Karlsruhe Institute of Technology

DEPARTMENT OF PHYSICS
Geophysical Institute

Application of machine learning for the extrapolation of seismic data

Master's Thesis
of

Amelie Cathrine Nüsse

at the Geophysical Institute

Reviewer:	Prof. Dr. Thomas Bohlen
Second Reviewer:	Prof. Dr. Dirk Gajewski

Date of submission: 22.11.2022

Erklärung zur Selbstständigkeit

Ich versichere, dass ich diese Arbeit selbstständig verfasst habe und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe, die wörtlich oder inhaltlich übernommenen Stellen als solche kenntlich gemacht und die Satzung des KIT zur Sicherung guter wissenschaftlicher Praxis in der jeweils gültigen Fassung beachtet habe.

Karlsruhe, den 22.11.2022, ____________________________

Amelie Cathrine Nüsse

Abstract

Low frequencies in seismic data are often challenging to acquire. Without low frequencies, though, a method like full-waveform inversion might fail due to cycle-skipping. This thesis aims to investigate the potential of neural networks for the task of low-frequency extrapolation to overcome aforementioned problem. Several steps are needed to achieve this goal: First, suitable data for training and testing the network must be found. Second, the data must be pre-processed to condition them for machine learning and efficient application. Third, a specific workflow for the task of low-frequency extrapolation must be designed. Finally, the trained network can be applied to data it has not seen before and compared to reference data. In this work, synthetic data are used for training and evaluation because in such a controlled experiment the target for the network is known. For this purpose, 30 random but geologically plausible subsurface models were generated based on a simplified geology around the Asse II salt mine, and used for finite-difference simulations of seismograms. The corresponding shot gathers were pre-processed by, among others, normalizing them and splitting them up into patches, and fed into a convolutional neural network (U-Net) to assess the network's performance and its ability to reconstruct the data. Two different approaches were investigated for the task of low-frequency extrapolation. The first approach is based on using only low frequencies as the network's target, while the second approach has the full bandwidth as target. The latter yielded superior results and was therefore chosen for subsequent applications. Further tests of the network design led to the introduction of ResNet blocks instead of simple convolutions in the U-Net layers, and the use of the mean-absolute-error instead of the mean-squared-error loss function. The final network designed in this way was then applied to the synthetic data originally reserved for testing. It turned out that the chosen method is able to successfully extrapolate low frequencies by more than half an octave (from about 8 to 5 Hz) given the experimental setup at hand. Although the results start to deteriorate in the low-frequency band for larger offsets, full-waveform inversion will overall benefit from the application of the presented machine learning approach.

Contents

List of Figures

List of Tables

1. Introduction

Machine learning is becoming more and more successful with the increasing computational power and the increasing amount of labeled training data available. At present, machine learning surrounds us anywhere in our daily life. It might be as simple as email spam detection or advertisement recommendations but can be as advanced as voice assistants such as "Alexa" or advanced driver assistance systems, like traffic sign recognition in modern cars. However, machine learning is not only improving everyday life but is also a great tool to, for example, reveal underlying patterns or automate time-intensive processes in any data-intensive scientific discipline.

Geophysics is such a data-intensive discipline. For that reason, the interest in machine learning has grown significantly in the geophysics community in recent years. The first publications on machine learning in geophysics were in the late 1980s, focusing on seismic interpretation, such as horizon tracing (Liu et al., 1989) and bright spot detection (Huang et al., 1989). Since then, an exponential growth in the number of Society of Exploration Geophysicists (SEG) publications in the field of machine learning has been observed (Yu and Ma, 2021). Nowadays, machine learning is used in a wide variety of geophysical applications, which ranges from fault detection (Liu et al., 2020), detection of arrivals of the P- and S-phases of earthquakes (Zhu and Beroza, 2019) to facies classification (Qian et al., 2018). The aforementioned applications are all classification tasks, however, machine learning is also coming up in the field of inversion. Neural networks are used to predict velocity models from shot gathers (Yang and Ma, 2019; Kazei et al., 2021). Furthermore, Richardson (2018) implemented the full-waveform inversion (FWI) algorithm as a recurrent neural network.

FWI is another powerful tool in geophysics that enables us to utilize the entire information of the full waveform to obtain a model of the subsurface. Same as machine learning, FWI is an optimization problem. It has the goal to converge synthetic data towards the measured data (Virieux and Operto, 2009). Bunks et al. (1995) introduced the widely used multistage approach to converge the data. In this approach, the data are inverted in several stages covering different frequency bands, starting from low frequencies towards high frequencies. Low frequencies will give a smooth model of the subsurface, while high frequencies will reveal high-resolution details of the structures in the subsurface (Bunks et al., 1995). Furthermore, low frequencies allow for a deeper penetration depth, reduced side lobe energies of the wavelet, and improved absolute impedance estimation (ten Kroode et al., 2013).

Even though low frequencies are important for the further processing of seismic data, their acquisition is challenging. Receivers have to deal with a reduced signal-to-noise ratio at low frequencies (ten Kroode et al., 2013) and sources are usually not able to produce very low frequencies (Wehner et al., 2019). Therefore, seismic data often lack such frequencies.

Without low frequencies, FWI becomes a highly nonlinear optimization task. The objective function will be complex and can contain several local minima, if the starting model does not have sufficient accuracy. Hence, the FWI optimization algorithm might get stuck in one of the local minima instead of the global minimum. This is called cycle skipping (Hu et al., 2018). In this context, low frequencies help in making application of FWI on field data feasible.

1.1. Related work

The importance of low frequencies is well known, so different methods were developed to address the problem of missing low frequencies. One workaround is to change the existing FWI scheme. This has been achieved by either using advanced misfit functions (Warner and Guasch, 2016; Sun and Alkhalifah, 2020), conditioning of the gradient (Ma et al., 2012; Ovcharenko et al., 2018a) or additional constraints on the misfit (van Leeuwen and Herrmann, 2013; Zhang et al., 2018). Another approach is using data-driven methods to synthesize low-frequency information. Wu et al. (2013) suggest using the envelope for low-frequency information. Hu (2014) propose the beat-tone method, where the low-frequency information is taken from the difference between data of adjacent frequency bands. Li and Demanet (2016) implement a phase-tracking algorithm that separates seismic recordings into elementary events. Low frequencies can then be extrapolated by changing the wavelet. In recent years, machine learning techniques have been applied to the task of low-frequency extrapolation as well.

Ovcharenko et al. (2017) applied a deep-feedforward neural network. They use the real and imaginary parts of the frequency-wavenumber (f-k) spectra from each receiver as input to their neural network and trained the network to predict low-frequency spectra. The resulting spectra are close to each other, though. Especially in the far offsets, their fit is not very accurate. Ovcharenko et al. (2018b) extended their work by applying FWI to the extrapolated data. The network can predict general trends; however, small-scale details are not correctly reconstructed. Applying FWI to the data showed that the low frequencies were not reconstructed well enough to be useful for FWI. In their subsequent work, Ovcharenko et al. (2019) upgrade their network to a convolutional neural network. The input and output are high-frequency spectra and single low-frequency spectra given for shot gathers instead of single receivers as in the previous work.

Jin et al. (2018) introduced a network to extrapolate low-wavenumbers by combining their beat-tone method (Hu, 2014) with the Inception network (Szegedy et al., 2015). The extrapolated data showed better results when used in FWI than the beat-tone approach alone. In their subsequent work, Hu et al. (2021) introduced a progressive transfer learning approach by integrating the previous network into a physics-based FWI workflow. In this approach, the training data evolve as the velocity model is updated with FWI. Following the physics-guided approach further, Hu et al. (2020) extended their workflow by a pretext task to obtain a better initial starting model. With the integration of conventional FWI into the machine learning workflow, the network lacks efficiency. Jin et al. (2021) changed their previous approach from frequency-domain to time-domain, by using a U-Net architecture instead of the Inception network. The conventional FWI is replaced with a truncated FWI to reduce the time for the FWI loop.

In contrast to the others, Sun and Demanet (2018) proposed to apply a CNN directly to bandwidth-limited data in the time domain. They use band-limited shot gathers as input to their network and predict low-frequency shot gathers from the input. The presented traces show a promising fit. In their subsequent work, Sun and Demanet (2019) verified the predictions of their network by using the data for FWI. With the extrapolated frequencies, cycle-skipping can be omitted.

In their more recent approaches instead of training on complete shot gathers, Sun and Demanet (2020b) used a trace-by-trace approach. This results in extrapolated low-frequency traces that are accurate enough to improve FWI. The trace-by-trace approach, however, suffers from decreased coherence between neighboring traces. To expand their network to elastic data, Sun and Demanet (2020a) trained their network on multi-component data. Training on the v_x and v_y components enables the prediction of low frequencies for multi-component data. Sun and Demanet (2022) changed the architecture of the used network to improve their result even further. Instead of conventional convolutions, dilated convolutions (van den Oord et al., 2016) are used and instead of single traces, multiple traces of neighboring receivers are used. Especially for diving waves, a great fit could be achieved, which gave reasonable low-wavenumber FWI results.

Fabien-Ouellet (2020b) used a recursive convolutional neural network to generate low-frequency data. Compared to other work, the network is not bound to a certain frequency for input and target but extrapolates the central frequency to half the center frequency of the input data. The extrapolated data show a good fit and the network is able to denoise the very low-frequency data. The different frequency bands can then be used directly for FWI to overcome the cycle-skipping problem (Fabien-Ouellet, 2020a).

Wang et al. (2020) trained a U-Net, see section 2.1.5.1, on band-limited shot gathers. Compared to the other approaches, not low frequencies, but extended bandwidths are used as targets. Quantified error values as well as residual plots are missing in this publication, which makes it hard to evaluate the results.

Aharchaou et al. (2021) proposed a sequence-to-sequence approach in the frequency-wavenumber domain. In contrast to the others, the network is not trained on synthetic data, but on field data acquired with ocean-bottom nodes (OBN). The trained network is then applied to towed-streamer data. For strong-amplitude events the extrapolated data show a good match to the true data. In parts of the data with a weaker signal-to-noise ratio, the signal is not recovered correctly. In another approach Aharchaou and Baumstein (2020) applied a U-Net to time domain data from their ocean-bottom node recordings. The resulting low frequencies could not recover all events perfectly; some events ended up with too large amplitudes. Furthermore, the incoherent noise in the OBN field data also inherently influenced the towed-streamer prediction.

Nakayama and Blacquière (2021) utilized a U-Net for simultaneously extrapolating low frequencies, deblending and data reconstruction. The network is trained on synthetic data as well as field data.

In their latest approaches Ovcharenko et al. (2021) use shot gathers of two different frequency bands as input to a generative adversarial network (GAN). The idea is to integrate field data into the training process. The aim of the network is to correctly reconstruct the low-frequency field data. The results of this approach seem to be promising. However, a quantified error is missing in the papers.

Another approach of Ovcharenko et al. (2022) is the implementation of a network for multi-task learning. Instead of just implementing the low frequencies, also an initial velocity model is predicted by the network. The advanced loss function helps to increase the accuracy of the low frequencies and an improved initial starting model helps FWI to overcome the cycle-skipping problem. With decreasing frequency and/or increasing arrival time, the match between the target and predicted data reduces. Nevertheless, the results are still good enough to mitigate cycle-skipping.

1.2. Thesis outline

To overcome the cycle-skipping problem in FWI, low-frequency information is needed. Such information can be obtained by different techniques. The aim of this thesis is to investigate the potential of a deep learning approach to extrapolate frequencies below 8 Hz and develop such an approach. This thesis consists of 5 chapters:

After the introduction, which has already given an overview of related work, chapter 2 discusses the theoretical background to understand the methods used in this work. First, the basic ideas of machine learning are presented, which is essential to understand the network presented in this work. Then, I shortly cover the methodology of seismic modeling with finite differences as well as the background information on FWI.

Chapter 3 deals with the preparation of synthetic data. This includes the random generation of subsurface velocity models, the simulation of wave propagation in the models to obtain synthetic seismic data, and the development of a pre-processing routine for the modeled data to serve as training data for the neural network.

The pre-processed data are then used as training data to face the task of low-frequency extrapolation with a U-Net (chapter 4). Two different approaches are discussed, which is followed by several hyperparameter tests to find the optimal setting for the network. Then, the final result is presented and further studied.

Finally, my findings are summarized in chapter 5. Furthermore, an outlook is given on how this approach can be improved and the resulting data applied in FWI.

2. Theoretical background

Three topics are addressed in this thesis: Deep learning to extrapolate low frequencies, seismic modeling to generate training data, and full-waveform inversion as motivation for the extrapolation of low frequencies. In this chapter, the background information on those disciplines is presented.

2.1. Deep learning

This section provides an overview of the theory of deep learning. Since deep learning is a broad topic, only parts of the theory can be discussed within the scope of this thesis. For additional details, I refer to the books of Bishop (2006) and Goodfellow et al. (2016). In the following, artificial neurons and neural nets are introduced. Afterwards, different activation functions and their advantages are discussed. Next, convolutional neural networks and the architecture of encoder-decoder networks are explained.

2.1.1. Neurons and neural networks

Deep learning is based on artificial neural networks (Goodfellow et al., 2016). These are inspired by neurons in the biological brain (Rumelhart, 1988). In Figure 2.1 a sketch of an artificial neuron is displayed. Neurons consist in their basic form of a series of linear combinations:

$$a = \sum_{i=1}^{D} w_i x_i + w_0 \ , \tag{2.1}$$

where x_1,...,x_D are the input variables, w_i the weight of the ith input and w_0 the bias (Bishop, 2006). The result a is called activation. The output z of a neuron is then a transformation of the activation

$$z = h(a) \ , \tag{2.2}$$

where $h(\cdot)$ is a differentiable activation function (Bishop, 2006). Different activation functions are discussed in section 2.1.2.

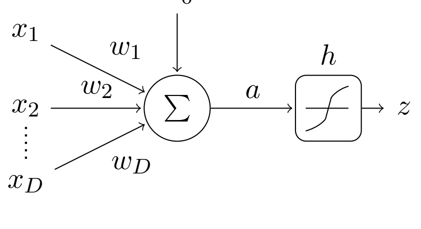


Figure 2.1.: Sketch of an artificial neuron. The input $x_1, x_2, \ldots, x_D$ is scaled with individual weights $w_1, w_2, \ldots, w_3$ and summed up as activation a. The activation is then transformed by an activation function h to form the output z.

