posicionada no centro do eixo x do modelo e $z = 0$
2. 10 fontes posicionadas 200 metros uns dos outros no eixo x e $z = 0$
3. 25 fontes posicionadas 120 metros uns dos outros no eixo x e $z = 0$
4. 50 fontes posicionadas 60 metros uns dos outros no eixo x e $z = 0$

Alterou-se também a proporção de modelos de velocidades e dados sísmicos utilizadas para treinamento e teste da rede. Anteriormente, apenas 20 modelos, dos 1020 criados, foram utilizados para teste. Entretanto, esta quantidade de dados poderia ocasionar em uma análise enviesada dos resultados obtidos com a U-Net, já que as métricas estatísticas seriam computadas em um universo muito reduzido do conjunto de dados, onde poderia não existir muita variabilidade. Como forma de mitigar isto, resolveu-se utilizar 80%

do conjunto de dados, cerca de 816 modelos, para treinamento e os 20% restantes (204 modelos) foram separadas unicamente para teste. Dos 816 modelos de treinamento, 20% foram reservados para validação do modelo a cada época (aproximadamente 163 modelos). Assim, a análise torna-se mais robusta e capaz de determinar se a U-Net está generalizando bem ou não a partir dos dados apresentados.

A estratégia adotada para estes experimentos fixa a frequência de pico em 4Hz quando há variação no número de fontes e fixa o número de fontes em 25 para avaliar a variação da frequência de pico. Assim como no experimento anterior, compara-se os resultados graficamente e estatisticamente com base nas mesmas métricas após o treinamento completo da U-Net no conjunto de dados separado para teste. A Figura 25 ilustra a processo de decaimento da curva da função de erro a cada época de treinamento e validação. Nota-se que, apesar de o decaimento da função de erro na Figura 25(a) ser bastante irregular, já que existem diversos picos durante a validação da época, todas as curvas de validação sobrepuseram as de treinamento, indicando a ausência de *overfitting* ou *underfitting* no aprendizado da U-Net.

Os resultados gráficos da estimação de um único modelo base pertencente ao conjunto de dados de teste pode ser visto na Figura 26. Este modelo base possui camadas inclinadas e onduladas e uma estrutura simples de falha que pode ser identificada pela elipse amarela na Figura 26(a). Uma análise direta das imagens leva à conclusão de que as Figuras 26(c), 26(f) e 26(g) são as melhores estimações para o modelo base utilizado já que além de conterem as camadas bem posicionadas, com identificação das ondulações e inclinação e alta precisão das velocidades em cada camada (assim como nos outros modelos), também possuem uma representação razoável da estrutura de falha.

De todo modo, não é seguro inferir tão diretamente que estas representações são as melhores porque o modelo de velocidades em questão representa apenas um exemplo em um vasto conjunto de dados de teste. Talvez seja o caso de a FCN ter de fato estimado um modelo de alta resolução a partir dos dados sísmicos utilizado para treinamento, mas também podem existir casos nos quais o modelo estimado difere bastante de seus modelos-base.

Uma associação entre os experimentos e seu tempo computacional é conduzido antes de proceder-se com a análise estatística dos resultados. Nota-se na Figura 27(a) que conforme aumenta-se o número de fontes, o custo computacional para treinar a U-Net também aumenta. Entretanto, o tempo de treinamento é muito pouco afetado com as mudanças na frequência de pico. Isto ocorre porque o número de fontes influencia diretamente no tamanho do dado sísmico, onde fontes extras significam a adição de matrizes de tamanho $n_t \times n_x$ ao dado, onde n_t e n_x são, respectivamente, o número de

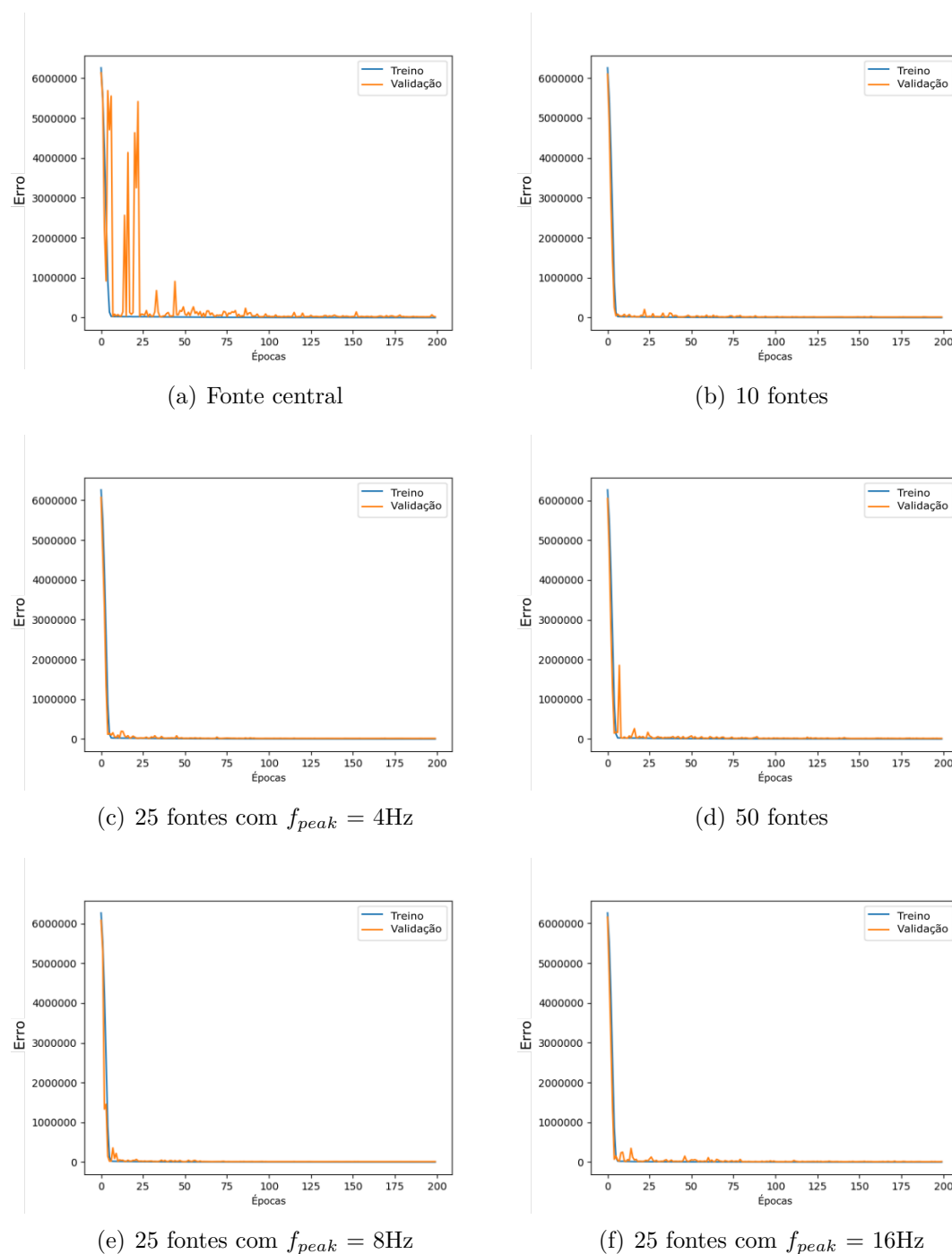


Figura 25 – Ilustração do decaimento da função de erro a cada época de treinamento e validação da U-Net para os dados sísmicos gerados com uma fonte central (a), 10 fontes (b), 25 fontes e 4Hz (c), 50 fontes (d), 25 fontes e 8Hz (e) e 25 fontes e 16Hz (f).

amostras no tempo e no eixo horizontal definidos para modelagem.

Como a frequência de pico impacta na resolução do dado sísmico, não há adição de matrizes, apenas modificação nos valores numéricos das matrizes já existentes. Assim, sua influência no custo computacional parece ser arbitrária, já que nos experimentos realizados

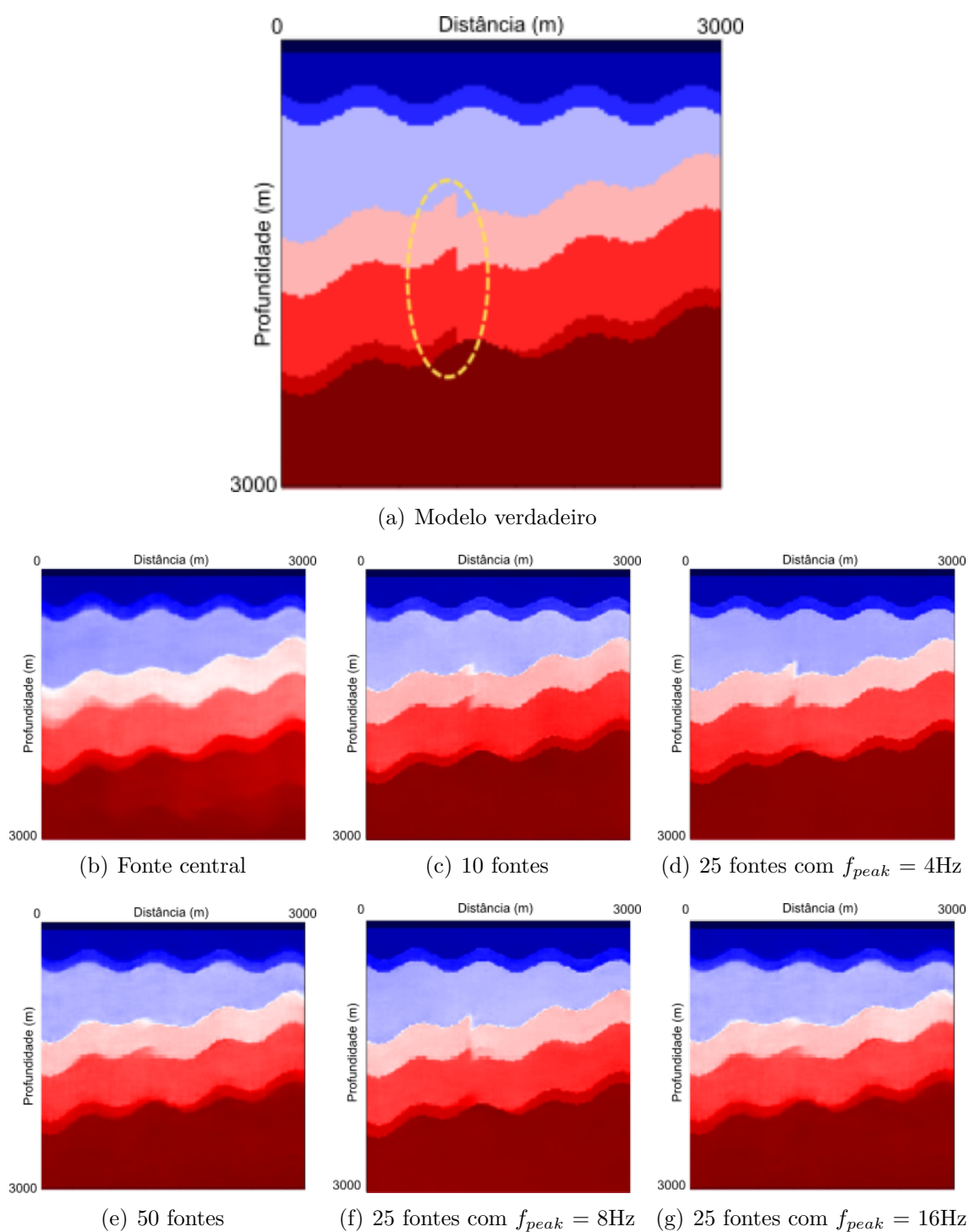


Figura 26 – Comparação qualitativa entre o modelo verdadeiro (a) e os modelos estimados por diferentes instâncias da U-Net, as quais possuem a mesma arquitetura, mas foram treinadas com dados sísmicos gerados a partir de diferentes configurações. Primeiro, foi fixado o valor de $f_{peak} = 4\text{Hz}$ e variou-se as fontes em fonte posicionada no centro do modelo (b), 10 fontes (c), 25 fontes (d) e 50 fontes (e). Em seguida o número de fontes foi fixado em 25 e f_{peak} foi variado em 8Hz (f) e 16Hz (g).

o que obteve menor tempo de treinamento foi aquele com maior frequência de pico (16Hz), enquanto que a frequência de pico de 8Hz obteve o maior tempo de treinamento.