In deep learning, neural networks usually consist of several layers of neurons (Goodfellow et al., 2016). A simple network structure is displayed in Figure 2.2. Each node or unit represents one neuron. The first layer is called input layer, the last layer is called output layer. All remaining layers are named hidden layers. This means, every deep learning model consists of at least three layers. In the sketch, only one hidden layer is used. For the neurons in the hidden units, equation 2.1 changes to

$$a_j = \sum_{i=0}^{D} w_{ji}^{(1)} x_i + w_{j0}^{(1)} \ , \tag{2.3}$$

where j is the index of the neuron in the current layer and the superscript (1) shows the number of the current layer, here the first hidden layer. The transformed activation z_j is then the new input for neurons in the output layer. Therefore, the second layer is of the form

$$a_k = \sum_{j=1}^{M} w_{kj}^{(2)} z_j + w_{k0}^{(2)} \ . \tag{2.4}$$

With $\sigma(\cdot)$ as the final activation function, we can express the output y_k of the network in terms of the input variables and the adjustable weights as

$$y_k(\mathbf{x}, \mathbf{w}) = \sigma \left(\sum_{j=1}^{M} w_{kj}^{(2)} h \left(\sum_{i=0}^{D} w_{ji}^{(1)} x_i + w_{j0}^{(1)} \right) + w_{k0}^{(2)} \right) \ . \tag{2.5}$$

The network is a forward operator applied to the input variables x_i which gives the output variables y_k depending on the weights w. (Bishop, 2006)

2.1.2. Activation functions

There are several types of activation functions that are typically used in deep learning, see Figure 2.3. The simplest version is a linear function. Most of the time deep learning problems are nonlinear, though. Therefore, a nonlinear activation function is more beneficial. Usually sigmoidal functions are used, such as the logistic function or the tanh function (Bishop, 2006). The logistic sigmoid function $s(x)$ is defined as

$$s(x) = \frac{1}{1 + e^{-x}} \tag{2.6}$$

and has a value range from 0 to 1, see Figure 2.3a (Goodfellow et al., 2016). The tanh function has a similar appearance:

$$\tanh(x) = \frac{2}{1 + e^{-2x}} \tag{2.7}$$

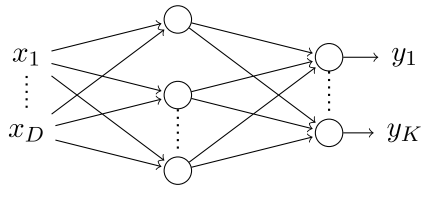


Figure 2.2.: Sketch of a two-layered fully connected neural network, where circles depict single neurons. The input has D features, the output consists of K features. The layer between the input and the output layer is the hidden layer. Each neuron has an additional bias term which is not shown in the sketch.

However, it is shifted and scaled compared to the logistic function, see Figure 2.3b (Goodfellow et al., 2016). Its values range from -1 to 1. If the target of a network has the same value range, sigmoidal functions work well for output neurons. However, they saturate quickly for very negative and very positive values, which makes training difficult (Goodfellow et al., 2016). Therefore, they are only sensitive for values close to zero. For this reason, the rectified linear unit (ReLU) was introduced. The ReLU is defined as

$$\mathrm{ReLU}(x) = \max(0, x) \tag{2.8}$$

The ReLU behaves just like a linear activation function for positive values but is zero for negative values, see Figure 2.3c (Glorot et al., 2011). With that, the gradient is high whenever the output is nonzero, and zero elsewhere. Through active units, the gradient can flow very well without the problem of a vanishing gradient, which is one of the problems sigmoidal functions face (Glorot et al., 2011). However, the neuron cannot learn anything for samples where the unit is inactive, as the gradient is zero (Goodfellow et al., 2016). To overcome this problem, different variants of the ReLU have been developed in recent years, such as the leaky ReLU (Maas et al., 2013), the parametric ReLU (He et al., 2015) or the ELU (Clevert et al., 2016), by changing the negative side of the ReLU to a nonzero function.

Another more advanced version is the scaled exponential linear unit (SELU), which was introduced by Klambauer et al. (2017) and is defined as

$$\mathrm{SELU}(x) = \begin{cases} \lambda x, & \text{if } x > 0, \\ \lambda\alpha(e^x - 1) & \text{otherwise} , \end{cases} \tag{2.9}$$

with $\lambda \approx 1.050700987$ and $\alpha \approx 1.673263242$. With this design of the SELU, a normalization of the activations to zero mean and unit variance is achieved. Having negative and positive values allows to control the mean. The saturation region allows to dampen the variance as there the derivative converges towards zero. A slope larger than one for positive values allows to increase the variance (Klambauer et al., 2017), see Figure 2.3d. Compared to the ReLU the exponential part of the SELU has another advantage. The gradient is nonzero for negative values, which means that the information can flow backward through these units. Normalizing neuron activations allows for robust training of neurons and avoids exploding or vanishing gradients (Klambauer et al., 2017).

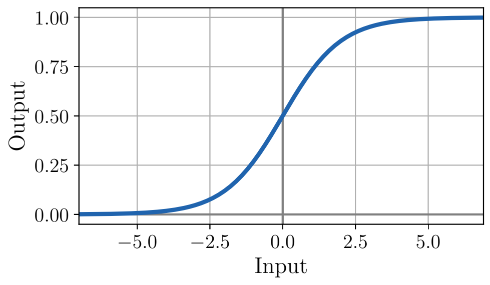
2.1.3. Training neural networks

To train a neural network means to adjust the trainable parameters θ in such a way that the objective function $J(\theta)$ is optimized (Goodfellow et al., 2016; Bishop, 2006). Note that θ corresponds not only to the weights w, but also includes other trainable parameters. Often, the objective function is minimized and can therefore be called the error function, loss function, or cost function (Goodfellow et al., 2016). Typically, minimizing is done iteratively with a gradient-based optimizer (Goodfellow et al., 2016). Hereby, mostly stochastic gradient descent (SGD) or variations of it are used (Goodfellow et al., 2016). The cost function is often a sum over the per-example loss:

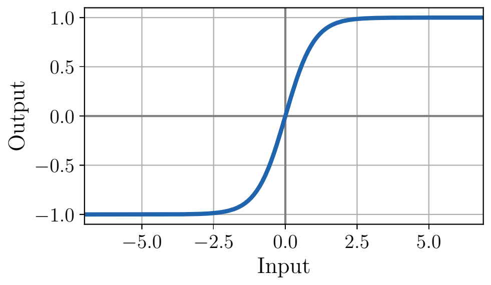
$$J(\theta) = \frac{1}{m} \sum_{i=1}^{m} L\left(x^{(i)}, y^{(i)}, \theta\right) \ , \tag{2.10}$$

where m is the number of training samples (Goodfellow et al., 2016). Thus, the gradient can be expressed as:

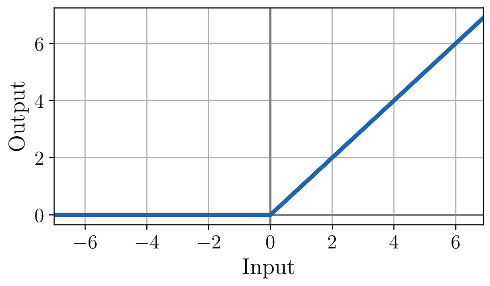
$$\nabla_\theta J(\theta) = \frac{1}{m} \sum_{i=1}^{m} \nabla_\theta L\left(x^{(i)}, y^{(i)}, \theta\right) \ , \tag{2.11}$$



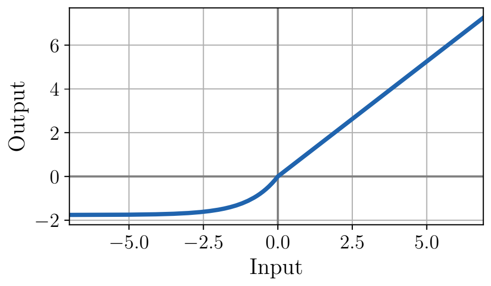
(a) Logistic Sigmoid



(b) Tanh



(c) ReLU



(d) SELU

Figure 2.3.: Common activation functions used in deep learning.

The more training data are used, the more computationally expensive the gradient computation will be. Therefore, SGD uses an estimation of the gradient based on a small set of samples. Each step of the algorithm uses a so-called minibatch $\mathbb{B} = \left\{x^{(1)}, \ldots, x^{(m')}\right\}$ to estimate the gradient:

$$g = \frac{1}{m'} \nabla_\theta \sum_{i=1}^{m'} L\left(x^{(i)}, y^{(i)}, \theta\right) \tag{2.12}$$

Next, the parameters θ are updated in downhill direction:

$$\theta \leftarrow \theta - \epsilon g \ , \tag{2.13}$$

where ϵ is the learning rate or step size. When using the SGD algorithm, the computation time per update does not increase with the number of training samples. The number of epochs is a hyperparameter that specifies the number of passes through the complete training data set. Meaning, if the data set consists of n minibatches, each epoch will consist of n updates of the gradient. (Goodfellow et al., 2016)

Adam optimizer

The SGD algorithm (equation 2.13) can be slow. Hence, there are different implementations of optimization algorithms. Here, only the Adam optimizer will be discussed, as it is used for training the network in this work. The Adam optimizer is an adaptive learning-rate optimizer and was introduced by Kingma and Ba (2015). The name is an abbreviation for "adaptive moments". The update scheme for Adam is

$$m \leftarrow \rho_1 m + (1-\rho_1) g \tag{2.14}$$

$$v \leftarrow \rho_2 v + (1-\rho_2) g \odot g \tag{2.15}$$

$$\hat{m} \leftarrow \frac{m}{1-\rho_1^t} \tag{2.16}$$

$$\hat{v} \leftarrow \frac{v}{1-\rho_2^t} \tag{2.17}$$

$$\Delta\theta \leftarrow -\epsilon \frac{\hat{s}}{\sqrt{\hat{r}}+\delta} \tag{2.18}$$

$$\theta \leftarrow \theta + \Delta\theta \ , \tag{2.19}$$

where g is the gradient as defined in equation 2.12. The first moment, the mean m, and the second moment, the uncentered variance v, of the gradient are initialized as zero. $\hat{m}$ and $\hat{v}$ are the bias-corrected moment terms. $\rho_{1,2}$ are exponential decay rates. The algorithm updates the moving averages of the gradient and the squared gradient, which are estimates of the first and second moment, respectively. Compared to SGD, for the Adam optimizer the update direction is not necessarily the gradient direction but takes previous gradient directions into account.

Gradient calculation by backpropagation

Training of a neural network starts with the initialization of the weights to small random values (Goodfellow et al., 2016). The information of the input is then forward propagated through the network with equation 2.5 and the cost function J is calculated from the output. To calculate the gradients for optimization, the information of the cost function needs to flow back through the network, which is called backpropagation (Rumelhart et al., 1986; Goodfellow et al., 2016). For the derivation of the backpropagation algorithm, I follow the original derivation of Rumelhart et al. (1986). Backpropagation is based on the chain rule of calculus. Let us assume we have a simple multi-layer feedforward neural network as discussed in section 2.1.1. The output y of the neuron j in the last layer is then

$$y_j = \sigma\left(\sum_i z_i w_{ij}\right) \ , \tag{2.20}$$

where $\sigma(\cdot)$ is a nonlinear activation function and z_i is the input of the neuron, which is simply the output of the ith neuron in the previous layer. When d is the desired output of the network, we can design an objective function that compares the target and the prediction, for example a squared error:

$$J = \frac{1}{2}\sum_c \sum_j \left(y_j^{(c)} - d_j^{(c)}\right)^2 \ , \tag{2.21}$$

where c indicates different input-output pairs. To minimize J with SGD, the partial derivative of J with respect to the weights w is needed. The first step of the backward pass is to compute $\frac{\partial J}{\partial y}$. Differentiating equation 2.21 for a particular input-output pair c gives the following:

$$\frac{\partial J}{\partial y_j} = y_j - d_j \tag{2.22}$$

Applying the chain rule to get $\frac{\partial E}{\partial a}$ leads to:

$$\frac{\partial J}{\partial z_j} = \frac{\partial J}{\partial y_j} \cdot \frac{\partial y_j}{\partial a_j} \ , \tag{2.23}$$

where a_j is the linear combination of the weights w_{ij} and the input x_j. Finally, the partial derivative $\frac{\partial J}{\partial w_{ji}}$ can be calculated as

$$\frac{\partial J}{\partial w_{ji}} = \frac{\partial J}{\partial a_j} \cdot \frac{\partial a_j}{\partial w_{ij}} \tag{2.24}$$

$$= \frac{\partial J}{\partial y_j} \cdot \frac{\partial y_j}{\partial a_j} \cdot \frac{\partial a_j}{\partial w_{ij}} \tag{2.25}$$

$$= (y_j - d_j)\,\sigma'(a_j)x_i \ . \tag{2.26}$$

It can be seen that the activation function h must be at least partially differentiable. Furthermore, the partial derivative with respect to x_i can be calculated as

$$\frac{\partial J}{\partial x_i} = \sum_j \frac{\partial J}{\partial a_j} \cdot \frac{\partial a_j}{\partial x_i} \tag{2.27}$$

$$= \sum_j \frac{\partial J}{\partial a_j} \cdot w_{ij} \ . \tag{2.28}$$

Hence, the gradients for all input features to the neuron of the last layer are known. As those features are the output of neurons in the layer before, the gradient can be calculated in the same manner for the next layer, passing the information from the last layer all the way to the first layer. In essence, the backpropagation of a deep neural network is a repeated application of the chain rule.

2.1.4. Convolutional neural networks

For grid-like data, for example, 1D time series or 2D images, a fully connected network such as the one discussed in section 2.1.1 cannot take account of the fact that data points nearby in these grid structures have a higher correlation to the current data point than data points farther away (Bishop, 2006; Goodfellow et al., 2016). Convolutional neural networks (CNNs) overcome this by using convolutions:

$$s(t) = \int x(a)w(t-a)da \tag{2.29}$$

$$= x(t) * w(t) \ , \tag{2.30}$$

where x is referred to as input, w as kernel or filter, and s as feature map. As data are usually not stored continuously, a discrete convolution is used for machine learning approaches:

$$s(t) = x(t) * w(t) = \sum_{a=-\infty}^{\infty} x(a)w(t-a) \tag{2.31}$$

However, for multi-dimensional tensors that are often used in deep learning, also the convolution has to be multi-dimensional. For 2D data the discrete convolution changes to:

$$S(i,j) = I(i,j) * K(i,j) = \sum_m \sum_n I(m,n)K(i-m,j-n) \tag{2.32}$$

To reduce the range of values for m and n, the commutative law is used:

$$S(i,j) = K(i,j) * I(i,j) = \sum_m \sum_n I(i-m,j-n)K(m,n) \tag{2.33}$$

In many neural network libraries, such as TensorFlow (Abadi et al., 2015), convolution is often implemented as a cross-correlation,

$$S(i,j) = I(i,j) * K(i,j) = \sum_m \sum_n I(i+m,j+n)K(m,n) \ , \tag{2.34}$$

which is similar to a convolution but without the flipped kernel. The network will then learn the flipped kernel instead of the original kernel for a convolution. In the end, both implementations give the same result (Goodfellow et al., 2016). Figure 2.4 shows an example of a convolution as implemented in TensorFlow. The kernel slides over the input and a linear combination of the overlying elements of kernel and input is written in the corresponding output position.

The three main advantages of convolutional layers are that a CNN has sparse interactions, shared weights, and equivariant representations (Goodfellow et al., 2016; Bishop, 2006). Sparse interactions are possible as the kernel is chosen to be significantly smaller than the input data. With that, also the dimensions of the matrix multiplication decrease, meaning less parameters need to be trained. To reduce the amount of trainable weights even further, shared weights are used. This means that the weights of the kernel are the same for every location on the input. As the kernel values are tied, the convolution is also equivariant to translation. This implies that if a kernel is sensitive for a particular feature, then the same result is produced for each possible position of the feature in the input.

Furthermore, some variants of the convolution can be used to change the output size. Setting a stride allows one to subsample by letting the kernel pass over some positions (Dumoulin and Visin, 2016). This reduces the computational cost and size of the output feature map. To avoid a larger reduction of the output size, zero-padding can be introduced. The idea is to insert rows and columns with zeros around the original input to make it wider. Depending on the number of zeros added, the size of the output will stay the same as the input, or it is reduced (Goodfellow et al., 2016). The width of the resulting output W_{out} for an input of width W_{in} that passes through a convolution can be calculated as

$$W_{out} = \frac{W_{in} - K + 2P}{S} + 1, \tag{2.35}$$

with a kernel of size K, a padding of P and a stride of S (Dumoulin and Visin, 2016).

Batch normalization

As discussed in section 2.1.2 when introducing the SELU, normalized activations are beneficial for the training process. Another idea, which is nowadays the standard for CNNs, is batch normalization, or in short, batch norm (Ioffe and Szegedy, 2015). This technique reparameterizes the current minibatch H to get a normalized output

$$H' = \frac{H - \mu}{\sigma}, \tag{2.36}$$

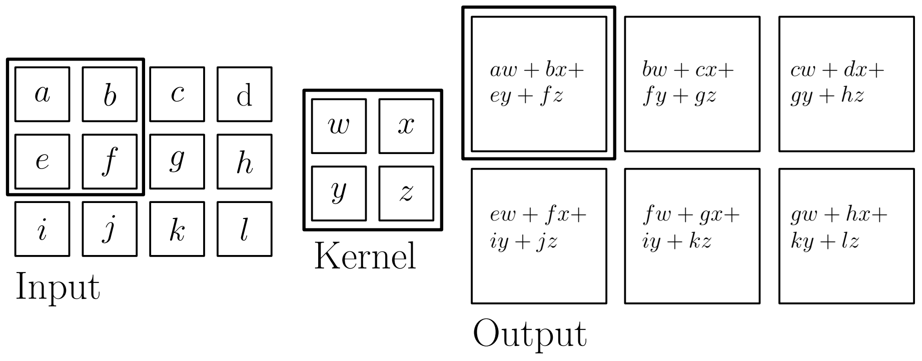


Figure 2.4.: Schematic of a convolution, here implemented as a cross-correlation. The kernel slides over the input and the values are multiplied and summed up accordingly. The boxes show how the upper-left element of the output is formed. Figure drawn after Goodfellow et al. (2016)

where μ denotes the mean of the current batch and σ the standard deviation. When backpropagating through the network, we will also backpropagate through the batch norm layer. This hinders the network to simply increase the standard deviation or the mean, as this gets cancelled out by the normalization process (Goodfellow et al., 2016). After training, the network is applied to the test data set. In this case, μ and ρ are not calculated from the minibatch as before. Instead, a running average is used which was calculated during training. Typically, the output of a batch norm layer is not simply H' but $\gamma H' + \beta$, where γ and β are trainable parameters. This allows the network to have any mean and standard deviation (Goodfellow et al., 2016).

2.1.5. Autoencoders

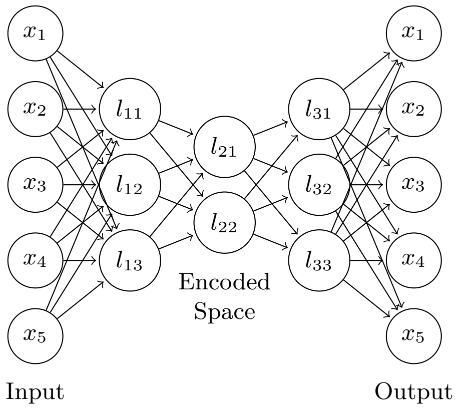
With the theory introduced in the previous subsections, an infinite number of different architectures can be designed. Here, I restrict myself to one class of networks, the so-called autoencoders (AE), as they will be used in this thesis.

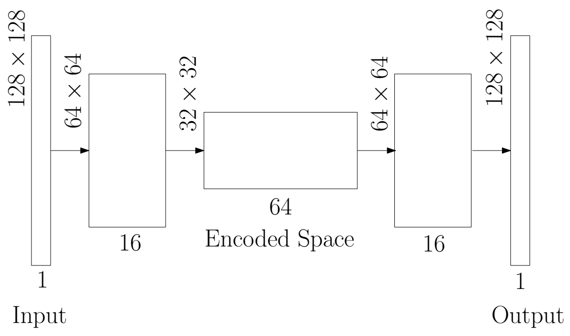
An autoencoder is designed to reconstruct the input as the output. This might seem like a useless application, but typically not the output, but the information in the hidden layer is of interest (Goodfellow et al., 2016). In Figure 2.5a the sketch of the architecture of an autoencoder is displayed. The network consists of two parts: The encoder maps the input x into an encoded representation $q = f(x)$ and the decoder tries to restore the input from the hidden representation, $y = r(q)$ (Goodfellow et al., 2016). To encourage the network to learn a meaningful encoded representation of the input data, the hidden space must be smaller than the input space. If a linear decoder and an MSE loss are used to train the network, the encoder will produce an encoded representation that is simply the principle component analysis (PCA) of the data (Bourlard and Kamp, 1988). Using a nonlinear decoder will give a nonlinear generalization of the PCA in the hidden space, while the network is only trained on copying the input (Goodfellow et al., 2016). The traditional applications of autoencoders are dimensionality reduction and feature learning (Goodfellow et al., 2016). Autoencoders are trained in an unsupervised way, which means that the data do not have labels. However, they find application as well in supervised tasks such as classification: Once the network has learned a meaningful representation of the data, the encoder can be used separately from the decoder to encode the data. The code can then be used as input to a supervised classification model, with the advantage that the classification network gets data already compressed down to its most important information (Gogoi and Begum, 2017).

Autoencoders also work for grid-like data such as time series and images. The fully-connected layers are simply replaced with convolutional layers as discussed in section 2.1.4 (Masci et al., 2011), see Figure 2.5b. Still, the encoded space has a smaller dimension than the input. To compensate for the spatial compression of the input, the number of channels is increased. Different channels can be seen as 2D maps, showing where certain features can be found in the input (Goodfellow et al., 2016).

2.1.5.1. U-Net

As the encoder of an autoencoder compresses the input to smaller spatial dimensions, high-resolution location information will get lost. For traditional tasks, such as data compression, a very high accuracy in the reconstruction is not important. However, when using those networks outside of their traditional applications, such as in regression and segmentation tasks, a high accuracy of the output is crucial. To overcome this problem, skip connections between the encoder and decoder are introduced. They enable the transport of location information from the encoder to higher resolution layers of the decoder (Ronneberger et al., 2015).





(b) Convolutional autoencoder (CAE)

(a) Stacked autoencoder (SAE)

Figure 2.5.: Sketches of the architecture of autoencoders. For the convolutional autoencoder, the arrows correspond to 3x3 convolution. The numbers below the boxes depict the number of channels and the numbers on the sides the size of the feature maps.

This architecture was introduced by Ronneberger et al. (2015) as the U-Net, which was implemented for biomedical segmentation tasks. Figure 2.6 shows the architecture of the U-Net, where the origin of its name can be easily seen, as the network is U-shaped. Similar to an autoencoder, the U-Net consists of a contracting and an expanding part, but it uses skip connections between the two parts. This improves the correct localization of class labels in the output (Ronneberger et al., 2015). In the original U-Net max pooling is used to decrease the input size. Max pooling (Zhou and Chellappa, 1988) writes the maximum value in a sliding window to the output. To undo the pooling in the expansive part, up-convolutions are used. An up-convolution is the transpose of a convolution (Dumoulin and Visin, 2016). Nowadays, the idea of the U-Net with slight variations is used for various different tasks, also outside segmentation. In geophysics, variants of the U-Nets can, among others, detect faults (Liu et al., 2020), which is indeed a segmentation task, invert seismic data for seismic velocities (Yang and Ma, 2019), pick P- and S-wave arrivals of earthquakes (Zhu and Beroza, 2019) or extrapolate low frequencies (Aharchaou and Baumstein, 2020; Jin et al., 2021; Ovcharenko et al., 2020; Nakayama and Blacquière, 2021; Wang et al., 2020).

2.1.5.2. Residual U-Net

A variation of the U-Net is a residual U-Net also called UResNet or Res-U-Net, which was also introduced for the first time in the field of biomedical image segmentation by Guerrero et al. (2018) and Xiao et al. (2018). Compared to the U-Net as discussed in section 2.1.5.1, it has ResNet blocks instead of convolutional layers. The architecture of a ResNet block is shown in Figure 2.7. ResNet blocks consist of convolutional layers but have shortcuts connecting shallower parts of the network directly with deeper parts. The operation in the shortcut is in the ResNet layer a simple identity mapping (He et al., 2016). The shortcut connections have an important advantage: The convolutional layers, where the trainable parameters can be found, now do not have to learn a representation $H(x)$, but only the residual $F(x) = H(x) - x$, which improves the training (He et al., 2016). In contrast, skip connections do not add on the data but concatenate the data from the encoder as additional channels to the decoder (Ronneberger et al., 2015).

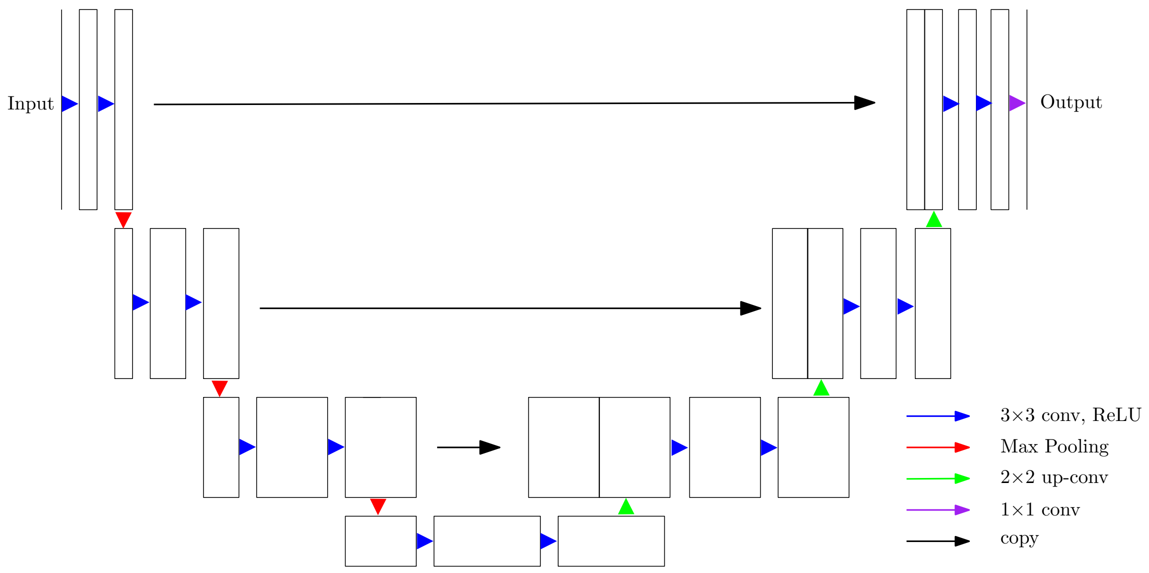


Figure 2.6.: Architecture of the U-Net after Ronneberger et al. (2015). The original U-Net is one layer deeper but for display reasons only 3 layers are shown. The black arrows depict the characteristic skip-connections.

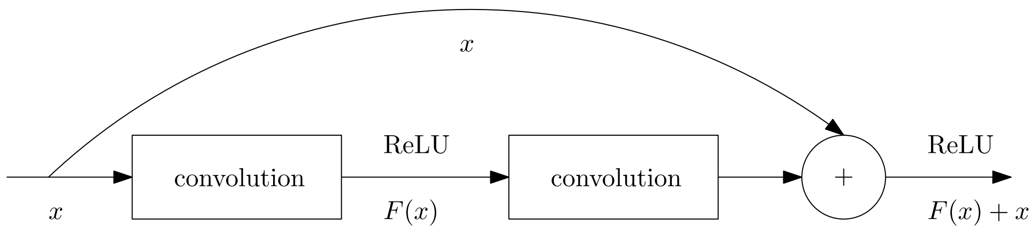


Figure 2.7.: ResNet Block after He et al. (2016). With the shortcuts that are an identity mapping, the network only has to learn the residual $F(x)$.

2.2. Seismic modeling

In this thesis, the network is trained and applied on seismic data. Therefore, the basics of seismic wave propagation in acoustic media will be discussed in the following. Afterwards, the finite-difference method for seismic modeling is introduced. This information will be needed for generating training data and the application of the results in full-waveform inversion.