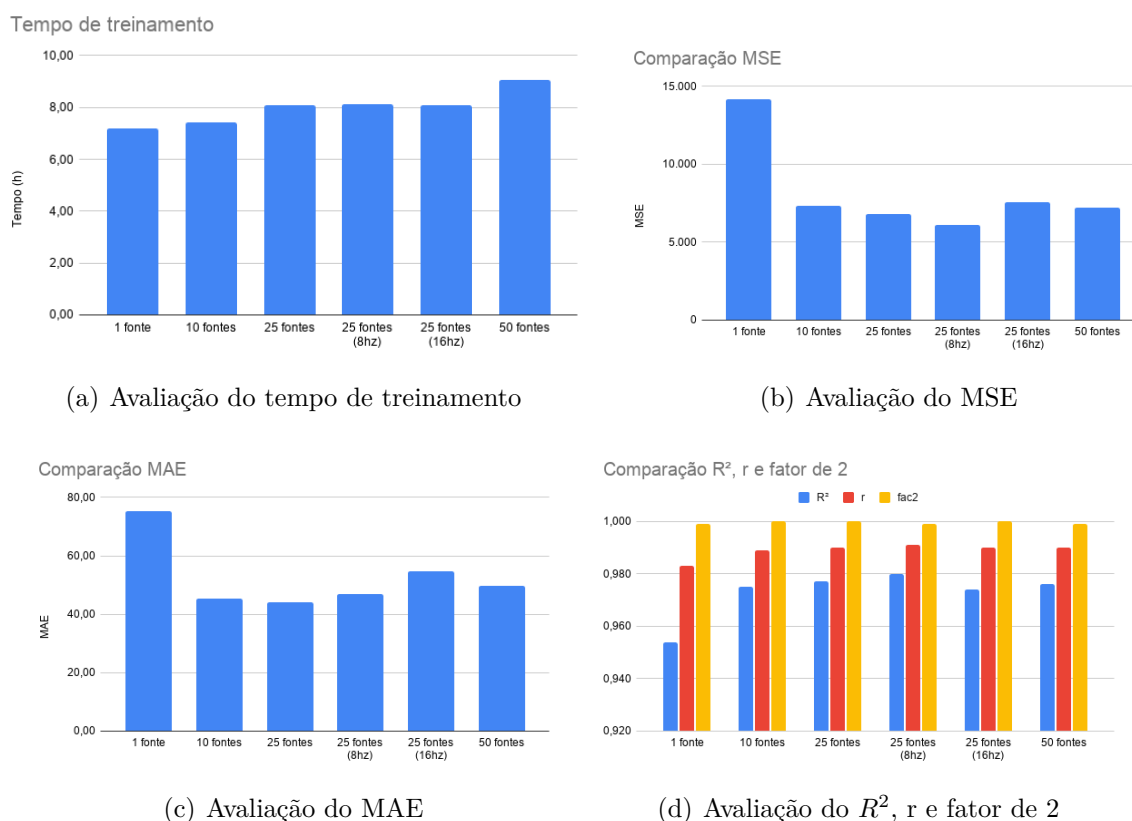


Figura 27 – Avaliação do tempo de treinamento (a) e avaliação estatística de cada experimento considerando as métricas MSE (b), MAE (c) e correlação de Pearson, coeficiente de determinação R^2 e fator de 2 (d).

As Figuras 27(b), 27(c) e 27(d) apresentam respectivamente os resultados obtidos com as métricas MSE, MAE e R^2 , r e fator de 2 para cada experimento. A avaliação destas métricas mostra que elas, em geral, alcançaram valores próximos, mas é possível analisar de antemão que o treinamento da U-Net com dados sísmicos que possuem um número maior de fontes não indica necessariamente uma estimativa melhor em comparação as outros. Mesmo o experimento com 50 fontes apresentando bons resultados, obteve-se melhores métricas usando dados sísmicos com outras quantidades de fontes, por exemplo ao usar-se dados sísmicos de 10 e 25 fontes. Com ambos as métricas estatísticas foram melhores que na configuração de 50 fontes, sendo que a configuração de 25 obteve resultados ainda melhores que a configuração de 10 fontes.

Além disso, foi possível melhorar ainda mais os resultados dos experimentos com 25 fontes ao modificarmos a frequência de pico de 4Hz para 8Hz, alcançando o menor valor para a função de erro (MSE) e o mais alto valor de R^2 e r dentre todos os experimentos.

Isto, todavia, aconteceu à custa de piorar levemente o resultado do MAE e do fator de dois em comparação com o experimento de 25 fontes e 4Hz.

Por outro lado, a diminuição do número de fontes, tampouco um outro aumento na frequência de pico foram traduzidos em melhoria na estimação do modelo de velocidades. Os piores resultados pertencem ao experimento com uma única fonte central. Neste caso, os valores de R^2 , r e fator de dois, apesar de mostrar pequenas diferenças em comparação com os outros experimentos, foram os menores e o MSE e MAE os maiores dentre todos. Este resultado está, certamente, relacionado à irregularidade do decaimento da função de erro ilustrada na Figura 25(a), já que os resultados obtidos no processo de validação podem ser refletidos no processo de teste do modelo *deep learning* já que ambos possuem modelos não utilizados durante o processo de treinamento. Ademais, o experimento com 16Hz resultou em métricas piores do que o experimento que utiliza 50 fontes.

Mesmo possuindo as piores métricas, os dados sísmicos com a fonte central foram invertidos com sucesso pela U-Net em um modelo de velocidades aceitável. Isto provavelmente aconteceu devido ao tamanho da subsuperfície e do modelo de velocidades utilizados, já que são considerados pequenos de uma perspectiva geofísica. Entretanto, não se pode confirmar que isto se repita conforme a subsuperfície se torna maior apenas com as análises deste trabalho.

4.2 Otimização de Hiperparâmetros

Os modelos de velocidades utilizados neste experimento são os mesmos do experimento anterior. Entretanto, apenas os dados sísmicos gerados pela configuração de 25 fontes e frequência de pico de 8Hz são consideradas aqui. Assim, este experimento possui dois propósitos: buscar formas de melhorar os modelos estimados pela U-Net e analisar como modificações em dois de seus hiperparâmetros (função de ativação e otimizador) influenciam no treinamento da rede e, conseqüentemente, na estimação do modelo de velocidades.

Os experimentos foram executados com a combinação de três otimizadores (RMS-prop, Adam e Adamax) e 4 funções de ativação (ReLU, Leaky ReLU, ELU e Parametric ReLU). Sendo assim, as 4 funções de ativação são utilizadas em conjunto com cada um dos 3 otimizadores. Os resultados, assim como nos demais experimentos, são apresentados de forma gráfica a partir de um modelo base e estatisticamente com base nas métricas já apresentadas. Além dessa análise, compara-se também a taxa de decaimento da função de erro para cada combinação de otimizador e função de ativação. A Figura 28 exhibe tal comparação. Percebe-se, em geral, que o processo de treinamento não foi afetado por *overfitting*, tampouco por *underfitting*, já que a curva de validação permanece rente à curva

de treinamento durante a maior parte das épocas, ainda que haja casos onde o valor da função de erro oscile consideravelmente, como nos picos identificados nas Figuras 28(e), 28(f), 28(g), 28(h), 28(i), 28(j) e 28(k).

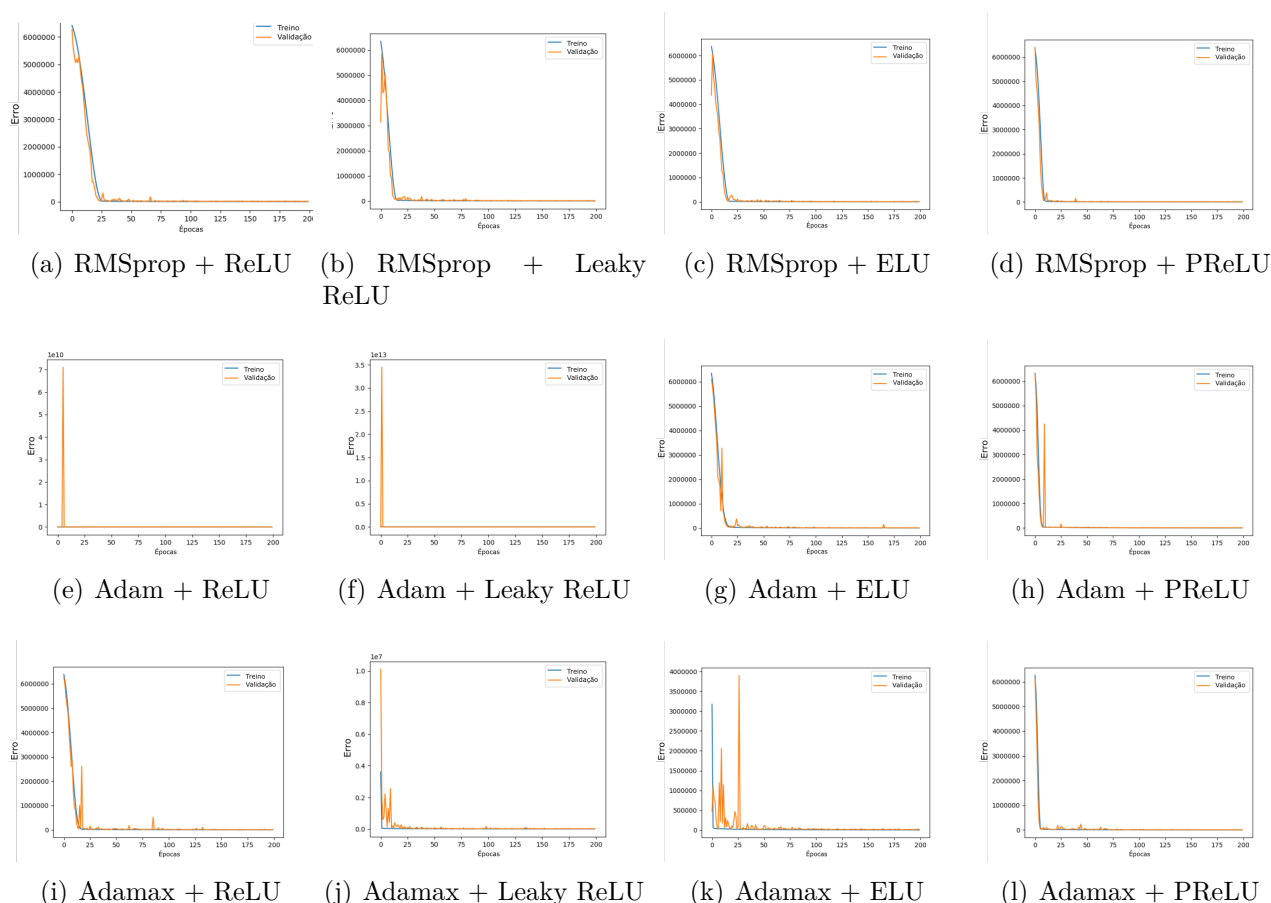
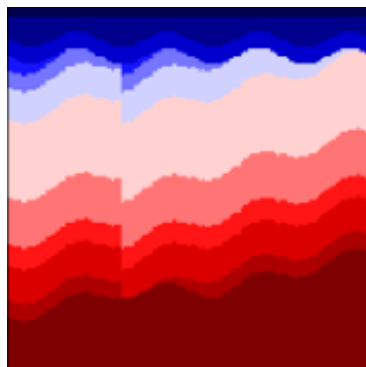


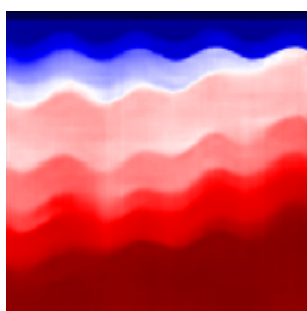
Figura 28 – Comparação do decaimento da função de erro época a época durante treinamento e validação da U-Net -Net ao modificar o otimizador e a função ativação para RMSprop + ReLU (a), RMSprop + Leaky ReLU (b), RMSprop + ELU (c), RMSprop + PReLU (d), Adam + ReLU (e), Adam + Leaky ReLU (f), Adam + ELU (g), Adam + PReLU (h), Adamax + ReLU (i), Adamax + Leaky ReLU (j), Adamax + ELU (k) e Adamax + PReLU (l).

A Figura 29 demonstra o mesmo modelo de velocidades estimado pela U-Net treinada com diferentes combinações de função de ativação e otimizador e como estes resultados se comparam ao modelo base. É notável como as diferentes combinações influenciam diretamente o resultado final da rede. Em alguns casos, o modelo estimado aparenta ser suavizado, onde as estruturas de algumas camadas não podem ser vistas claramente, mesmo sendo possível determinar que a extensão dos valores de velocidade são compatíveis com o modelo base.

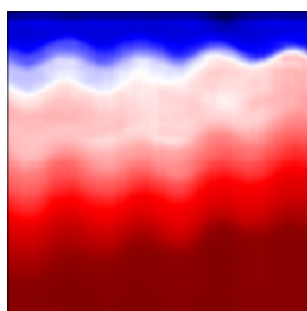
Existem ainda dois outros casos: um intermediário que engloba modelos de velocidades ainda suavizados, mas com uma divisão mais clara de suas camadas; e outro no qual enquadram-se os modelos mais detalhados. De acordo com a Figura 29, os modelos



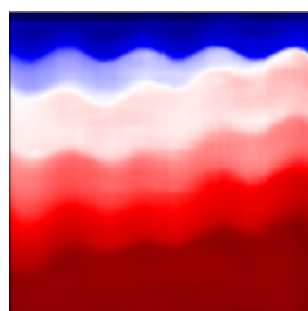
(a) Modelo verdadeiro



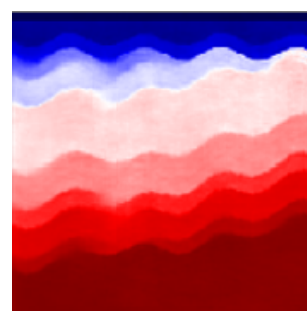
(b) RMSprop + ReLU



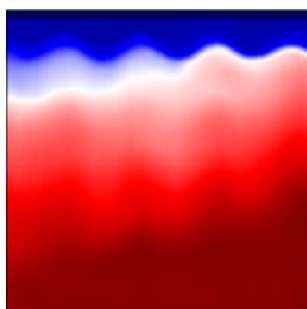
(c) RMSprop + Leaky ReLU



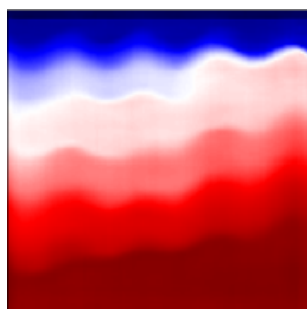
(d) RMSprop + ELU



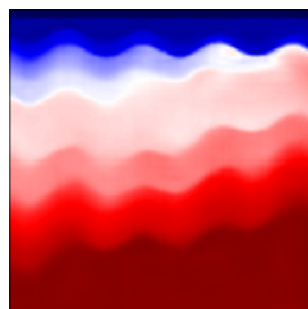
(e) RMSprop + PReLU



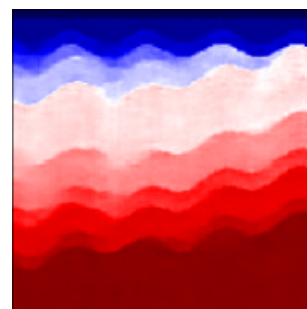
(f) Adam + ReLU



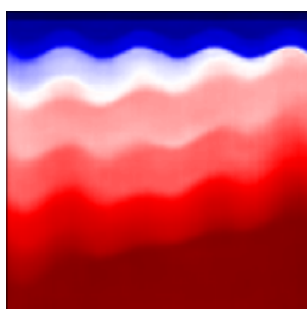
(g) Adam + Leaky ReLU



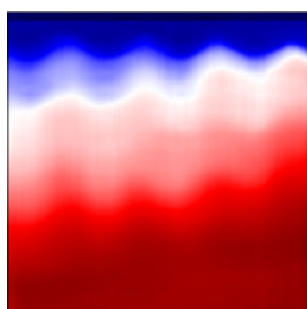
(h) Adam + ELU



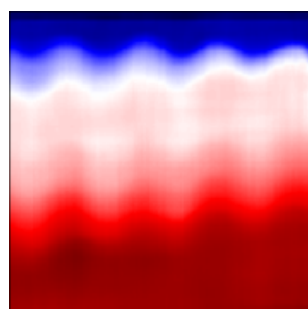
(i) Adam + PReLU



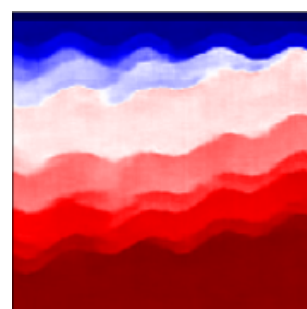
(j) Adamax + ReLU



(k) Adamax + Leaky ReLU



(l) Adamax + ELU



(m) Adamax + PReLU

Figura 29 – Comparação do modelo verdadeiro (a) com os modelos de velocidades estimados pela U-Net ao modificar o otimizador e a função ativação para RMSprop + ReLU (b), RMSprop + Leaky ReLU (c), RMSprop + ELU (d), RMSprop + PReLU (e), Adam + ReLU (f), Adam + Leaky ReLU (g), Adam + ELU (h), Adam + PReLU (i), Adamax + ReLU (j), Adamax + Leaky ReLU (k), Adamax + ELU (l) e Adamax + PReLU (m).

em que é possível visualizar-se melhor as camadas, i.e., os que possuem mais detalhes, são aqueles resultantes do uso da função de ativação PReLU (Figuras 29(e), 29(i) e 29(m)), já a representação intermediária e modelos suavizados variam com as outras funções de ativação. Por um lado, a função PReLU aparenta ser menos sensível à mudança no otimizador, o que possivelmente explica a geração de modelos de alta resolução ao longo de todos os experimentos. Por outro lado, como cada uma das funções de ativação ora gera representações intermediárias, ora representações suaves, é plausível dizer que o otimizador é determinante para a resolução do modelo.

Estendendo-se a análise para todos os modelos de teste no conjunto de dados, a Figura 30 apresenta cada uma das métricas computadas para cada uma das combinações de otimizadores e funções de ativação retratadas em gráficos de barra. Uma combinação ideal possui os menores valores para MSE e MAE e os maiores valores para as outras três métricas. Entretanto, existe uma certa tolerância a esta regra dependendo de quanto os valores das métricas diferem uns dos outros, i.e., se a combinação possui os melhores valores em três métricas e as outras duas diferem levemente dos melhores encontrados, então esta combinação ainda pode ser considerada como a melhor encontrada.

Antes de indicar, porém, a melhor combinação, é importante associar os resultados gráficos e estatísticos. Na análise anterior, a ativação com o PReLU demonstrou produzir modelos de alta resolução independentemente do uso dos otimizadores RMSprop, Adam ou Adamax. Se a análise estatística considerasse apenas o MSE, a estimação com a PReLU apenas teria os melhores resultados quando utilizada em conjunto com Adamax, já que a combinação de Adam com as outras funções de ativação, RMSprop + Leaky ReLU e RMSprop + ELU obtiveram valores melhores. Entretanto, considerando-se todas as métricas, pode-se verificar que a função PReLU, quando não gera os melhores valores, possui métricas levemente abaixo das melhores encontradas. Isto pode explicar o porquê de esta métrica ser menos sensível à mudança de otimizadores. Além disso, espera-se que os resultados da U-Net treinada com a PReLU sejam os melhores já que esta função, diferentemente das demais, adapta sua saída conforme o treinamento ocorre, provendo uma ativação neural mais robusta.

Os resultados estatísticos da U-Net treinada com a função ELU também demonstram ser interessantes, especificamente quando utilizada com os otimizadores RMSprop ou Adam, onde baixos valores de MSE e MAE foram obtidos em conjunto com valores competitivos para a correlação de Pearson e fator de dois. Já para o caso da ReLU e Leaky ReLU, espera-se que o segundo supere os resultados do primeiro, mas isto só ocorre para o caso do Leaky ReLU + RMSprop. Isto, mais uma vez, justifica a superioridade do PReLU já que o valor constante explicado no Leaky ReLU pode influenciar negativamente seus resultados. Assim, é seguro afirmar que a melhor combinação de função de ativação



Figura 30 – Avaliação estatística das combinações das função de ativações e otimizadores para o (a) MSE, (b) MAE, (c) correlação de Pearson, (d) coeficiente de determinação R^2 e (e) fator de 2

e otimizador, com base tanto em resultados gráficos, quanto estatísticos, é PReLU + Adamax.

4.3 Redução de Dimensionalidade

Este experimento tem como objetivo discutir e analisar a aplicação de dois métodos para redução de dimensionalidade de dados sísmicos: 1) *principal component analysis* (PCA) e 2) *autoencoders* convencionais. Uma terceira opção de técnica de redução de

dimensionalidade seria a utilização de *variational autoencoders* (VAEs), a qual pode obter melhores resultados que os *autoencoders* padrões. Esta técnica, porém, não foi utilizada devido à natureza dos dados comprimidos que ela gera.