2.2.1. Seismic wave propagation in acoustic media

For infinitesimal deformations, the linear elasticity theory is valid and we assume a linear relation between stress and strain, also known as Hooke's law , using Einstein summation convention:

$$\sigma_{ij} = C_{ijkl}\epsilon_{kl}, \quad i,j,k,l \in [1,3] \ , \tag{2.37}$$

where σ is the stress tensor, ϵ is the strain tensor, and the stiffness tensor C contains the elasticity moduli (Lay and Wallace, 1995). The number of elasticity moduli can be reduced to 21 due to the symmetry properties of σ, ϵ and C and the conservation of volume density of elastic energy (Lay and Wallace, 1995). For isotropic elastic media, the number further decreases to two independent parameters, known as the Lamé parameters λ and μ. With that, the stress-strain relation for isotropic media can be written as

$$\sigma_{ij} = \lambda\theta\delta_{ij} + 2\mu\epsilon_{ij} \ , \tag{2.38}$$

$$\epsilon_{ij} = \frac{1}{2}\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right) \ , \tag{2.39}$$

where θ is the cubic dilatation and can be calculated as the trace of ϵ (Lay and Wallace, 1995). To describe wave propagation, not only the stress-strain relation is needed but also the equation of motion, which can be derived from Newton's second law:

$$\rho\frac{\partial v_i}{\partial t} = f_i + \frac{\partial \sigma_{ij}}{\partial x_j} \ , \tag{2.40}$$

where we consider that $\frac{\partial^2 u_i}{\partial t^2} = \frac{\partial v_i}{\partial t}$ (Lay and Wallace, 1995). The left-hand side is the inertial force and the right-hand side corresponds to external body forces and stress gradients in the medium (Lay and Wallace, 1995). Taking the time derivative of equation 2.38, leads to the final equation for the velocity-stress formulation of wave propagation:

$$\frac{\partial \sigma_{ij}}{\partial t} = \lambda\frac{\partial\theta}{\partial t}\delta_{ij} + 2\mu\frac{\partial\epsilon_{ij}}{\partial t} \tag{2.41}$$

With equations 2.40 and 2.41 elastic wave propagation is expressed. In this thesis only acoustic media are considered. Hence, only pressure waves are propergating in a medium and no shear waves exist. From this follows that the second Lamé parameter, the shear modulus, is zero and the resulting seismograms, as well as the used equations, simplify. With hydrostatic stress or pressure p defined as the mean of normal stress σ_{ii}, so $-p = \frac{1}{3}\sigma_{ii}$, the stress-strain relation can be rewritten as

$$\sigma_{ij} = \lambda\theta\delta_{ij} = \lambda\delta_{ij}\epsilon_{nn} = \lambda\delta_{ij}\frac{\partial u_n}{\partial x_n} = -p\delta_{ij} \ . \tag{2.42}$$

By inserting equation 2.42 into equations 2.40 and 2.41, we can simplify the equation of motion:

$$-\rho\frac{\partial v_i}{\partial t} = f_i + \frac{\partial p}{\partial x_i} \tag{2.43}$$

With equation 2.41 the time derivative of the pressure is

$$\frac{\partial p}{\partial t} = -\lambda\frac{\partial\epsilon_{nn}}{\partial t} = -\lambda\frac{\partial v_n}{\partial x_n} \ . \tag{2.44}$$

Equation 2.44 is the first-order acoustic wave equation.

2.2.2. Finite-difference method

In the following, only equations for the 2D case are discussed, as modeling will be done in two dimensions only. The equation of motion (equation 2.43) needs to be discretized in order to be able to solve it numerically. Using a constant grid spacing Δh in both dimensions and a time sampling Δt, the parameters are now only defined at discrete coordinates $(x, y, t) = (k \cdot \Delta h, l \cdot \Delta h, n \cdot \Delta t)$ (Köhn, 2011). The basic idea of the finite-difference (FD) method is to approximate the derivatives by finite differences (Moczo et al., 2004). For a function $f(x, t)$ that has continuous derivatives in space and time, we can express the derivatives on a discrete grid as

$$\frac{\partial f(x,t)}{\partial t} \approx \frac{f(x, t+\Delta t) - f(x,t)}{\Delta t} \,, \tag{2.45}$$

$$\frac{\partial f(x,t)}{\partial x} \approx \frac{f(x+\Delta h, t) - f(x,t)}{\Delta h} \,. \tag{2.46}$$

To ensure that the partial differential equations refer to the same grid point, a staggered grid is used, where some model parameters are shifted by half the grid spacing (Virieux, 1986; Levander, 1988). Figure 2.8 illustrates the staggered grid and the positioning of the model parameters. Furthermore, the pressure wavefield is computed on different time steps than the velocity fields. With equation 2.45 applied to the parameters v and p we get expressions for the second-order time derivatives:

$$\left.\frac{\partial v_i}{\partial t}\right|^{n+\frac{1}{2}} = \frac{v_i^{n+1} - v_i^n}{\Delta t} \tag{2.47}$$

$$\left.\frac{\partial p}{\partial t}\right|^{n} = \frac{p^{n+\frac{1}{2}} - p^{n-\frac{1}{2}}}{\Delta t} \,, \tag{2.48}$$

where n indicates the time step. By substituting the time derivatives in equations 2.43 and 2.44, we can express the velocity and pressure at the next time step using parameters from earlier time steps:

$$v_i^{n+1} = v_i^n - \frac{\Delta t}{\rho} \left.\frac{\partial p}{\partial x_i}\right|^{n+\frac{1}{2}} - \frac{\Delta t}{\rho} f_i^{n+\frac{1}{2}} \tag{2.49}$$

$$p^{n+\frac{1}{2}} = p^{n-\frac{1}{2}} - \Delta t \cdot \lambda \left.\frac{\partial v_i}{\partial x_i}\right|^{n} \tag{2.50}$$

The equations are valid for a force source. If a pressure source S_p is used, the source signal is added to the pressure term:

$$v_i^{n+1} = v_i^n - \frac{\Delta t}{\rho} \left.\frac{\partial p}{\partial x_i}\right|^{n+\frac{1}{2}} \tag{2.51}$$

$$p^{n+\frac{1}{2}} = p^{n-\frac{1}{2}} - \Delta t \cdot \lambda \left.\frac{\partial v_i}{\partial x_i}\right|^{n} + \Delta t \cdot \lambda S_p^n \tag{2.52}$$

Furthermore, the derivatives in space need to be considered as well. The second-order expression is of similar style as the time derivatives:

$$\left.\frac{\partial p}{\partial x}\right|_{k+\frac{1}{2},l}^{n+\frac{1}{2}} = \frac{p_{k+1,l}^{n+\frac{1}{2}} - p_{k,l}^{n+\frac{1}{2}}}{\Delta h} \,. \tag{2.53}$$

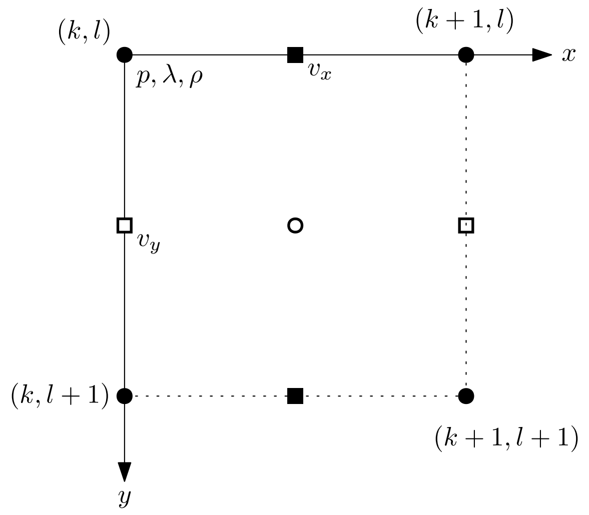


Figure 2.8.: 2D staggered grid as suggested by Virieux (1986) and Levander (1988).

However, typically higher-order operators are used because they increase the accuracy of the simulation:

$$\left[\sum_{k=1}^{N}\beta_k(2k-1)\right]\left.\frac{\partial p}{\partial x}\right|_{k+\frac{1}{2},l}^{n+\frac{1}{2}} = \frac{1}{\Delta h}\sum_{k=1}^{N}\beta_k\left(p_{k+1,l}^{n+\frac{1}{2}} - p_{k,l}^{n+\frac{1}{2}}\right)$$
$$+\frac{1}{\Delta h}\sum_{k=1}^{N}\sum_{l=1}^{N}\beta_k\,\frac{((k-\frac{1}{2})\Delta h)^{2l-1}}{(2l-1)}\left.\frac{\partial^{(2l-1)}p}{\partial x^{(2l-1)}}\right|_{k+\frac{1}{2},l}^{n+\frac{1}{2}} + \mathcal{O}(\Delta h)^{2N}\,, \tag{2.54}$$

where β_k are the FD coefficients (Köhn, 2011). With a higher-order operator not only the neighboring grid points are considered, but also N adjacent points. For the eighth order in space, the next four adjacent points in each direction are used. The term of the direct neighbors has a larger FD coefficient than the one of the grid points farther away. In a similar way, $\frac{\partial v}{\partial x}$ is expressed, also usually in higher order in space. The time derivative is usually kept as a second-order operator. By combining equations 2.51 and 2.52 with the spatial derivatives, we get the FD scheme for the wave equation:

$$p_{k,l}^{n+\frac{1}{2}} = p_{k,l}^{n-\frac{1}{2}} - \Delta t\cdot\lambda_{k,l}\left(\left.\frac{\partial v_x}{\partial x}\right|_{k,l}^{n} + \left.\frac{\partial v_y}{\partial y}\right|_{k,l}^{n}\right) + \Delta t\cdot\lambda S_p^n \tag{2.55}$$

$$v_{x,k+\frac{1}{2},l}^{n+1} = v_{x,k+\frac{1}{2},l}^{n} - \frac{\Delta t}{\rho_{k+\frac{1}{2},l}}\left.\frac{\partial p}{\partial x}\right|_{k+\frac{1}{2},l}^{n+\frac{1}{2}} \tag{2.56}$$

$$v_{y,k,l+\frac{1}{2}}^{n+1} = v_{y,k,l+\frac{1}{2}}^{n} - \frac{\Delta t}{\rho_{k,l+\frac{1}{2}}}\left.\frac{\partial p}{\partial y}\right|_{k,l+\frac{1}{2}}^{n+\frac{1}{2}} \tag{2.57}$$

As shown in Figure 2.8, ρ is only defined on full grid points, but for the equation of motion, results at half grid points are needed. To obtain a stable result, those points need to be arithmetically averaged between the two neighboring grid points:

$$\rho_{k+\frac{1}{2},l} = \frac{\rho_{k,l}+\rho_{k+1,l}}{2} \tag{2.58}$$

The value of $\rho_{k,l+\frac{1}{2}}$ is estimated similarly (Bohlen and Saenger, 2006).

2.2.2.1. Numerical dispersion and instabilities

The grid spacing Δh and the time sampling Δt need to be chosen carefully to avoid numerical dispersion and instability. For the grid spacing we have the trade-off between a fine sampling which gives more accurate results, and a coarser sampling which is less computationally expensive. According to Nyquist's theorem, at least two grid points per wavelength are needed to be able to describe the wave correctly. However, usually more points per wavelength are recommended. The theoretical largest grid size can be calculated by

$$\Delta h \leq \frac{\lambda_{min}}{n} = \frac{v_{min}}{n f_{max}} , \tag{2.59}$$

where λ_{min} is the minimum wavelength, which can be calculated from the minimum velocity v_{min} divided by the maximum frequency f_{max} (Köhn, 2011). The factor n is the number of grid points per minimal wavelength and depends on the FD order, see Table 2.1 (Köhn, 2011). To ensure the stability of the simulation, the time sampling has to be chosen such that it fulfills the Courant-Friedrichs-Lewy criterion (Courant et al., 1928), here for the 2D case:

$$\Delta t \leq \frac{\Delta h}{\gamma \sqrt{2} v_{max}} , \tag{2.60}$$

where γ is the sum of the FD coefficients, i.e., it also depends on the FD order, see Table 2.2. The factor $\sqrt{2}$ is determined by the dimension D of the simulation, so in a 2D simulation $D = 2$. Furthermore, the maximum velocity of the model v_{max} has an influence on the time sampling.

2.2.2.2. Boundary conditions

At the model boundaries, absorbing boundaries are implemented to avoid artificial reflections. The basic idea is to dampen waves that come close to the model boundaries. One of the methods are perfectly matched layers (PML) introduced by Komatitsch and Martin (2007). The perfectly matched layers stretch the coordinates of the wave equation in the frequency domain, leading to exponentially decaying plane waves. As PMLs are only reflectionless for an exact solution of the wave equation but not for an FD approach, an additional damping function is used (Köhn, 2011).

2.3. Full-waveform inversion

Full-waveform inversion (FWI) was first introduced by Tarantola (1984b) and aims to solve the inverse problem, that is, to find a model m that describes the observed data d_{obs}. The opposite problem, finding the data for a given model, is straightforward, because they can be modeled as discussed in section 2.2.2. In the inverse problem, however, there can be several models that describe the data, but the optimal model is searched for. Traditional inversion methods like tomography do not use all the information contained in the data, but only parts of it, e.g. traveltimes. In FWI, however, the entire information of the waveform is used, including amplitudes.

In the following, the theory behind FWI with the adjoint-state method will be discussed and a typical FWI workflow will be presented.

Table 2.1.: Number of grid points n per minimum wavelength dependent on the FD order for Taylor coefficients (Köhn, 2011).

FD order	2	4	6	8
n	12	8	6	5

Table 2.2.: Factor γ for the Courant criterion dependent on the FD order for Taylor coefficients (Köhn, 2011).