Conforme dito anteriormente, um VAE representa o dado de entrada em dimensão reduzida por meio de parâmetros de média e variância com base em uma distribuição estatística. Entretanto, as características de um dado sísmico são importantes para a estimação do modelo de velocidades e mantê-las traz significado físico para o modelo de inversão com base em *deep learning*. No decorrer desta seção será mostrado que isto pode ser obtido com o *autoencoder*, mas não com o PCA. Ainda assim, decidiu-se pela utilização do PCA por ser um modelo amplamente utilizado para redução de dimensionalidade de dados e para servir como uma base comparativa com os dados reduzidos pelo *autoencoder* em uma etapa seguinte do experimento.

Separou-se 60 dados sísmicos modelados com 25 fontes e frequência de pico de 8Hz para compor o conjunto de dados para redução de dimensionalidade. A princípio, 60 dados sísmicos é um valor pequeno para o compor o conjunto de dados. Entretanto, a redução é realizada tiro a tiro, em vez de realizada no dado sísmico por completo. Sendo assim, como cada modelo possui 25 tiros, o conjunto de dados possui um total de 1500 amostras. Assim sendo, o conjunto de dados possui a seguinte forma $(n_{amostras}, n_t, n_x) = (1500, 1500, 150)$, ou seja, é uma estrutura tri-dimensional, onde cada amostra é um dado bi-dimensional de $(n_t, n_x) = 1500 \times 150$. A redução de dimensionalidade tem o objetivo de reduzir o dado sísmico de $(n_t, n_x) = 1500 \times 150$ para uma representação de $(n_t, n_x) = 50 \times 5$, ou seja, uma redução de aproximadamente 99,89% no tamanho do dado.

A implementação do algoritmo de PCA utilizada espera que o conjunto de dados possua uma composição bi-dimensional, ou seja, seguindo um formato de $(n_{amostras}, n_{características})$. Remodelou-se o conjunto de dados tri-dimensional, portanto, para obedecer esta restrição, transformando-o em $(n_{amostras}, n_{características}) = (1500, 225000)$. Percebe-se que este procedimento por si só já altera a estrutura física do dado sísmico, já que cada amostra em n_t é concatenada sequencialmente, fazendo com o que o dado perca um pouco sua representação temporal. Outra alternativa, mas que acarretaria também em alteração na estrutura física, seria transpor o dado sísmico antes de remodelá-lo, mantendo a representação temporal ao custo de sacrificar a representação espacial do dado (n_x). Neste experimento apenas o primeiro caso é tratado. Seguindo a ideia anterior para remodelar o dado sísmico, o resultado do PCA também deverá ser de $(n_{amostras}, n_{componentes})$. Assim, o total final é de 250 componentes.

Como a execução do PCA foi realizada com base no total de componentes a ser encontrado, é necessário calcular o quanto de variância estas componentes representam do

conjunto original. O valor encontrado foi de, aproximadamente, 94%, demonstrando que boa parte dos dados originais podem ser representados pelas componentes resultantes da técnica PCA aplicada.

O próximo passo é discutir a redução de dimensionalidade para o *autoencoder*. Neste caso, utiliza-se como conjunto de testes 50% do conjunto de dados original, além de tiros de outros 2 modelos não considerados durante o treinamento. Os dados sísmicos destes dois modelos foram construídos usando 25 fontes e, respectivamente, $f_{peak} = 4\text{Hz}$ e $f_{peak} = 16\text{Hz}$. Tais modelos foram utilizados com o intuito de verificar a generalização do *autoencoder* independente da frequência de pico com qual o dado foi modelado.

A avaliação do Autoencoder pode ser realizada de duas formas: com base no erro quadrático médio (função de *loss* utilizada para treinamento) e no resultado da reconstrução do dado sísmico após o *decoder*. No caso deste trabalho, o foco será na análise qualitativa dos resultados, já que considera-se ser mais importante que a análise pura da função MSE. Isto porque os valores que compõem o dado sísmico são muito pequenos. Logo, o erro pode, também, ter um valor muito pequeno e não condizer com o real desempenho do *autoencoder*.

A Figura 31 mostra o resultado quando um tiro de um dado sísmico é submetido à redução de dimensionalidade, respectivamente, com o *autoencoder* e com o PCA. Nesta imagem fica claro como a alteração da estrutura física do dado sísmico afeta sua redução no PCA. Com o *autoencoder*, no entanto, nota-se que a estrutura geral do tiro central é mantida: a onda propaga-se do meio do modelo, mesmo com a dimensão espacial reduzida em $n_x = 5$.

Uma forma de garantir que esta representação do *autoencoder* é, de fato, fiel, é através da análise da reconstrução pelo módulo *decoder* do *autoencoder*. A Figura 32 mostra o resultado da reconstrução de alguns tiros dos modelos de teste utilizados com o *autoencoder*. Nota-se que a reconstrução do tiro é bastante similar com o original, principalmente para o valor de $f_{peak} = 8\text{Hz}$. Ainda assim mesmo quando o dado possui mais detalhes, i.e., é modelado com alta frequência de pico, ou quando $f_{peak} = 4\text{Hz}$, o dado foi reconstruído possuindo poucos artefatos, considerando que nenhum destes dois tipos de tiros foram utilizados para o treinamento do *autoencoder*. Sendo assim, entende-se que a representação comprimida gerada pelo *encoder* do *autoencoder* é fiel ao dado original.

A próxima etapa realizada no experimento foi o treinamento da U-Net com os dados reduzidos pelas técnicas discutidas na seção anterior. Os dados sísmicos reduzidos foram os mesmos utilizados no experimento de otimização de hiper-parâmetros com 25 fontes e 8Hz. Entretanto, os dados sísmicos agora possuem um tamanho de $(n_t, n_x) = (50, 5)$. Desta forma, compara-se graficamente e através de métricas estatísticas a estimação de

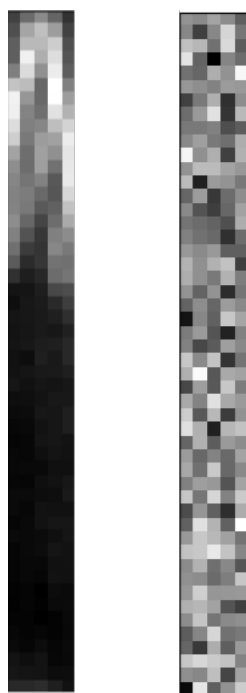


Figura 31 – Resultado da redução de dimensionalidade de um tiro de um dado sísmico com *autoencoder* (tiro à esquerda) e PCA (tiro à direita).

modelos de velocidades a partir de dados sísmicos reduzidos por meio das técnicas PCA e *autoencoder*.

A Figura 33 mostra o mesmo modelo estimado pela U-Net treinada utilizando os dados sísmicos reduzidos gerados pelo PCA (Figura 33(b)) e pelo *autoencoder* (Figura 33(a)). Em geral, observa-se que a U-Net pôde gerar modelos com uma boa resolução. Entretanto, a figura mostra que o resultado do treinamento da U-Net com dados de dimensionalidade reduzida gera modelos estimados de menor resolução que os obtidos com os experimentos anteriores. Ainda assim, os resultados encontrados com os dados gerados pelo *autoencoder* possuem uma resolução relativamente melhor que os gerados pelo PCA.

Através da análise estatística pode-se entender os resultados de maneira mais total. Todas as métricas obtidas com o treinamento da U-Net com os dados gerados pelo *autoencoder* são melhores que as do PCA, com exceção do fator de 2 que possui o mesmo valor nos dois casos. O MSE, por exemplo, função de *loss* utilizada para o treinamento da U-Net, no caso dos dados do *autoencoder* é praticamente metade do que no caso do PCA. O mesmo vale para o MAE. Já nos casos do R^2 e r os valores com os dados do AE foram ligeiramente melhor. As métricas obtidas com a U-Net, tanto para os dados gerados pelo AE, quanto para os dados gerados pelo PCA são vistos na Figura 34.

Apesar dos valores numéricos obtidos com a U-Net serem inferiores àqueles obtidos nos outros experimentos, o resultado geral deste experimento é positivo. Mostrou-se que

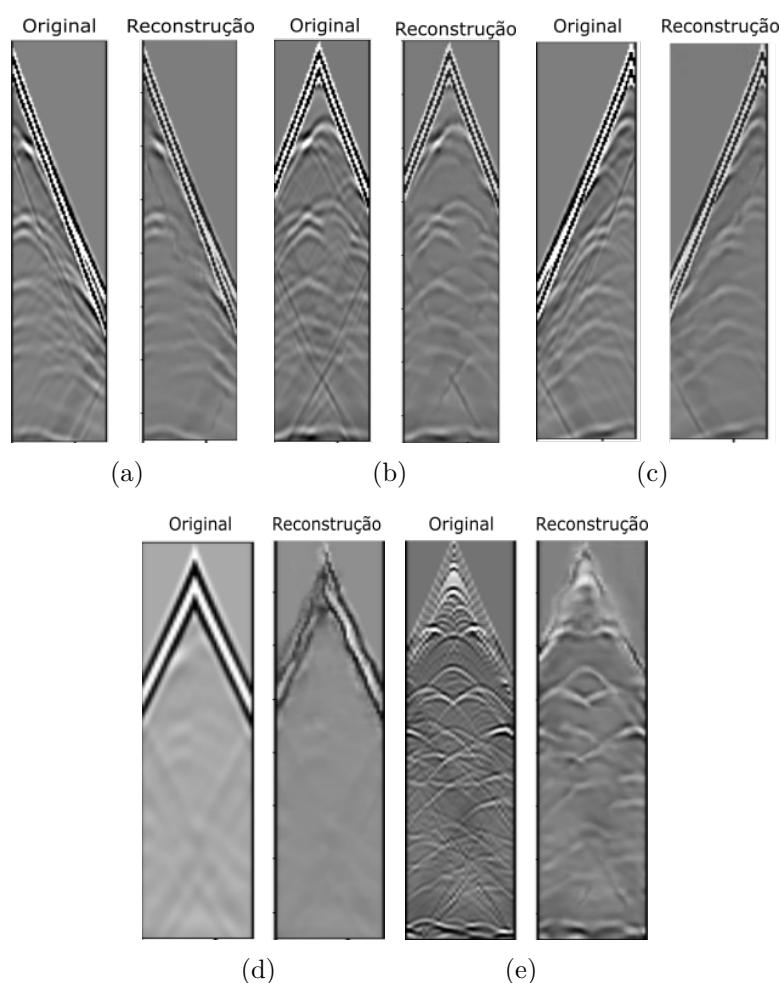


Figura 32 – Tiros originais e reconstruídos de um dado sísmico modelado com $f_{peak} = 8\text{Hz}$ e fontes posicionadas no (a) início, (b) centro e (c) final, (d) tiro de um dado sísmico modelado com $f_{peak} = 4\text{Hz}$ e fonte posicionada no centro e (e) tiro de um dado sísmico modelado com $f_{peak} = 16\text{Hz}$ e fonte posicionada no centro.

técnicas de redução de dimensionalidade podem ser aplicadas direta e individualmente nos tiros que compõem um dado sísmico, não sendo necessário a utilização do dado completo. Isto significa redução do custo computacional para processar os dados e em modelos menos complexos, no caso de modelos de *deep learning*. Além disso, mostrou-se que manter o aspecto físico do tiro de um dado sísmico é um fator importante para se obter uma melhor estimativa de modelos de velocidades com a U-Net. Em suma, este experimento demonstrou com sucesso como dados sísmicos podem ter sua dimensão reduzida por meio de técnicas como *autoencoder* e PCA e como mesmo com esta redução um modelo de *deep learning*, como a U-Net, pode gerar modelos de boa resolução.

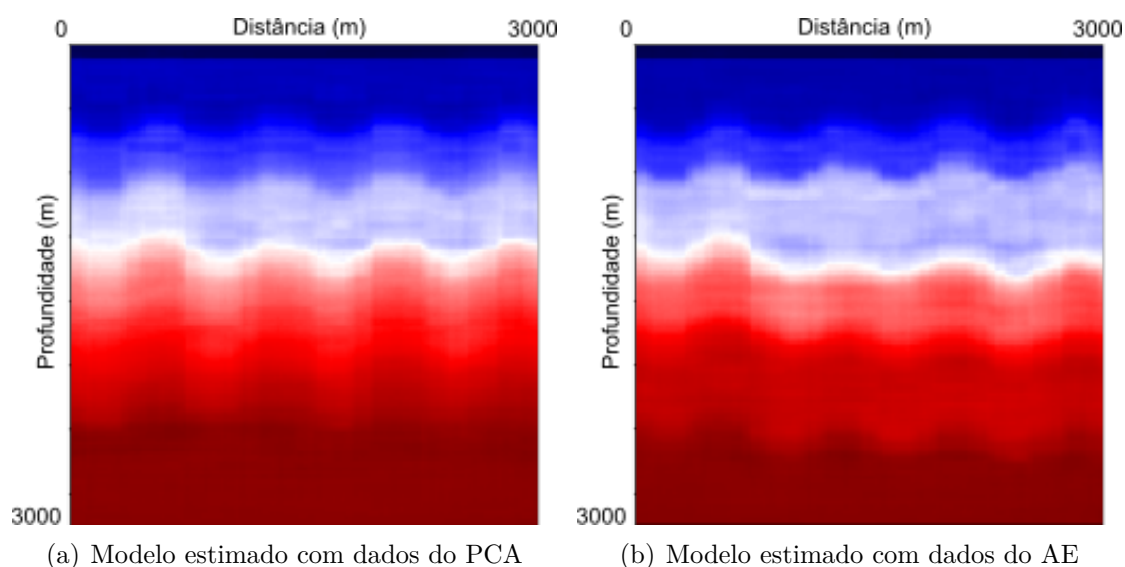
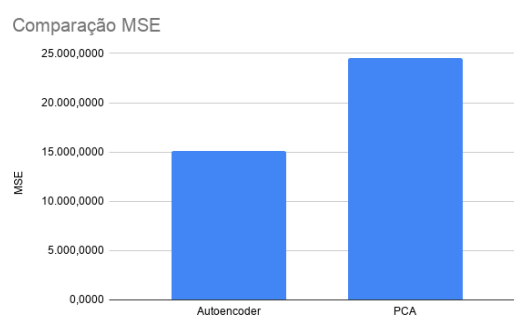
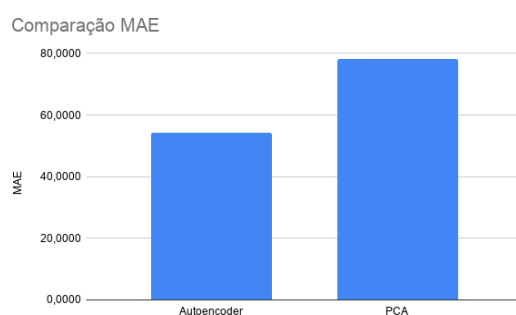


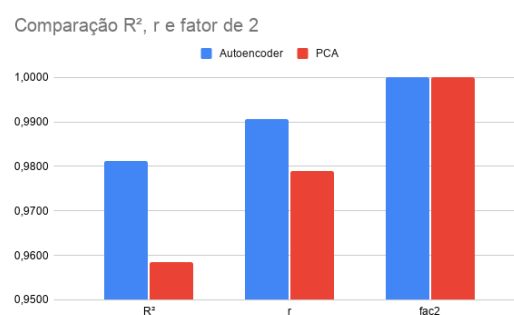
Figura 33 – Resultado dos modelos estimados pela U-Net utilizando os dados sísmicos reduzidos pelo *autoencoder* (a) e PCA (b)



(a) Avaliação do MSE



(b) Avaliação do MAE



(c) Avaliação do R^2 , r e fator de 2

Figura 34 – (a) Avaliação do tempo de treinamento e avaliação estatística de cada experimento do (b) MSE e MAE, (c) correlação de Pearson, coeficiente de determinação R^2 e fator de 2 utilizando dados reduzidos

4.4 Utilização de HyperColumn

O intuito deste experimento é substituir a U-Net por outra *fully convolutional network*, a HyperColumn, com o intuito de validar o uso de *deep learning* para inversão

sísmica e estimação de modelos de velocidades com outro modelo além do U-Net. O modelo de HyperColumn utilizado possui uma constituição similar ao *encoder* da U-Net usada para inversão de dados sísmicos completos (sem redução de dimensionalidade), ou seja, aplica-se duas convoluções seguidas, cada uma com *batch normalization*, antes de aplicar uma operação de *pooling* (representadas neste caso por uma convolução de $\text{stride} = 2$).

Aplica-se primeiro uma convolução de um único filtro com *batch normalization* na saída da operação de *pooling* e em seguida aplica-se uma interpolação bilinear para construir cada *hypercolumn* do modelo. Neste caso, o primeiro *hypercolumn* é gerado a partir a primeira operação de *pooling*, o segundo *hypercolumn* a partir do segundo *pooling* e assim por diante. Ao final, concatena-se todos os *hypercolumns* gerados em uma única camada e utiliza-se uma última camada de convolução com base na saída da camada concatenada para caracterizar a saída final da rede. A Figura 35 apresenta a composição da rede HyperColumn utilizada.

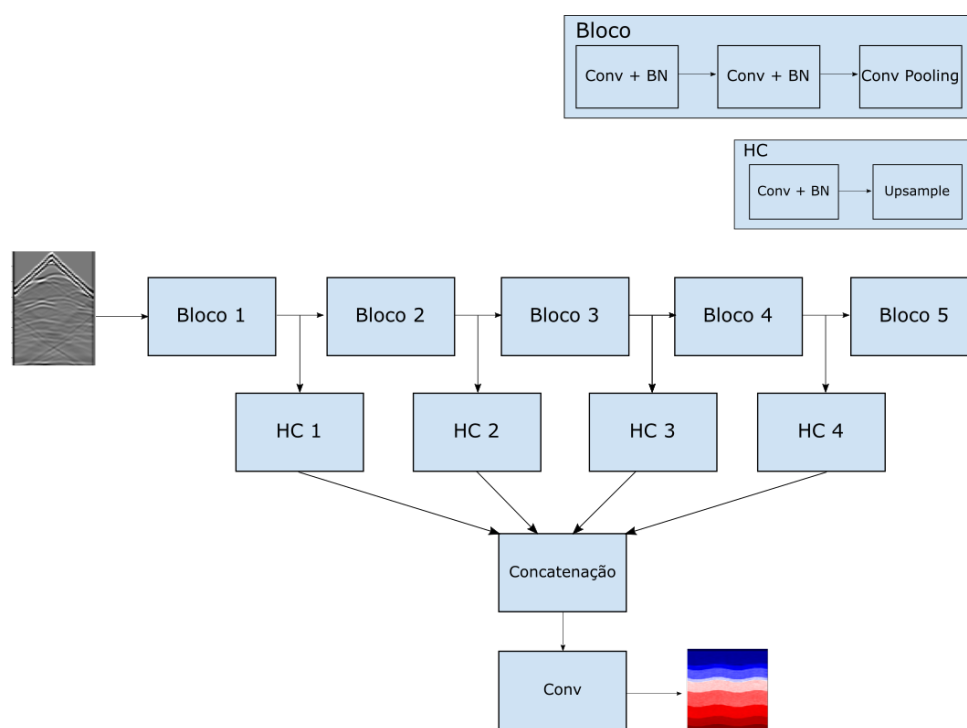


Figura 35 – Representação das camadas utilizadas no modelo de HyperColumn.

Os dados sísmicos e modelos de velocidades utilizados neste experimento foram os mesmos utilizados no experimento de otimização de hiperparâmetros, ou seja, dados sísmicos de 25 fontes e frequência de pico de 8 Hz. Como a HyperColumn utilizada possui menos camadas e é, portanto, menos computacionalmente custosa que a U-Net, decidiu-se treiná-la por 300 épocas em vez das 200 épocas utilizadas nos demais experimentos. Avalia-se o desempenho da HyperColumn para estimação de modelos de velocidades por meio

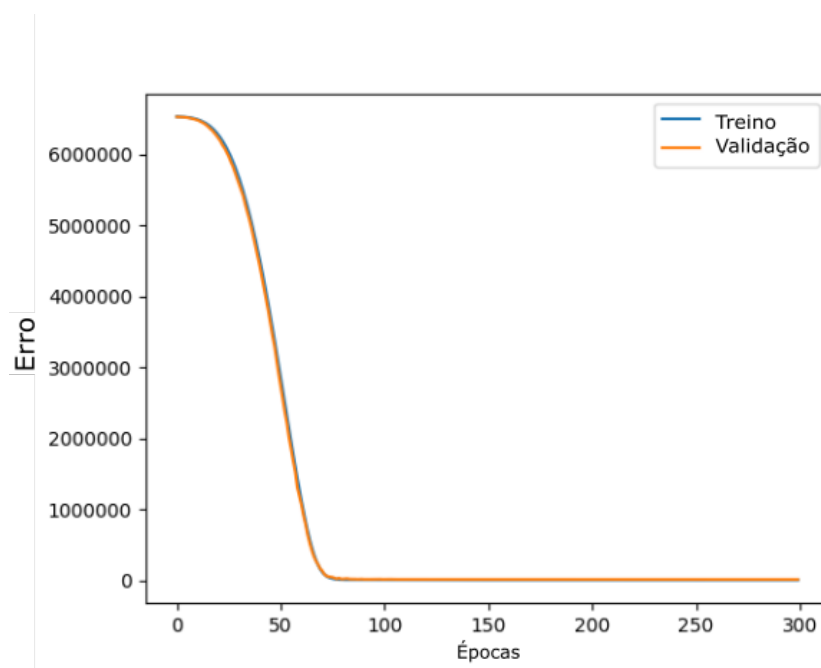


Figura 36 – Monitoramento e comparação do decaimento da função de erro para a etapa de treinamento e validação a cada época do HyperColumn

de seu desempenho computacional, isto é, o tempo necessário para treinar o modelo, por meio da análise qualitativa e quantitativa de seu treinamento e por meio da comparação da curva de decaimento da função de erro durante as etapas de treinamento e validação, assim como foi feito nos experimentos anteriores.

A Figura 36 mostra que, assim como nos experimentos anteriores, as curvas de validação e treinamento permanecem sobrepostas durante todas as épocas de treinamento. Isto significa que o modelo possui uma boa generalização, isto é, os casos de *overfitting* e *underfitting* estão sendo evitados.