FD order	2	4	6	8
γ	1	$\frac{7}{6}$	$\frac{149}{120}$	$\frac{2161}{1680}$

2.3.1. Misfit function

Similar to training neural networks, full-waveform inversion is an optimization problem. Starting from a model m, synthetic data d_{syn} are modeled, just as discussed in section 2.2.2, which can be seen as applying a nonlinear forward operator $\mathcal{F}(\cdot)$ to the model m:

$$d_{syn} = \mathcal{F}(m) \tag{2.61}$$

When interpreting equation 2.61 with neural networks in mind, m corresponds to the input, d_{syn} to the output and $\mathcal{F}(\cdot)$ to a feedforward network. The misfit between the synthetic data and the observed data d_{obs}, in machine learning jargon the target or label, can be expressed by an objective function $J(d_{obs}, d_{syn})$. The commonly used misfit function in FWI is an L2 norm of the data residual $\Delta d = d_{syn} - d_{obs}$:

$$J(m) = \frac{1}{2} \sum_s \sum_t \sum_r \|\Delta d(x_r, x_s, t, m)\|^2 \ , \tag{2.62}$$

where s accounts for all source positions, r for all receiver positions and t for all steps in time. Alternatively, also a normalized L2 norm is possible:

$$J_{norm} = \frac{1}{2} \sum_s \sum_r \sum_t \left(\frac{d_{syn}}{\|d_{syn}\|} - \frac{d_{obs}}{\|d_{obs}\|} \right)^2 \ , \tag{2.63}$$

(Choi and Alkhalifah, 2012). To find the optimal model the misfit function needs to be minimized. Similar to the training of feed-forward neural networks, a gradient-based approach is used to solve the optimization problem. Starting at an initial model m_0 the gradient of J with respect to m is calculated. Then the model is updated as follows:

$$m_{i+1} = m_i - \alpha P_i \nabla_m J(m_i) \ , \tag{2.64}$$

where α is the step length, or learning rate in machine learning, and P_i a preconditioning operator which approximates the inverse Hessian matrix of the misfit function (Köhn, 2011). The preconditioning operator can be used, for example, for the tapering around source and receiver positions (Köhn, 2011). By comparing equation 2.64 with equation 2.13, we can see that in machine learning we update the trainable parameters θ, which are part of the forward operator, but in FWI we update the model, which would correspond to the input for traditional neural networks.

2.3.2. Calculating the gradient

For calculating the gradient, the adjoint state method is used as it only requires two forward simulations (Plessix, 2006). The method consists of three basic steps. First, the forward wavefield is modeled to obtain the observed data d_{obs}. Then the data residual $\Delta d = d_{syn} - d_{obs}$ is calculated. The residual is used to calculate the adjoint wavefield by forward modeling with the time-reversed residuals at the receiver locations as sources. Therefore, this wavefield is also called the backpropagated residual wavefield. This is the second forward simulation. Now the cross-correlation between the forward and adjoint wavefields is calculated, giving the gradient of the objective function (Tarantola, 1984a; Plessix, 2006).

2.3.3. Multi-scale approach

With FWI being a highly nonlinear optimization problem, the misfit function has multiple local minima (Bunks et al., 1995). Optimization algorithms might get stuck in a local minimum instead of the global minimum, which is called cycle-skipping. This might occur when the initial model is too far away from the true model and the observed and modeled data are out of phase by more than half a period. To mitigate the cycle-skipping problem, Bunks et al. (1995) introduced the multiscale approach. While the original publication suggested to go from large to fine grid scales, nowadays the inversions are done in different, increasing frequency bands. In Figure 2.9 the idea behind the multistage approach is shown. Using low frequencies in the beginning smoothens the misfit function, which allows the optimizer to converge towards the minimum. The model parameters found in this way are then used as initial guess for the next frequency band. This repeats for all frequency bands. In the end, the low frequencies give the information for a smooth model and with increasing frequencies, more and more high-frequency details can be resolved. With the multistage approach also the importance of the low frequencies becomes obvious. Without low frequencies present in the data, the inversion does not start at the misfit curve at the bottom of Figure 2.9, but in any of the other curves. If the initial guess is not good enough, we might end up in a local minimum already in the first stage, which will prevent the convergence towards the global minimum in later stages.

2.3.4. FWI workflow

With the theory in mind the actual workflow of FWI is briefly discussed in the following. Figure 2.10 displays the schematic of the workflow. To start the FWI, the observed data d_{obs} and an initial model m_0 are needed. The forward problem as discussed in section 2.2.2 is solved for the initial model to get the synthetic data d_{syn} and the wavefield $u(x, y, t)$. The residuals Δd between the synthetic and the observed data are calculated. The residuals are then backpropagated to obtain the the adjoint wavefield $\Psi(x, y, t)$. Then the model update Δm is calculated based on $u(x, y, t)$ and $\Psi(x, y, t)$. The gradient is preconditioned and the conjugate gradient is used for the update. With the estimated step length μ_n, the model can be updated with $m_{n+1} = m_n - \mu_n \Delta m$. If the misfit is smaller than a certain threshold or the maximum number of iterations is reached, the inversion will stop. Otherwise, the next iteration will start with the new initial model m_{n+1}.

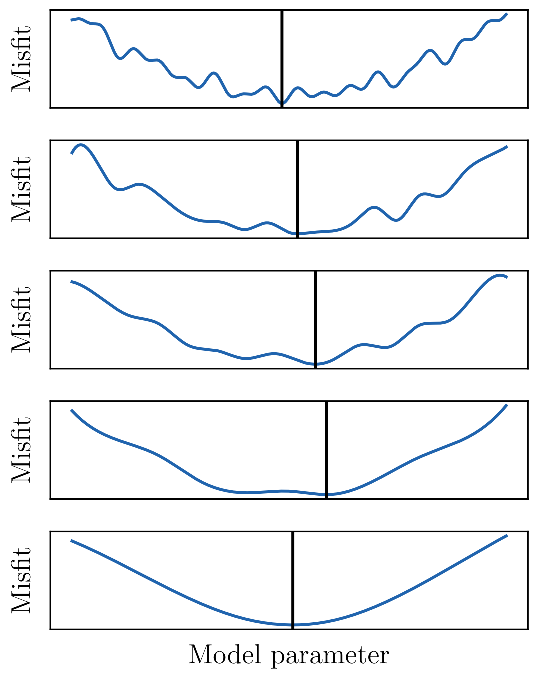


Figure 2.9.: Sketch for a 1D misfit function for different frequency bands in the multiscale approach after Bunks et al. (1995). The frequency content decreases from top to bottom. Vertical lines mark the global minimum of each misfit function.

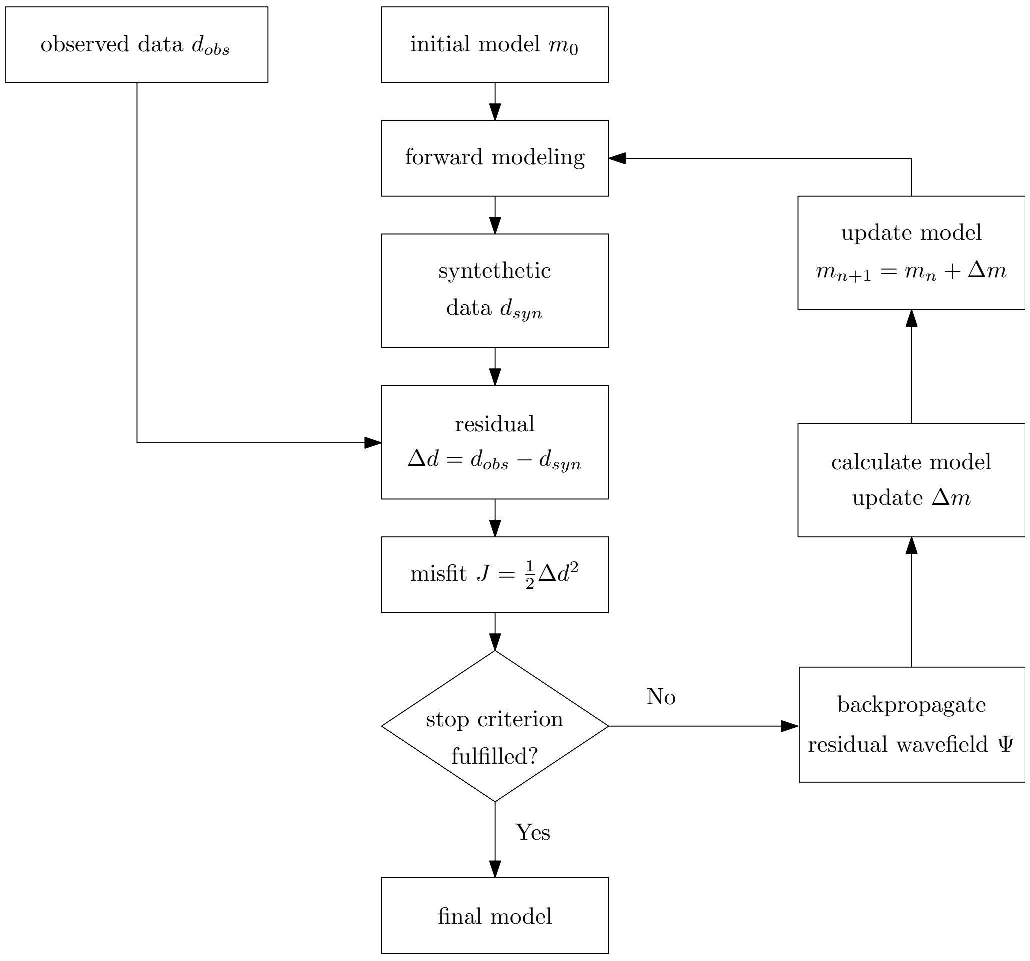

observed data d_{obs}
initial model m_0
forward modeling
syntethetic data d_{syn}
update model $m_{n+1} = m_n + \Delta m$
residual $\Delta d = d_{obs} - d_{syn}$
calculate model update Δm
misfit $J = \frac{1}{2}\Delta d^2$
stop criterion fulfilled?
No
backpropagate residual wavefield Ψ
Yes
final model

Figure 2.10.: Flowchart for full-waveform inversion.

3. Data generation and preparation

A sufficiently large amount of training data is needed to train a neural network. As such a large amount of field data with the required characteristics is often not available, this demand is fulfilled by generating synthetic data. Generating synthetic data has the additional advantage that the frequency content of the data can be shaped as needed. This chapter presents the generation of velocity models, discusses the acquisition design for seismic modeling and explains the development of a pre-processing routine.

3.1. Model building

Subsurface models are needed to simulate synthetic data. In the related literature, these models are generated using different methods. Some authors model on pre-existing data sets, mostly benchmark data sets (Wang et al., 2020; Fang et al., 2020; Ovcharenko et al., 2020; Fabien-Ouellet, 2020b; Jin et al., 2021). Sun and Demanet (2020b, 2022) use benchmark models as well, but extract random patches from these models to simulate the wave propagation in smaller and more varying models. Other authors create their own synthetic velocity models, differentiating between random velocity models (Ovcharenko et al., 2017, 2018b, 2019) and random models that follow a user-defined trend (Nakayama and Blacquière, 2021; Ovcharenko et al., 2022, 2021). Hu et al. (2021, 2020) create a single realistic model for training. The networks of Aharchaou et al. (2021) and Aharchaou and Baumstein (2020) are trained solely on field data. Therefore, they can skip the process of modeling synthetic data.

This work follows the approach of Nakayama and Blacquière (2021) and Ovcharenko et al. (2022) and generates models randomly but with a defined trend to keep the models geologically plausible. With a future application to field data in mind, the synthetic models are chosen to resemble the subsurface structures around the Asse II salt mine (Pollok et al., 2018). A smooth model taken from Şortan (2022, Figure 3.1) serves as a starting point for the generation of several different subsurface models. In contrast to the real model, the synthetic models have a water column as first layer, which acts as a half space to avoid the unfavorable effects of a free surface on the frequency content of the modeled data. The implemented algorithm generates random variations of the initial model. First, a rough outline of the salt body, where the P-wave velocity $v_p > 4000\,\mathrm{m/s}$, is extracted from the smooth model. To obtain a less complex wavefield, a simpler model is preferred. Therefore, the surface topography is neglected and a flat surface is used instead.

The first step towards a varying model is to alter the appearance of the top salt. The basic concept used to create a random interface is depicted in Figure 3.2. Nine nodes are defined equally spaced along the salt outline. To alter the model, the nodes are randomly moved vertically and horizontally in a given range, see Table 3.2. To avoid a lot of energy getting reflected out of the model, an additional constraint is set for the two nodes at the model borders: The two nodes are forced to be shallower than their neighboring point, which

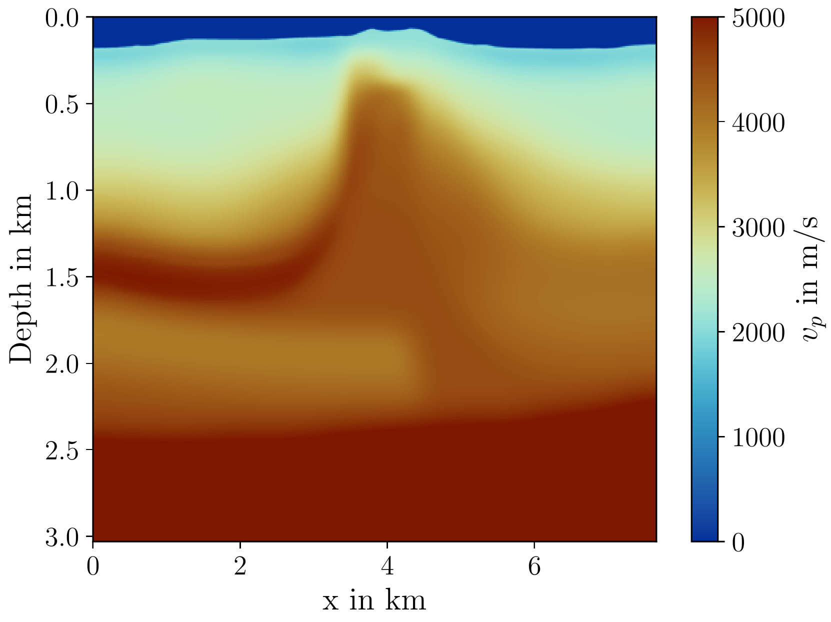
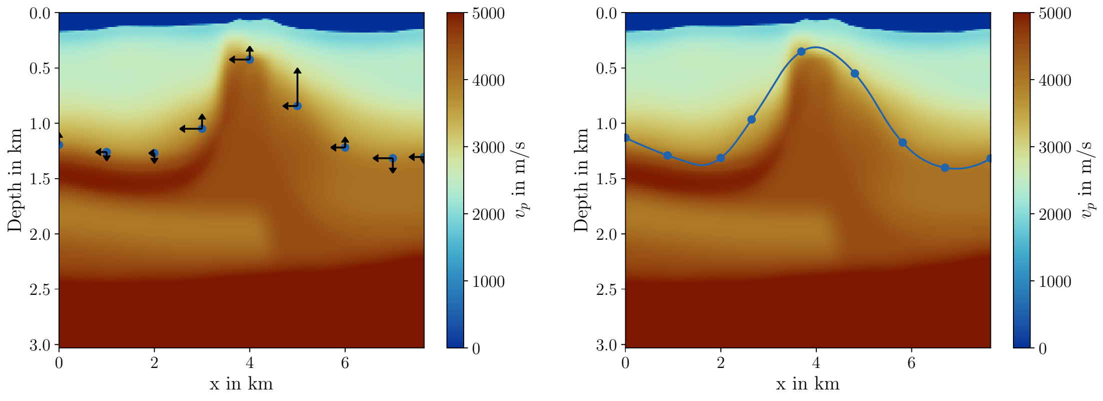


Figure 3.1.: Original smooth v_p velocity model of the salt dome at Asse II, taken from Șortan (2022). Note that this model has a strong surface topography.