Inicia-se a comparação dos resultados por meio da análise de dois modelos de velocidades estimados pela rede *HyperColumn*, apresentados na Figura 37(b) e Figura 37(d). Os dois modelos estimados são apresentados em conjunto com suas contrapartes verdadeiras vistas na Figura 37(a) e na Figura 37(c). Nota-se que o primeiro modelo é relativamente mais simples que o segundo, já que possui camadas relativamente bem espaçadas e não possui estrutura de falhas, mesmo que haja inclinação em suas camadas. Entretanto, mesmo possuindo uma dificuldade maior, por conta da presença de estrutura de falhas, percebe-se que a rede *HyperColumn* conseguiu estimar melhor o segundo modelo em vez do primeiro.

A Figura 37(b) mostra justamente que as dificuldades encontradas pelo *HyperColumn* no primeiro modelo foram determinar com exatidão as ondulações das camadas e

separar com maior exatidão cada uma das camadas presentes. Nota-se no modelo estimado que as 3 camadas mais profundas do modelo verdadeiro foram mescladas em 2 e que a velocidade na camada de maior extensão não estimada com exatidão. Em contrapartida, a Figura 37(d) mostra que todas as camadas, e suas respectivas ondulações, assim como a estrutura de falha, foram estimadas de forma bastante similar ao modelo verdadeiro.

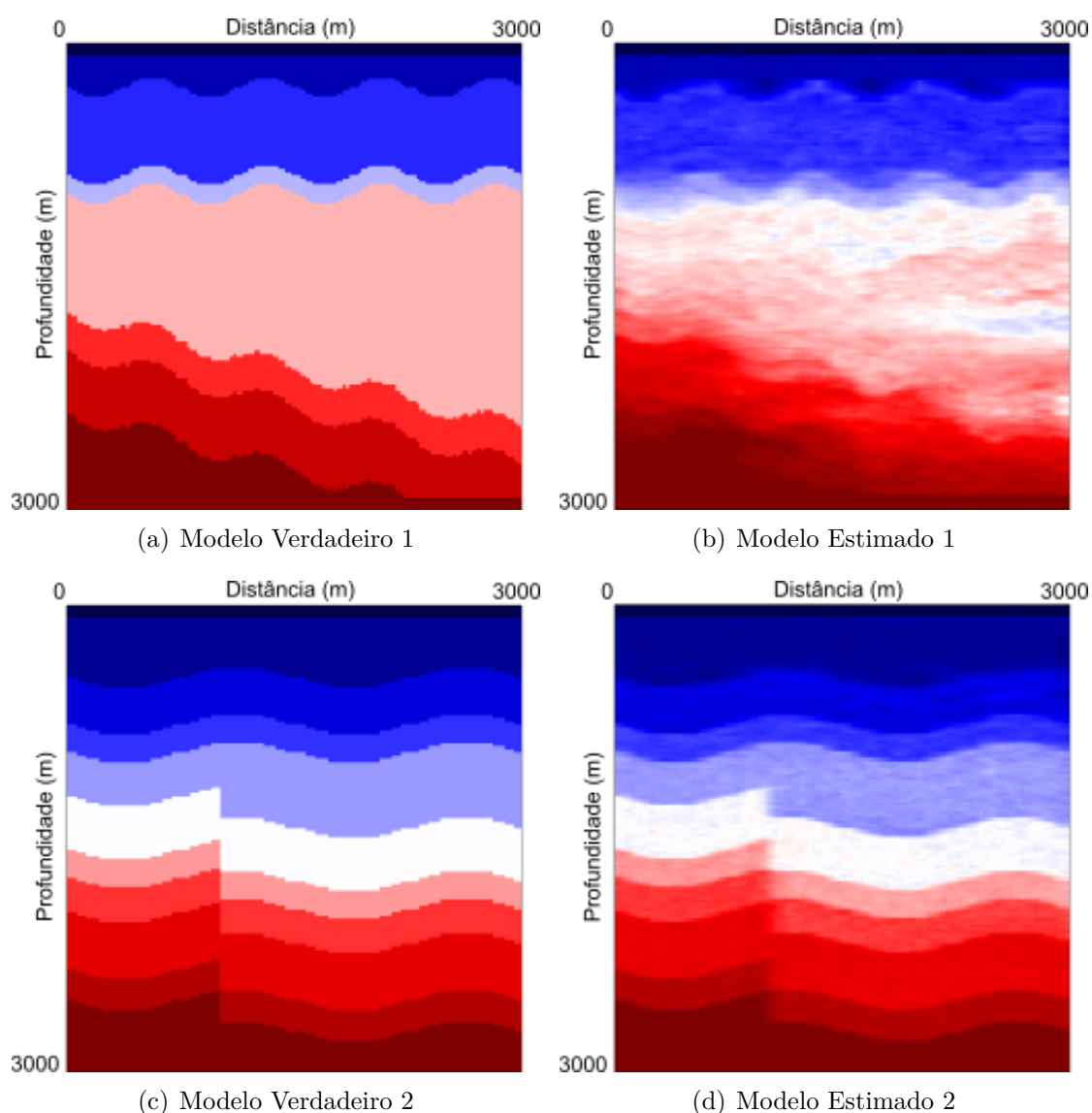


Figura 37 – modelos de velocidades sintéticos estimados utilizando a rede HyperColumn

Este comportamento inconstante ilustrado com estes dois modelos de velocidades ocorreu em outras amostras do conjunto de dados, apesar de não ser algo bastante recorrente. A análise das métricas de avaliação do modelo, mostradas na Tabela 2 em conjunto com o tempo de treinamento, podem auxiliar na explicação de tal comportamento. Primeiro, é importante indicar o tempo de treinamento do modelo de *deep learning*. Apesar de ser treinado por mais épocas, a *HyperColumn* foi treinada totalmente com praticamente metade

do tempo que a U-Net nos experimentos que não utilizam redução de dimensionalidades, o que indica que, mesmo utilizando o dado sísmico completo, gasta-se menos recursos computacionais para estimar um modelo de velocidades com uma resolução aceitável.

Com relação às métricas, percebe-se que o valor de MAE obtido com o *HyperColumn* é o menor em todos os experimentos realizados nesta dissertação. Isto explica a precisão de valores de velocidade nas camadas que foram estimadas com maior clareza pelo *HyperColumn*. Nota-se também que os valores de R^2 , r e fator de dois foram compatíveis com os obtidos nos demais experimentos realizados.

Os últimos pontos a serem analisados são a função de erro (MSE) e o processo de estimação utilizando a rede *HyperColumn*. É possível notar que o valor obtido com o MSE, assim como as demais métricas, compete com os valores obtidos anteriormente, o que explica os poucos casos de inconstância na estimação dos modelos de velocidades e indica que essa inconstância pode estar associada a outros fatores. Além disso, a Figura 36 oferece uma análise extra desta métrica. Nota-se que a taxa de decaimento da função de erro é bastante reduzida nas primeiras épocas, principalmente quando comparadas a outras figuras que ilustram a curva de decaimento. Isto pode ser traduzido em uma dificuldade de ajuste dos parâmetros de treinamento nestas primeiras épocas, o que pode, de certo modo, afetar a capacidade de inferência da *HyperColumn*.

	MSE	MAE	R^2	r	Fator de 2	Treinamento
HyperColumn	6362,7656	38,9869	0,9858	0,9929	1,0000	4,67 horas
U-Net (Melhor resultado)	5212,8091	40,7969	0,9846	0,9923	0,9488	8,14 horas
U-Net (Dados reduzidos)	15063,1153	54,1617	0,9813	0,9906	1,0000	< 1 hora

Tabela 2 – Métricas de avaliação para a HyperColumn e duração do treinamento em horas em comparação às demais técnicas

O primeiro fator pode ser estar associado ao processo utilizado pela *HyperColumn* para extração de características e inversão do modelo sísmico em um modelo de velocidades. Como a estimação é realizada por meio da concatenação das camadas de *hypercolumn* e estas camadas estão associados ao redimensionamento do mapa de atributos logo após a operação de *pooling*, é possível que a informação de localização de algumas características não estejam sendo recuperadas devidamente e, conseqüentemente, algumas camadas são mal estimadas. Isto ocasiona a discussão de um segundo fator: a quantidade de épocas para treinamento.

A Figura 38 mostra como o processo de treinamento da *HyperColumn* avança

baseado no modelo da Figura 38(a). Nota-se que na época 90 a estrutura geral do modelo já está bem definida, mas ocorreram erros na estimação da velocidade presente nas camadas assim como na Figura 37(b), além de algumas ondulações nas camadas não estarem claramente determinadas. Conforme o treinamento avança, nota-se que na época 150 as ondulações estão um pouco mais claras, mas ainda há o problema da mesclagem de camadas. Na última época, os problemas ainda persistem, mas a estimação aparenta estar um tanto mais clara quando comparada com os casos anteriores. Um terceiro ponto surge ao analisarmos a Figura 36: o comportamento da curva pode indicar que uma otimização dos hiperparâmetros utilizados poderia ser feita para evitar a taxa de decaimento reduzida nas primeiras épocas de treinamento.

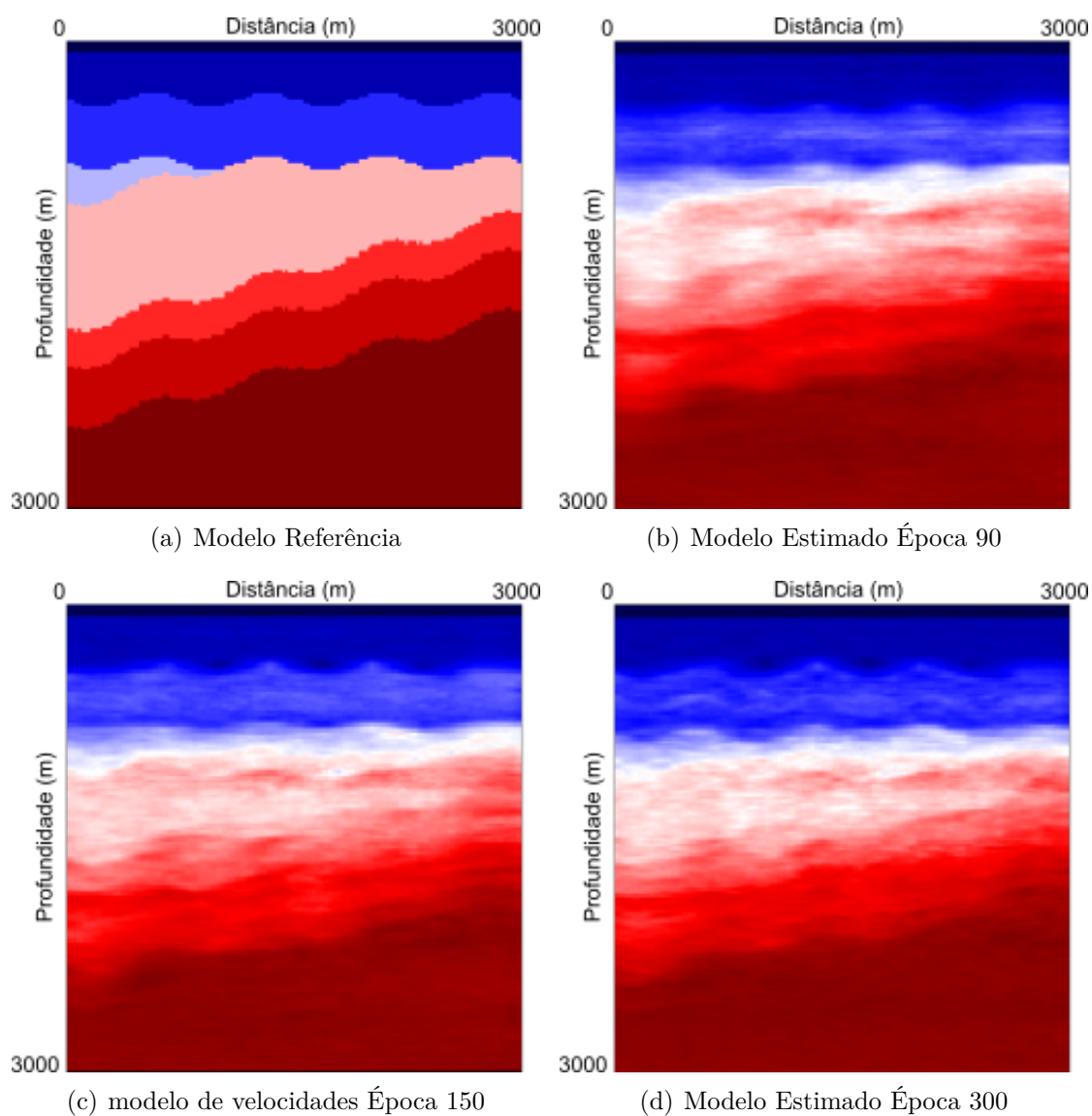


Figura 38 – Avanço do processo de treinamento da rede *HyperColumn* do (a) modelo de velocidades de referência, observando-se o resultado na (b) época 90, (c) época 150 e (d) última época.

Assim, inferimos que é provável que o treinamento da rede por um maior número de épocas, além de uma maior robustez na configuração da rede (otimização de hiperparâmetros), sejam alternativas a serem consideradas para evitar os dois pontos discutidos anteriormente, produzindo resultados mais fiéis e menos inconstantes.

Como o escopo deste experimento é validar o uso de um modelo de *deep learning* além da U-Net, tais alternativas não foram avaliadas. Conclui-se, portanto, que mesmo com certos problemas na estimação do modelo de velocidades, outros modelos de *deep learning*, como a *HyperColumn*, podem, em geral, ser aplicados para inversão sísmica, obtendo resultados satisfatórios tanto qualitativamente, quanto quantitativamente.

5 Considerações Finais

Com estes experimentos iniciais realizados, a U-Net demonstrou ser uma ferramenta promissora para a criação de modelos iniciais de velocidades de alta resolução, i.e., com grande teor de detalhes, para serem utilizados com a técnica FWI. Apesar de alguns modelos iniciais de velocidade gerados com essa técnica carecerem de certos detalhes, como a identificação correta de estruturas de falhas ou demonstrarem certa dificuldade em posicionar corretamente camadas com inclinação, os resultados foram satisfatórios e, uma vez finalizado o processo de treinamento e validação da rede, o método é bastante rápido em estimar um modelo de velocidades com base no sismograma.

Além disso, o uso de *deep learning* aparenta, inicialmente, requerer menos recursos computacionais e interferência humana para construção de um modelo inicial de velocidades altamente correlacionado com o modelo verdadeiro. Outrossim, como o objetivo é gerar um modelo inicial para ser usado no FWI, é através da combinação de ambas as técnicas que se propõe estimar um modelo final de velocidade adequado, colocando de lado, assim, as deficiências identificadas na geração de modelos pela FCN.

Portanto, ao aplicar-se a FWI multiescala nos modelos iniciais estimados pela FCN, pôde-se corrigir tanto a identificação das estruturas de falhas, quanto o posicionamento das camadas com inclinação. Necessita-se, porém, estudar como os parâmetros de modelagem, como número de fontes ou frequência de pico, por exemplo, influenciam no processo de treinamento do modelo de *deep learning*. Adicionalmente, é importante, também, estudar-se o impacto que diferentes hiperparâmetros, como otimizadores ou funções de ativações, possuem nos resultados finais de estimação da U-Net. Assim, buscou-se endereçar estes pontos com outros experimentos com o intuito de melhorar ainda mais o resultado do modelo de velocidades estimado.

Os experimentos que variaram os parâmetros de modelagem ocorreram da seguinte forma: primeiramente fixou-se a frequência de pico da modelagem em 4Hz e variou-se os números de fontes em 1 fonte central, 10, 25 e 50 fontes. Em seguida fixou-se o número de fontes em 25 e variou-se a frequência de pico em 8 e 16Hz. Os resultados mostraram que as melhores métricas após o treinamento da U-Net foram obtidas com os experimentos com 10 e 25 fontes, sendo que o aumento da frequência de pico de 4Hz para 8Hz na modelagem significou uma melhoria ainda maior para a configuração com 25 fontes, principalmente com relação à função de erro. Aumentando-se a frequência de pico para 16Hz, porém, os resultados foram piores do que utilizando a configuração com 50 fontes. O pior resultado obtido foi com os dados sísmicos modelados com apenas 1 fonte central.

Em geral, os resultados são promissores, já que demonstram que uma U-Net pode

estimar modelos de velocidades a partir de sismogramas com poucas ou muitas fontes e alta frequência. As conclusões iniciais apontam que, dependendo do tamanho da subsuperfície, sismogramas com poucas fontes podem ser suficientes para treinar uma U-Net. Entretanto, conforme a subsuperfície cresce, mais fontes podem dar uma melhor representação da área. Além disso, os resultados indicam também que a U-Net, até certo ponto, pode ser menos sensível a altas frequências de pico já que os resultados melhoraram quando a modelagem mudou de 4 para 8Hz, mas pioraram quando utilizada com 16Hz.

O terceiro conjunto de experimentos considerou como o ajuste dos hiperparâmetros da técnica de *deep learning* pode afetar a estimação dos modelos de velocidades. Os dois hiperparâmetros ajustados foram o otimizador e as funções de ativação utilizados na U-Net. Os otimizadores utilizados foram RMSprop, Adam e Adamax, já as funções de ativação foram a ReLU, Leaky ReLU, ELU e Parametric ReLU. Uma boa combinação destes hiperparâmetros pode influenciar o quão detalhado a saída da U-Net será.

Os experimentos mostraram que os otimizadores são mais influentes do que as funções de ativação ao definir a resolução do modelo de saída, já que o uso de uma mesma função de ativação com diferentes otimizadores resultou em modelos de velocidades com diferentes graus de resolução. Assim, mostrou-se que a PReLU é menos sensível às mudanças de otimizador e sua combinação com o Adamax pode produzir modelos de velocidades de alta resolução. Provou-se isto tanto gráfica quanto estatisticamente, já que a U-Net treinada com esta combinação obteve os menores valores de MSE e MAE, ao mesmo tempo que obteve os maiores valores de R^2 , r e um competitivo valor para o fator de 2.

Os três primeiros experimentos realizados nesta dissertação mostraram um avanço e melhoria gradativos na tarefa de estimar um modelo de velocidades inicial a partir de um dado sísmico sintético proveniente de uma modelagem sísmica. Apesar de, ao final, obter-se um modelo de velocidades com alta resolução, foi necessário analisar outras abordagens para validar o uso de *deep learning* para estimação de modelos de velocidades.

Neste contexto, os últimos experimentos envolveram o uso das técnicas AE e PCA para redução de dimensionalidade de dados sísmicos, além do uso de outra FCN, a *HyperColumn*, para estimação de modelos de velocidades. Os resultados da redução de dimensionalidade mostraram ser interessantes, já que foi possível alterar a dimensionalidade dos tiros de um dado sísmico para um tamanho cerca de 99.9% menor utilizando dois métodos de abordagem diferentes e ainda utilizar a U-Net para inverter o dado sísmico reduzido em um modelo de velocidades com resolução razoável. Mesmo existindo uma diminuição de qualidade ao comparar os modelos de velocidades estimados com o dado reduzido com os modelos gerados nos experimento anteriores, vale frisar que os dados

reduzidos apresentam uma redução considerável em relação ao original, significando um modelo de *deep learning* menos complexo e, conseqüentemente, menor custo computacional para treinar o modelo e processar e armazenar o dado sísmico.

Os experimentos com o modelo *HyperColumn* também apresentaram resultados interessantes. A análise qualitativa apresentou um modelo com algumas falhas em sua estimação e um outro modelo estimado bastante próximo ao modelo de velocidades de referência. Discutiu-se que estas falhas, no entanto, podem estar associadas tanto ao número de épocas utilizadas para treinamento, quanto à necessidade de otimizar seus hiperpâmetros. A análise quantitativa, porém, mostrou que, em geral, os modelos estimados pela *HyperColumn* tiveram métricas competitivas com aqueles estimados pela U-Net com os dados sísmicos originais, apresentando, inclusive, uma melhoria da métrica MAE.

Este trabalho de dissertação apresentou como técnicas de *deep learning* podem ser utilizadas para processamento de dados sísmicos, seja ao estimar modelos de velocidades, seja ao aplicar a redução de dimensionalidade em dados sísmicos. Os resultados obtidos foram satisfatórios tanto no sentido interdisciplinar, ao considerar-se como a aplicação de conceitos de áreas diferentes e, a princípio, não correlatas podem ser utilizados em conjunto em diversos experimentos, quanto no sentido científico, ao considerar-se o avanço obtido a cada experimento para a estimação de modelos de velocidades e as possíveis estratégias que podem ser utilizadas ao treinar-se um modelo de *deep learning* como solução para o problema da inversão sísmica.

Referências

- ABADI, M. et al. Tensorflow: A system for large-scale machine learning. In: **12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16)**. [S.l.: s.n.], 2016. p. 265–283.
- ABDI, H.; WILLIAMS, L. J. Principal component analysis. **Wiley interdisciplinary reviews: computational statistics**, Wiley Online Library, v. 2, n. 4, p. 433–459, 2010.
- AGHDAM, H. H.; HERAVI, E. J. Guide to convolutional neural networks. **New York, NY: Springer. doi**, Springer, v. 10, p. 978–3, 2017.
- ALFORD, R.; KELLY, K.; BOORE, D. M. Accuracy of finite-difference modeling of the acoustic wave equation. **Geophysics**, Society of Exploration Geophysicists, v. 39, n. 6, p. 834–842, 1974.
- ARAYA-POLO, M. et al. Deep-learning tomography. **The Leading Edge**, Society of Exploration Geophysicists, v. 37, n. 1, p. 58–66, 2018.
- BAYSAL, E.; KOSLOFF, D. D.; SHERWOOD, J. W. Reverse time migration. **Geophysics**, Society of Exploration Geophysicists, v. 48, n. 11, p. 1514–1524, 1983.
- BISHOP, C. M. **Pattern recognition and machine learning**. [S.l.]: springer, 2006.
- BOX, G. E.; TIAO, G. C. **Bayesian inference in statistical analysis**. [S.l.]: John Wiley & Sons, 2011. v. 40.
- BUNKS, C. et al. Multiscale seismic waveform inversion. **Geophysics**, Society of Exploration Geophysicists, v. 60, n. 5, p. 1457–1473, 1995.
- CARRION, P. Reflection tomography in underwater acoustics. **The Journal of the Acoustical Society of America**, Acoustical Society of America, v. 97, n. 2, p. 1318–1321, 1995.
- CERJAN, C. et al. A nonreflecting boundary condition for discrete acoustic and elastic wave equations. **Geophysics**, Society of Exploration Geophysicists, v. 50, n. 4, p. 705–708, 1985.
- CHOLLET, F. et al. **Keras**. GitHub, 2015. Disponível em: <<https://github.com/fchollet/keras>>.
- COMMITTEE, I. M. T. **Source Separation for Simultaneous Seismic Data Acquisition**. IEEE Dataport, 2017. Disponível em: <<http://dx.doi.org/10.21227/H2TP46>>.
- CORDTS, M. et al. The cityscapes dataset for semantic urban scene understanding. In: **Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)**. [S.l.: s.n.], 2016.
- DJORK-ARNÉ, C.; UNTERTHINER, T.; HOCHREITER, S. Fast and accurate deep network learning by exponential linear units (elus). In: **Proceedings of the International Conference on Learning Representations (ICLR)**. [S.l.: s.n.], 2016. v. 6.