(a) **(b)**

Figure 3.2.: Used concept for generating a random interface: (a) Define nine nodes equally spaced along $v_p \leq 4000\,\mathrm{m/s}$. Randomly move each point. (b) Interpolation between the new nodes gives the outline of the interface.

results in a reflector dipping down towards the central part of the model. Then the grid points between the nodes are interpolated using a quadratic interpolation.

Next, the overlying interfaces are constructed similarly. A number of one to four interfaces above the salt structure can be chosen, where the initial depth of the interface depends on the depth of the top of the salt structure. Then the nodes are placed again along the initial interface structure, thereby making sure one point is above the minimum depth of the salt. Next, the points are moved randomly in a certain value range, see Table 3.2. The values between the nodes are found by cubic interpolation. Underneath the salt structure, a dipping layer is added; here, only the two points at the boundary are randomly varied and all points in between are linearly interpolated. Finally, a horizontal benchmark interface is added to the model at depth 2.9 km.

Now that all interfaces are constructed, the velocities need to be assigned to the layers. Each layer has a constant velocity value, assigned as shown in Table 3.1. This velocity is randomly varied in the range of the perturbation given in Table 3.2. Only the velocity of the water layer and the layer below the horizontal interface is kept constant at 1500 m/s and 4000 m/s, respectively.

Besides P-wave velocities also a density model is needed to model the data. For simplicity, the density is kept constant throughout the model with a density value of 2000 kg/m^3. For the simulation of acoustic media, the S-wave velocity is set to zero throughout the entire model. In total, 30 different models are generated following the algorithm described above. Some exemplary models can be seen in Figure 3.3. Although all models have similarities, they differ in the number of layers, the seismic velocities, and the shape of the salt structure and other interfaces.

3.2. Modeling training data

The training data are generated with the SOFI2D finite-difference (FD) modeling software (Bohlen et al., 2016). Only acoustic waves are modeled to keep seismograms simple. The generated v_p models and the ρ model are given as input. Before the actual simulation, the parameters must be defined. A flat-spectrum wavelet is chosen as the source signature, as it has a constant amplitude spectrum in a given frequency range, see Figure 3.4b. Therefore, the low frequencies, which are the focus of this work, have the same energy as the high frequencies. The used flat-spectrum wavelet is the autocorrelation of a linear sweep of 10 s length from 5 to 30 Hz, with a 1 s taper at the beginning and the end. A Blackman window is applied to the waveform as an additional taper. Figure 3.4a displays the resulting waveform and the waveform with the applied Blackman window. Using the Blackman window leads to a reduced number of undulations at the side lobes, which results in clearer seismograms. The resulting zero-phase wavelet is shifted in time by 0.2 s to serve as a source wavelet for FD modeling.

Table 3.1.: P-wave velocities v_i in m/s for different numbers of interfaces i above the salt. v_2 is the velocity of the second layer and so on, with $v_1 = 1500$ m/s being the velocity of the water layer.

# Interfaces	v_2 in m/s	v_3 in m/s	v_4 in m/s	v_5 in m/s
1	1900			
2	1900	2900		
3	1900	2800	3300	
4	1900	2400	3000	3500

Table 3.2.: Value ranges for the variable parameters of the model-generating algorithm.

Parameter	Lower Limit	Upper Limit
Layer velocity perturbation	$175\,\frac{\mathrm{m}}{\mathrm{s}}$	$175\,\frac{\mathrm{m}}{\mathrm{s}}$
Salt velocity	$4600\,\frac{\mathrm{m}}{\mathrm{s}}$	$5000\,\frac{\mathrm{m}}{\mathrm{s}}$
Dipping layer velocity	$3400\,\frac{\mathrm{m}}{\mathrm{s}}$	$3800\,\frac{\mathrm{m}}{\mathrm{s}}$
x-position salt nodes perturbation	$-500\,\mathrm{m}$	$500\,\mathrm{m}$
y-position salt nodes perturbation side lobes	$-50\,\mathrm{m}$	$50\,\mathrm{m}$
y-position salt nodes perturbation peak	$-300\,\mathrm{m}$	$100\,\mathrm{m}$
x-position layer perturbation	$-50\,\mathrm{m}$	$50\,\mathrm{m}$
y-position layer perturbation	$-100\,\mathrm{m}$	$50\,\mathrm{m}$
y-position dipping layer nodes	$2200\,\mathrm{m}$	$2600\,\mathrm{m}$

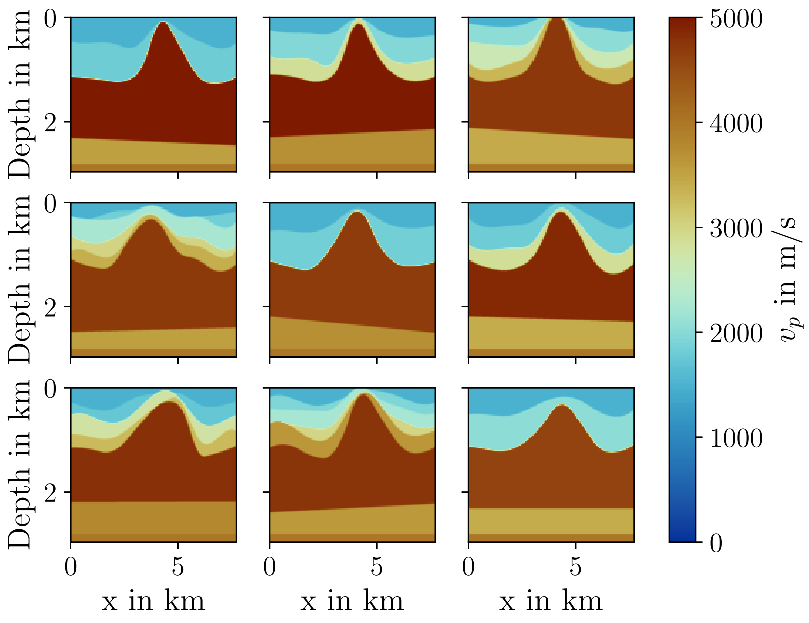
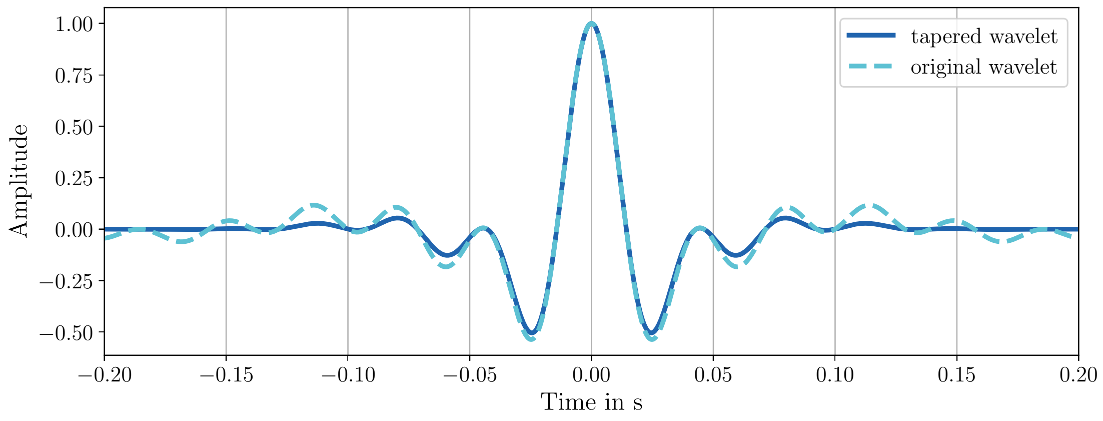
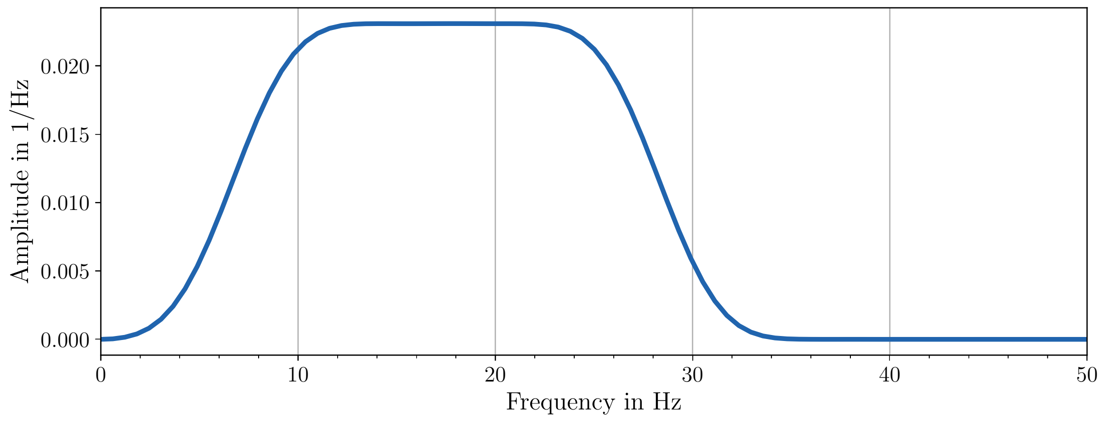


Figure 3.3.: Examples of different randomly generated v_p velocity models used for simulating shot gathers.



(a) Waveform



(b) Amplitude spectrum

Figure 3.4.: (a) Waveform and (b) amplitude spectrum of the tapered flat-spectrum wavelet which is used as source signature. The flat-spectrum wavelet is the autocorrelation of a linear sweep of 10 s length from 5 to 30 Hz, with a 1 s taper at the beginning and the end.

To minimize grid dispersion, a maximum grid size can be estimated using equation 2.59 and the values of Table 2.1 and the FD operator of eighth order:

$$\frac{v_{min}}{f_{max} \cdot n} = \frac{1800\,\frac{\mathrm{m}}{\mathrm{s}}}{30\,\mathrm{Hz} \cdot n} = \frac{60\,\mathrm{m}}{5} = 12\,\mathrm{m} \tag{3.1}$$

Here, the FD operator of order eight is chosen, as it has a high accuracy and coarser grids can be considered. To find a reasonable grid size, a convergence test is done. For that, a single shot is modelled for various grid sizes $\Delta h = 2, 4, 6, 8, 10\,\mathrm{m}$. With equation 2.60, the maximum allowed time step for each grid size can be calculated as

$$\Delta t \leq \frac{\Delta h}{\frac{2161}{1680} \cdot \sqrt{2} \cdot 5000\,\frac{\mathrm{m}}{\mathrm{s}}} \approx \begin{cases} 0.22\,\mathrm{ms}, & \text{for } \Delta h{=}2\,\mathrm{m} \\ 0.44\,\mathrm{ms}, & \text{for } \Delta h{=}4\,\mathrm{m} \\ 0.66\,\mathrm{ms}, & \text{for } \Delta h{=}6\,\mathrm{m} \\ 0.88\,\mathrm{ms}, & \text{for } \Delta h{=}8\,\mathrm{m} \\ 1.10\,\mathrm{ms}, & \text{for } \Delta h{=}10\,\mathrm{m} \end{cases} . \tag{3.2}$$

The shot is modeled with a time step of $0.2, 0.4, 0.6, 0.8, 1.0\,\mathrm{ms}$, respectively. By comparing the waveforms of a single trace (Figure 3.5), it can be seen that for the first arrival the waveforms of all grid sizes match quite well. However, at $\sim 0.78\,\mathrm{s}$ the trace calculated with grid size $10\,\mathrm{m}$ is out of phase. At time step $t \approx 0.9\,\mathrm{s}$, all traces show different waveforms. The traces modelled with smaller grid spacings show fewer differences in their waveforms. The trace calculated with grid spacing $4\,\mathrm{m}$ is closest to the one with $2\,\mathrm{m}$ grid spacing but requires less computing time, see Table 3.3. Therefore, a grid spacing of $4\,\mathrm{m}$ is chosen and the corresponding time sampling $\Delta t = 0.4\,\mathrm{ms}$.

To cross-check whether the output wavelets of the simulation match the source wavelets, the frequency spectra are compared for a homogeneous model. The first layer of the subsurface modes is water; therefore, an explosive source is used, which emits only P-waves just like an airgun in marine acquisition. This is sufficient, as only acoustic waves are used. Hydrophones are used as receivers that record pressure changes. In Figure 3.6 the resulting waveform and its spectrum are displayed. It can be seen that the output waveform does not match the input flat-spectrum wavelet and also that the amplitude spectrum has changed. In the output, more energy is given to the higher frequencies, which is unfavorable for the task of low-frequency extrapolation. A 1.5-integration of the input wavelet and a 180° phase shift solve the issue and the integrated wavelet matches the input wavelet, see Figure 3.6.

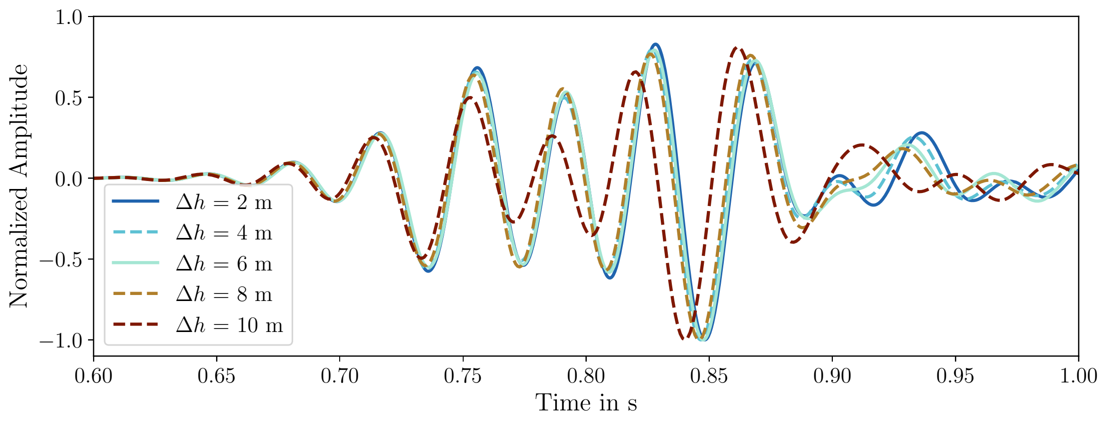


Figure 3.5.: Convergence test: Same trace modelled with different grid sizes Δh. The sampling interval is adjusted for each grid size.