DUCHI, J.; HAZAN, E.; SINGER, Y. Adaptive subgradient methods for online learning and stochastic optimization. **Journal of machine learning research**, v. 12, n. Jul, p. 2121–2159, 2011.

FRANCOIS, C. **Deep learning with Python**. [S.l.]: Manning Publications Company, 2017.

GORI, M. **Machine Learning: A constraint-based approach**. [S.l.]: Morgan Kaufmann, 2017.

HARIHARAN, B. et al. Hypercolumns for object segmentation and fine-grained localization. In: **Proceedings of the IEEE conference on computer vision and pattern recognition**. [S.l.: s.n.], 2015. p. 447–456.

HAYKIN, S. S. **Neural Networks and Learning Machines**. [S.l.]: New York: Prentice Hall, 2009.

HE, K. et al. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In: **Proceedings of the IEEE international conference on computer vision**. [S.l.: s.n.], 2015. p. 1026–1034.

HIGDON, R. L. Absorbing boundary conditions for elastic waves. **Geophysics**, Society of Exploration Geophysicists, v. 56, n. 2, p. 231–241, 1991.

HOLLAND, J. H. et al. **Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence**. [S.l.]: MIT press, 1992.

HUNTER, J. D. Matplotlib: A 2d graphics environment. **Computing in science & engineering**, IEEE, v. 9, n. 3, p. 90–95, 2007.

KENNEDY, J.; EBERHART, R. Particle swarm optimization. In: IEEE. **Proceedings of ICNN'95-International Conference on Neural Networks**. [S.l.], 1995. v. 4, p. 1942–1948.

KINGMA, D. P.; BA, J. Adam: A method for stochastic optimization. In: **Proceedings of the International Conference on Learning Representations (ICLR)**. [S.l.: s.n.], 2015.

KRIZHEVSKY, A.; SUTSKEVER, I.; HINTON, G. E. Imagenet classification with deep convolutional neural networks. In: **Advances in neural information processing systems**. [S.l.: s.n.], 2012. p. 1097–1105.

LECUN, Y. et al. Backpropagation applied to handwritten zip code recognition. **Neural computation**, MIT Press, v. 1, n. 4, p. 541–551, 1989.

LEWIS, W.; VIGH, D. Deep learning prior models from seismic images for full-waveform inversion. In: **SEG Technical Program Expanded Abstracts 2017**. [S.l.]: Society of Exploration Geophysicists, 2017. p. 1512–1517.

LIU, Z.; BLEISTEIN, N. Migration velocity analysis: Theory and an iterative algorithm. **Geophysics**, Society of Exploration Geophysicists, v. 60, n. 1, p. 142–153, 1995.

LONG, J.; SHELHAMER, E.; DARRELL, T. Fully convolutional networks for semantic segmentation. In: **Proceedings of the IEEE conference on computer vision and pattern recognition**. [S.l.: s.n.], 2015. p. 3431–3440.

MAAS, A. L.; HANNUN, A. Y.; NG, A. Y. Rectifier nonlinearities improve neural network acoustic models. In: **Proc. icml**. [S.l.: s.n.], 2013. v. 30, n. 1, p. 3.

MCCULLOCH, W. S.; PITTS, W. A logical calculus of the ideas immanent in nervous activity. **The bulletin of mathematical biophysics**, Springer, v. 5, n. 4, p. 115–133, 1943.

MIN, S.; LEE, B.; YOON, S. Deep learning in bioinformatics. **Briefings in bioinformatics**, Oxford University Press, v. 18, n. 5, p. 851–869, 2017.

MOJICA, O. et al. A comparison between two elitism-based global optimization methods for estimating a starting velocity model for fwi. In: EUROPEAN ASSOCIATION OF GEOSCIENTISTS & ENGINEERS. **81st EAGE Conference and Exhibition 2019**. [S.l.], 2019. v. 2019, n. 1, p. 1–5.

MOJICA, O. F.; KUKREJA, N. Towards automatically building starting models for full-waveform inversion using global optimization methods: A pso approach via deap+devito. In: **SEG Technical Program Expanded Abstracts 2019**. [S.l.]: Society of Exploration Geophysicists, 2019. p. 5174–5178.

NAIR, V.; HINTON, G. E. Rectified linear units improve restricted boltzmann machines. In: **Proceedings of the 27th international conference on machine learning (ICML-10)**. [S.l.: s.n.], 2010. p. 807–814.

OLIPHANT, T. E. **A guide to NumPy**. [S.l.]: Trelgol Publishing USA, 2006. v. 1.

PAN, G.; ABUBAKAR, A.; HABASHY, T. M. An effective perfectly matched layer design for acoustic fourth-order frequency-domain finite-difference scheme. **Geophysical Journal International**, Blackwell Publishing Ltd Oxford, UK, v. 188, n. 1, p. 211–222, 2012.

PEDREGOSA, F. et al. Scikit-learn: Machine learning in python. **Journal of machine learning research**, v. 12, n. Oct, p. 2825–2830, 2011.

PRATT, R. G. Seismic waveform inversion in the frequency domain, part 1: Theory and verification in a physical scale model. **Geophysics**, Society of Exploration Geophysicists, v. 64, n. 3, p. 888–901, 1999.

RONNEBERGER, O.; FISCHER, P.; BROX, T. U-net: Convolutional networks for biomedical image segmentation. In: SPRINGER. **International Conference on Medical image computing and computer-assisted intervention**. [S.l.], 2015. p. 234–241.

RÖTH, G.; TARANTOLA, A. Neural networks and inversion of seismic data. **Journal of Geophysical Research: Solid Earth**, Wiley Online Library, v. 99, n. B4, p. 6753–6768, 1994.

RUDER, S. An overview of gradient descent optimization algorithms. **arXiv e-Print arXiv:1609.04747**, 2017.

SAJEVA, A. et al. Estimation of acoustic macro models using a genetic full-waveform inversion: Applications to the marmousi model genetic fwi for acoustic macro models. **Geophysics**, GeoScienceWorld, v. 81, n. 4, p. R173–R184, 2016.

SANTOS, A. dos. Inversão de forma de onda aplicada à análise de velocidades sísmicas utilizando uma abordagem multiescala. **Dissertação de Mestrado, Universidade Federal da Bahia, Salvador, Brasil**, 2013.

SANTOS, P. N.; PESTANA, R. C. Full waveform inversion using an efficient preconditioning method for the gradient vector applied for different source signatures. **Brazilian Journal of Geophysics**, v. 34, n. 3, p. 405–418, 2016.

SEN, M. K.; STOFFA, P. L. **Global optimization methods in geophysical inversion**. [S.l.]: Cambridge University Press, 2013.

SHAW, R.; SRIVASTAVA, S. Particle swarm optimization: A new tool to invert geophysical data. **Geophysics**, Society of Exploration Geophysicists, v. 72, n. 2, p. F75–F83, 2007.

SHI, Y.; EBERHART, R. A modified particle swarm optimizer. In: IEEE. **1998 IEEE international conference on evolutionary computation proceedings. IEEE world congress on computational intelligence (Cat. No. 98TH8360)**. [S.l.], 1998. p. 69–73.

SOCHACKI, J. et al. Absorbing boundary conditions and surface waves. **Geophysics**, Society of Exploration Geophysicists, v. 52, n. 1, p. 60–71, 1987.

TARANTOLA, A. Inversion of seismic reflection data in the acoustic approximation. **Geophysics**, Society of Exploration Geophysicists, v. 49, n. 8, p. 1259–1266, 1984.

VERSTEEG, R. The marmousi experience: Velocity model determination on a synthetic complex data set. **The Leading Edge**, Society of Exploration Geophysicists, v. 13, n. 9, p. 927–936, 1994.

VIRIEUX, J.; OPERTO, S. An overview of full-waveform inversion in exploration geophysics. **Geophysics**, Society of Exploration Geophysicists, v. 74, n. 6, p. WCC1–WCC26, 2009.

VLACHOS, M. Dimensionality reduction. In: _____. **Encyclopedia of Machine Learning**. Boston, MA: Springer US, 2010. p. 274–279. ISBN 978-0-387-30164-8. Disponível em: <https://doi.org/10.1007/978-0-387-30164-8_216>.

WANG, W.; YANG, F.; MA, J. Velocity model building with a modified fully convolutional network. In: **SEG Technical Program Expanded Abstracts 2018**. [S.l.]: Society of Exploration Geophysicists, 2018. p. 2086–2090.

WU, Y.; LIN, Y.; ZHOU, Z. Inversionnet: Accurate and efficient seismic waveform inversion with convolutional neural networks. In: **SEG Technical Program Expanded Abstracts 2018**. [S.l.]: Society of Exploration Geophysicists, 2018. p. 2096–2100.

Glossário

Análise Qualitativa Analisa a qualidade do modelo estimado com base nas imagens resultantes.

Análise Quantitativa Analisa a qualidade do modelo estimado com base em métricas estatísticas.

Backpropagation Algoritmo de retropropagação de erro utilizado durante o processo de treinamento de uma rede neural.

Campo Local Induzido Soma ponderada de todas as entradas de um neurônio.

Deep Learning Subconjunto de *Machine Learning* que adiciona camadas consecutivas de representação para encontrar ainda mais representações de um dado de entrada.

Função de Ativação Define a saída de um neurônio com base em um campo local induzido.

Função de *loss* Função objetivo que guia o processo de aprendizagem de uma rede neural.

Inversão Sísmica Processo de geração de um modelo de velocidade por meio de um dado sísmico.

Kernel Matriz de valores aplicado em uma imagem para extração de características. Também conhecido como filtro.

Machine Learning Conjunto de técnicas que aprende representações úteis de um conjunto de dados de entrada e define regras para melhor descrevê-los.

Mapa de Atributos Características extraídas de uma imagem após a aplicação de uma operação de convolução.

Modelagem Sísmica Simula o processo de propagação de ondas em uma subsuperfície.

Modelo de Velocidade Representa os elementos de uma subsuperfície com base na velocidade que as ondas se propagaram neles.

Neurônio Artificial Menor unidade que compõe uma rede neural artificial.

Otimizador Responsável por guiar o espaço de busca ao ótimo global durante o processo de aprendizagem de uma rede neural com base em uma função objetivo.

Rede Neural *feedforward* Redes cuja composição de neurônios não é cíclica, ou seja, não possui retro-alimentação.

Sismograma Dado gerado por meio de aquisição ou modelagem sísmica que representa a propagação de onda em uma subsuperfície.

Stride Espaçamento do movimento de um *kernel* ao ser aplicado em uma imagem.