Table 3.3.: Computing time in seconds for a similar setting but with different grid spacings Δh. The sampling intervals are adjusted with the grid size.

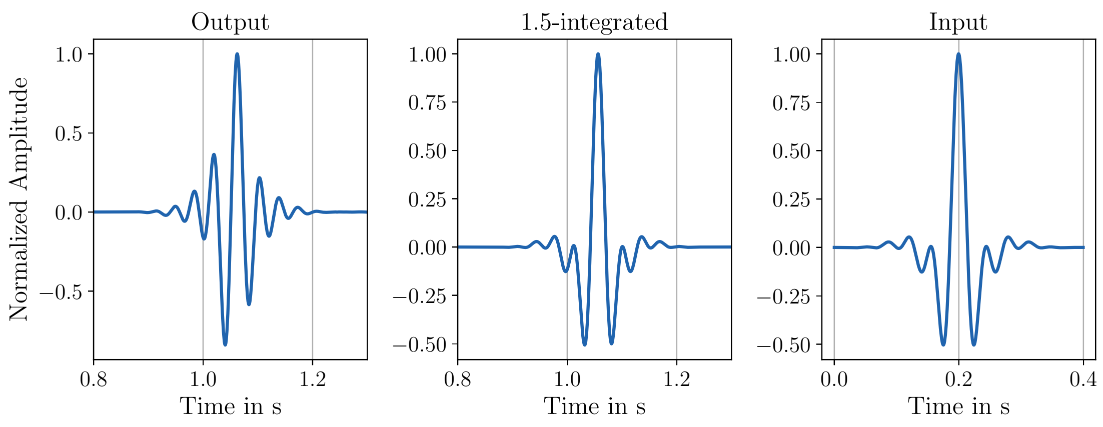
Δh in m	2	4	6	8	10
Computation time in s	903	135	52	20	11

The need for a 1.5-integration has two different reasons. First, a half-integration is needed to transform the data from the modelled 2D point source data to realistic 3D line source data (Forbriger et al., 2014). The remaining integration comes from using a pressure source, which is added to the pressure field, see equation 2.52, of the first-order wave equation. This results in the partial derivative in time of the source signal in the second-order wave equation. Therefore, the signal has to be integrated, to receive the original waveform. (Thorbecke, 2021)

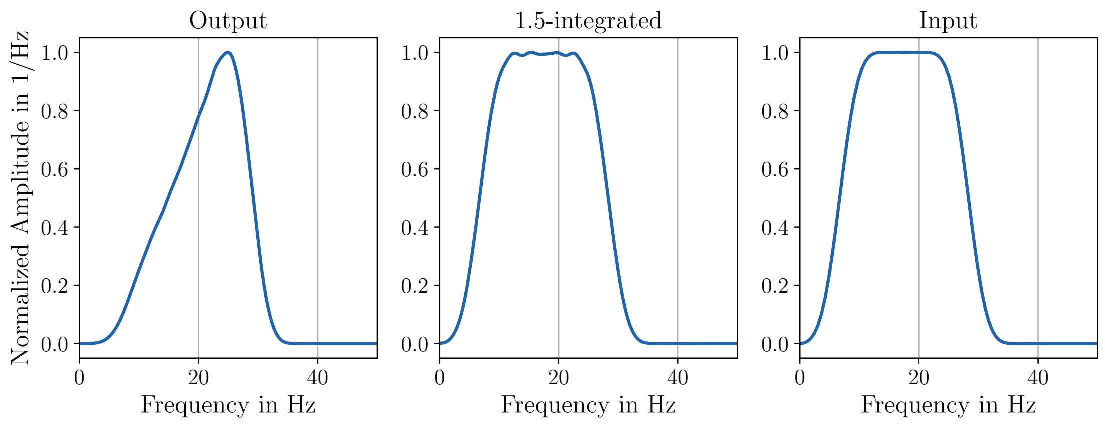
The acquisition geometry consists of 741 receivers with a spacing of 10 m, starting at 155 m and ending at 7555 m, and 37 sources with a spacing of 200 m, starting at 200 m and ending at 7400 m. Both, receivers and sources, are located at 12 m depth. With 30 models and 37 source locations, this results in 1110 shot gathers for training and testing the neural network. All four borders have CPML-boundaries of 15 grid points to avoid artificial boundary reflections. The assumed velocity in the boundary layer is 3500 m/s and a central frequency of 11 Hz.

Furthermore, a high-pass filter is designed so that it is not necessary to model seismograms with source wavelets of different bandwidths. For that, a second flat-spectrum wavelet is created from a linear sweep of 8 to 30 Hz and a length of 8.8 s. The same 1 s tapers as for the source wavelet are applied at the beginning and the end of the sweep. Furthermore, also the higher-frequent flat-spectrum wavelet is tapered with a Blackman window. In Figure 3.7 the spectra of the 5 to 30 Hz flat-spectrum wavelet, the 8 to 30 Hz flat-spectrum wavelet, and the filter designed to obtain the high-frequency wavelet are displayed. To show the effectiveness of this filter, a trace is modeled with both flat-spectrum wavelets as sources. Then the designed filter is applied to the seismogram with the larger bandwidth. The amplitude spectra of both seismograms are shown in Figure 3.8b. It can be seen that both spectra and with that also the resulting waveforms, see Figure 3.8a, are identical. This means, modeling the full bandwidth and filtering afterwards will lead to the same result as modelling with a filtered source wavelet. Therefore, modeling the data only once is sufficient.

With the tests done, the seismograms for all subsurface models are generated with a recording time of 3.5 s. For training the network it is important that the shot gathers differ from each other, to avoid redundant data. Therefore, some gathers are compared for different models but the same shot location, see Figure 3.9. It can be seen that the different number of interfaces already changes the appearance of the data, by introducing more reflections. However, the different salt structures also enable a variable data set. The first layer for all models is water with a constant velocity of $v_p = 1500$ m/s; therefore, the direct wave is similar for all models.



(a) Waveform



(b) Spectrum

Figure 3.6.: Raw output of SOFI2D, its 1.5-integrated signal for a homogeneous model with a flat-spectrum wavelet as input signal. The waveforms and spectra are normalized for comparison.

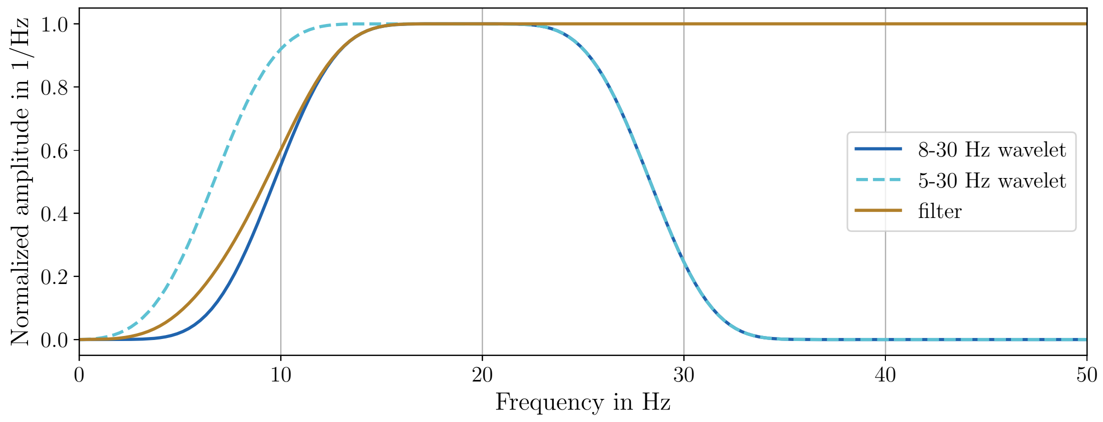
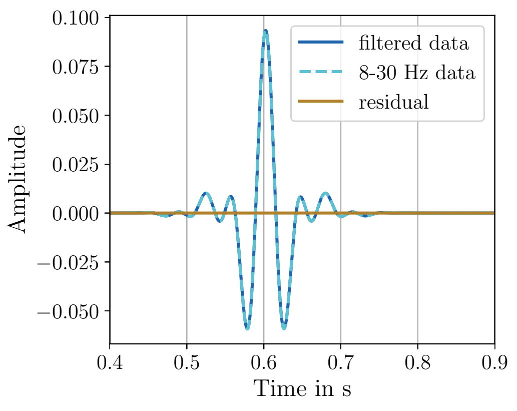
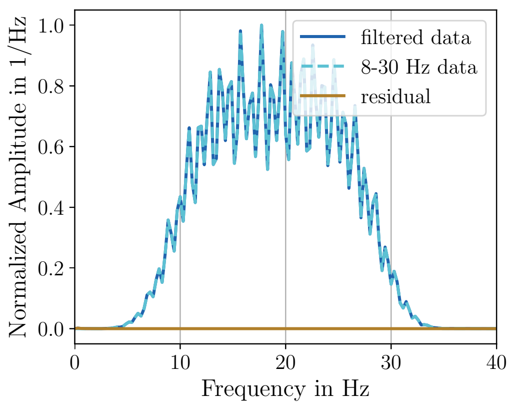


Figure 3.7.: Spectra of two flat-spectrum wavelets and the designed filter (brown) to transform the 5-30 Hz flat-spectrum wavelet into a 8-30 Hz flat-spectrum wavelet.

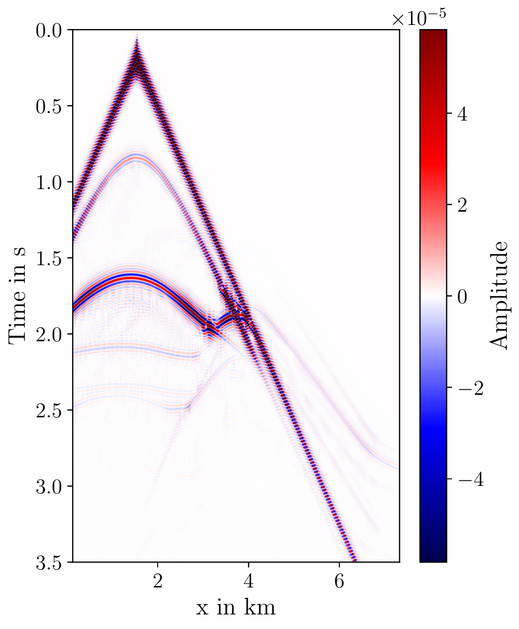


(a) Waveform

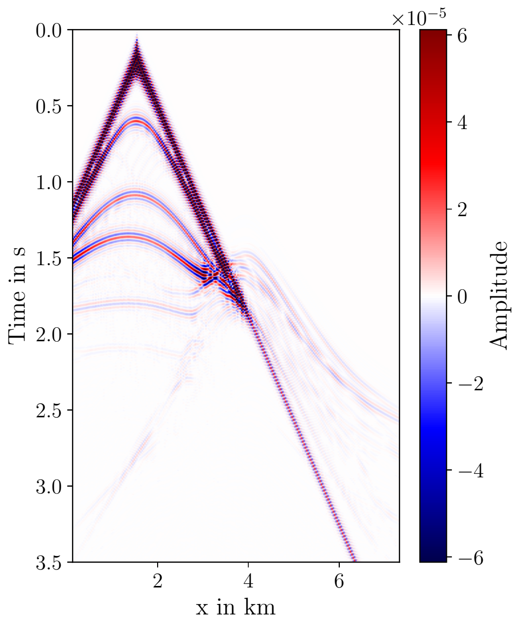


(b) Amplitude spectrum

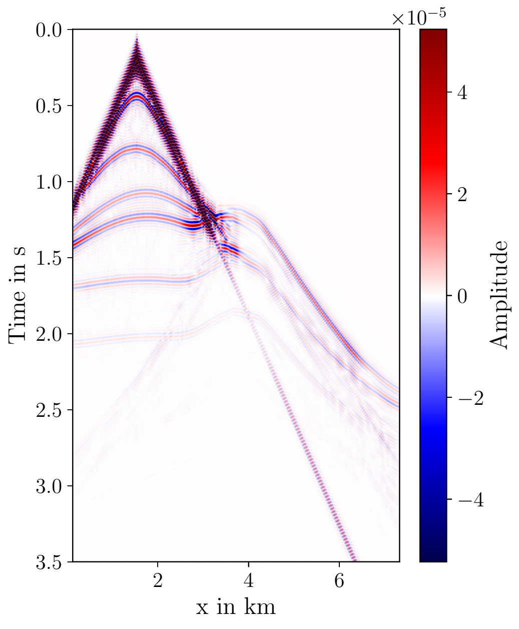
Figure 3.8.: Snippet of the waveform (a) and amplitude spectrum (b) of a trace modeled once with a 8 to 30 Hz flat-spectrum wavelet (dashed line) and once with a 5 to 30 Hz flat-spectrum wavelet, where the trace is high-pass filtered above 8 Hz afterwards (solid). The residual (brown) verifies the good match between the results of both methods.



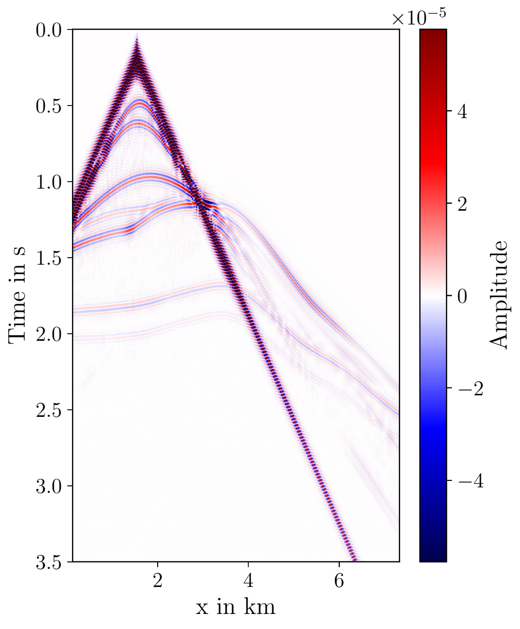
(a) Model 1



(b) Model 2



(c) Model 3



(d) Model 4

Figure 3.9.: Unprocessed shot gathers of the same shot location but with different subsurface models. Each shot gather is displayed with a 99-percentile clip. The synthetic data show large enough differences to be suitable for training the neural network.

3.3. Development of a preprocessing workflow

In this chapter, the development of the processing workflow and the network architecture using a reconstruction test is described. In such a test the network has to learn to reconstruct the input and therefore, the input and target data are the same.

The following steps are used as an initial processing routine to run the first reconstruction test. Before the data are read into the network they need to be preprocessed. The data are split into training, validation and test data sets with a 80 %-10 %-10 % split. Thus, the data from the first 24 models are used for training, the next three data sets for validation, and the last three data sets for testing. The basic data preparation consists of the 1.5-integration of the data, as discussed in section 3.2. For this, the data are scaled by the factor 1×10^8 to increase the value range. A linear taper is applied on the last 0.5 s to avoid artifacts from the integration. Then the actual 1.5-integration is performed. A high-pass filter is applied to dampen low-frequency artifacts from the integration below 3 Hz. Next, noise from a uniform distribution is added to the data to avoid that large parts of the data contain only zeros. The noise is factor 1×10^7 smaller than the maximum amplitude of the data. Finally, every shot gather is standardized and normalized with

$$\tilde{x} = \frac{\frac{x-\mu}{\sigma(x)}}{\max\left(\frac{x-\mu}{\sigma(x)}\right)} , \tag{3.3}$$

where μ is the mean and σ the standard deviation of the data x. The resulting data have a zero mean, a standard deviation of one and a value range between -1 and 1. This has the advantage that the data come from the same distribution, which is beneficial for neural networks (Ioffe and Szegedy, 2015), as discussed in section **??** and section **??**. The mean, standard deviation and the maximum value are saved to be able to transform the output of the network back to its original amplitude range.

A U-Net similar to the one discussed in section 2.1.5.1 is used as neural network. This approach to low-frequency extrapolation was also taken by Aharchaou and Baumstein (2020), Jin et al. (2021), Ovcharenko et al. (2020), Nakayama and Blacquière (2021), and Wang et al. (2020). Compared to the original U-Net by Ronneberger et al. (2015), the U-Net used in this thesis has some slight modifications, see Figure B.1. The max-pooling layer and the subsequent convolutional layer are replaced by a 3×3 convolution with a stride of two. This subsamples the data more efficiently than the traditional combination (Springenberg et al., 2015). Furthermore, deconvolutions are prone to checkerboard artifacts (Odena et al., 2016). To avoid these artifacts, the transpose convolutions in the expanding part of the network are replaced with simple upsampling layers, where the new values are repetitions of the existing values (Wang et al., 2020; Jin et al., 2021; Aharchaou and Baumstein, 2020). In addition, batch norm layers are introduced before the activation functions of every convolution to further improve the training.

To test this architecture with the data, reconstruction tests are conducted by training on the full bandwidth with the input data as target data. Initially, the network consists of two convolutions per layer and three layers in both the encoder and the decoder. ReLU activations are implemented in the hidden layers, and tanh activations are chosen for the last layer. The parameter updates are calculated for batch sizes of 64 with an Adam optimizer. A mean squared error is used as the loss function and a normalized root-mean-square (NRMS) value as introduced by Kragh and Christie (2002) for the evaluation to be able to compare the results with the work of Ovcharenko et al. (2020), who used the NRMS for the evaluation. The NRMS value is defined as

$$\text{NRMS} = \frac{200 \times \text{RMS}\left(a_t - b_t\right)}{\text{RMS}\left(a_t\right) + \text{RMS}\left(b_t\right)} , \tag{3.4}$$

where a_t and b_t are two traces that are compared in the time window t (Kragh and Christie, 2002). The RMS operator is given by:

$$\mathrm{RMS}\,(x_t) = \sqrt{\frac{\sum_{t_1}^{t_2} x_t^2}{N}}\;, \tag{3.5}$$

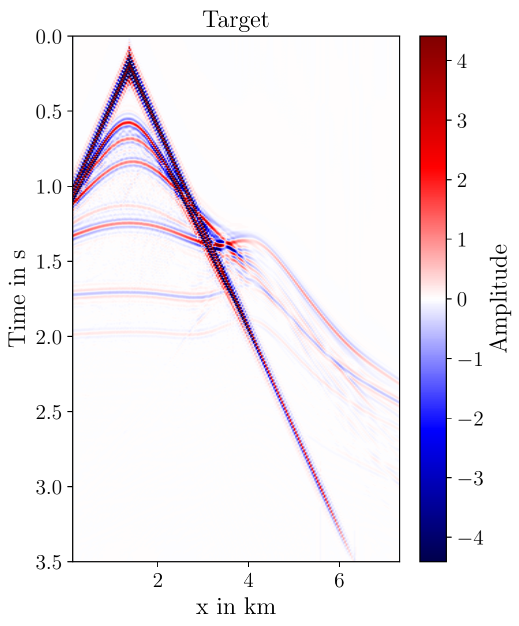
with N being the number of samples in the time window (Kragh and Christie, 2002). The NRMS value is within the interval $[0\,\%, 200\,\%]$ and is sensitive to small changes in amplitudes and small time shifts. Generally, a value between 10 to 30 % is considered an acceptable fit (Ovcharenko et al., 2020).

3.3.1. Split in patches

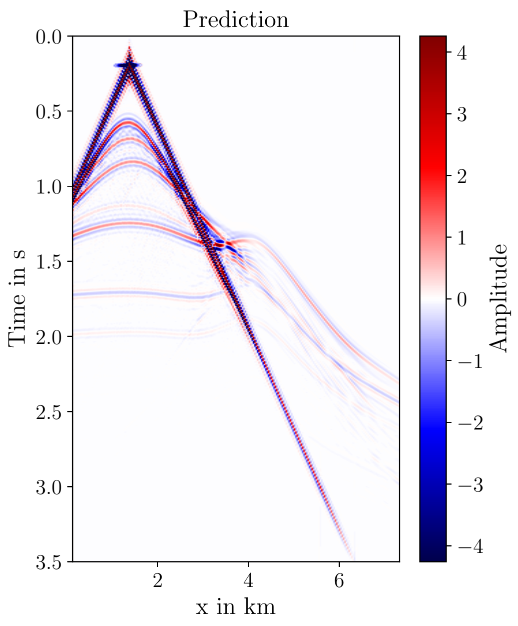
With the 24 training models mentioned earlier and 37 shots per model, 888 shot gathers can be used for training. The network is trained for 100 epochs on the full shot gathers. Due to memory issues, the batch size had to be decreased to three for the full shot gather and the number of graphics processing units (GPU) increased from one to three. Figure 3.10 shows an exemplary output for a shot gather of the test data set. The prediction is nearly identical to the target and only around the source artifacts are visible. The residual plot shows that the recovered amplitudes are not entirely correct. The good data fit corresponds to the low NRMS of 26.9 %. However, the required memory is so high that smaller solutions are needed.

More training data allow for more updates in the training process; therefore, either additional data need to be generated or more data need to be extracted from the existing data. Larger input data require more memory; hence, splitting the data into smaller quadratic patches helps with both: It is not only less expensive in memory, but also provides more data samples. Fang et al. (2020), Aharchaou and Baumstein (2020), and Ovcharenko et al. (2020) split their data into quadratic patches, as well. While Ovcharenko et al. (2020) randomly extract patches, Fang et al. (2020) and Aharchaou and Baumstein (2020) use a sliding window technique, which allows for easy reconstruction. The latter was used in this work, see Figure 3.11. The patch size should be a power of two, because with every layer a 3×3 convolution reduces the input size by a factor of two in both dimensions. A patch size of 128×128 data points is chosen, as this allows several patches along the receiver axis and covers enough data per patch along the time axis.

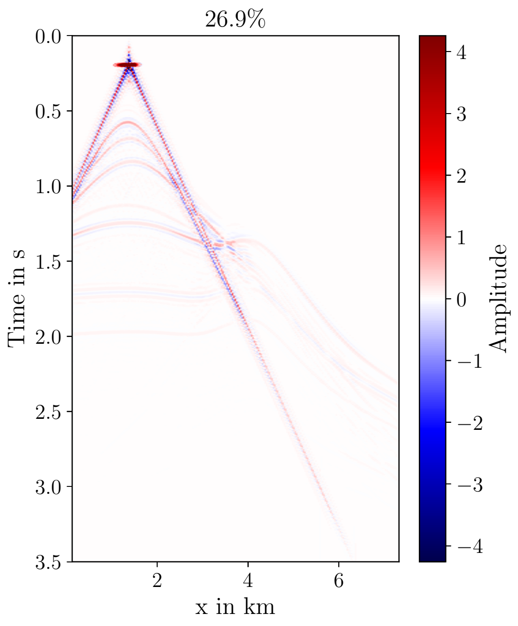
A patch size of 128×128 data points results in 414 patches per shot gather. Therefore, instead of training on $37 \cdot 24 = 888$ data samples, the network trains on $37 \cdot 24 \cdot 414 = 367\,632$ input samples. In addition to reducing memory and increasing the number of data samples, the variability of individual patches is larger than that of the full shot gathers, as can be seen in Figure 3.11. This helps with generalization, as each patch can be seen as containing a basic element of a shot gather (Fang et al., 2020). TensorFlow's `extract_patches` function considers only patches that fit completely inside the data (Abadi et al., 2015); therefore, zeros are padded at larger offsets and larger time steps, such that a multiple of the patch size fits inside. With that the entire data set can be covered with patches. The padded zeros are cropped afterwards to obtain the original size.



(a) Target

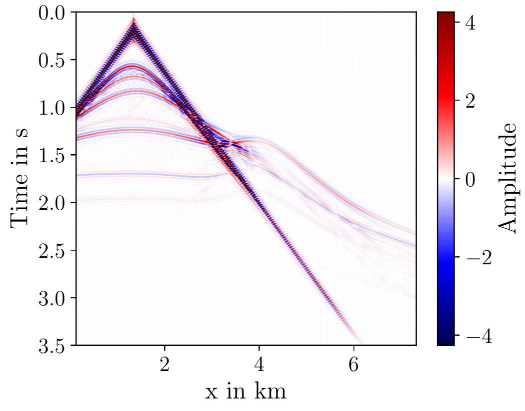


(b) Prediction

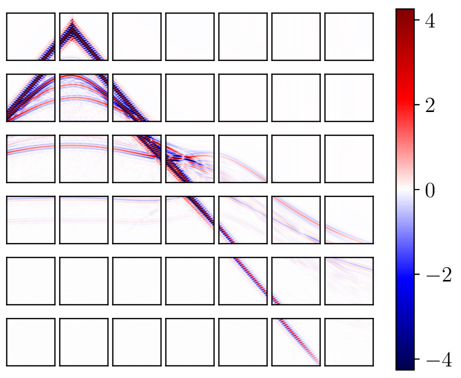


(c) Residual

Figure 3.10.: Shot gathers of (a) target, (b) prediction and (c) residual of a network trained on full shot gathers. (a-c) have the same clip, which is the 99-percentile of (a). The title of the residual plot gives the NRMS value, which indicates a good fit.



(a) Original Input Data



(b) Patches

Figure 3.11.: The data are split into patches to increase the number of training data and decrease the required memory. In this example, the data are split into patches of 100×1400 data points for visibility, in the used neural network much smaller patches of 128×128 data points are used.

3.3.2. Normalizing the data

With the decision to split the data into patches, the question arises, where to place this step in the processing routine. Therefore, it is tested whether normalizing before or after splitting the data in patches is more beneficial. When using patches, a batch size of 64 and one instead of three GPUs is enough to fit the data into memory. In theory, normalizing patch-wise means that every patch comes from the same distribution. While normalizing per-shot means that the ratio of the amplitudes is kept. If we imagine a patch that contains only noise, we can see a difference. In the case of normalizing per patch, the amplitudes of the noise will be boosted which encourages the network to try to focus on reconstructing noise. When normalizing before the splitting, the noise will still keep its low-amplitude values and might therefore not be as important for the network as the signal. Furthermore, normalizing before the splitting reduces the number of parameters that need to be stored significantly, as each scalar needs to be stored in order to reverse the normalization after prediction.

Shot normalization

Figure 3.12 depicts the results of normalization before subsampling into smaller patches. Target and prediction visually seem to have a great fit, though there is a bias on the prediction and the values around the direct wave are too high. The NRMS value of 13 % confirms the good fit. In the residual plot, small grid-like artifacts can be seen. They appear at the boundaries of the single patches. These artifacts are further discussed in section 3.3.3. Fang et al. (2020) use a shot-wise normalization, but do not see the low-frequency artifacts shown here.

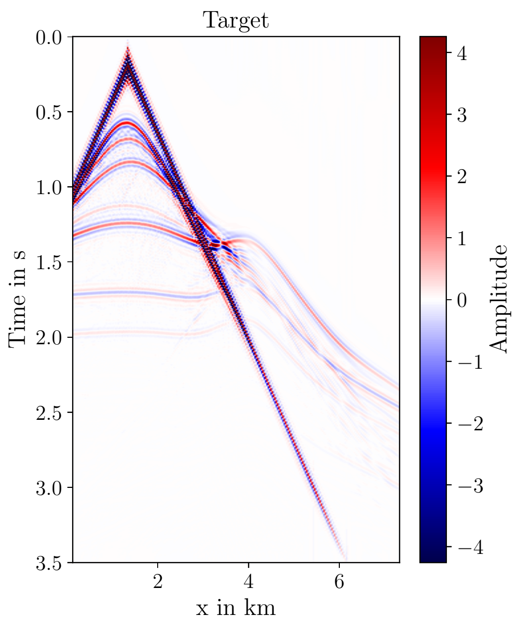
Patch normalization

Artifacts occur as well when the data are normalized after subsampling into smaller patches, see Figure 3.13. The recovered patches have different biases, which results in a blocky structure in the output. However, the amplitudes of the biases are small compared to the signals. Therefore, the prediction is still similar to the target and the NRMS value of 1.9 % is very small which indicates a nearly perfect fit. The artifacts are strongest at patches which consist of both very high amplitudes of the direct wave and very small amplitudes of the background noise. Ovcharenko et al. (2020) use a patch-wise normalization, but they do not report similar artifacts.

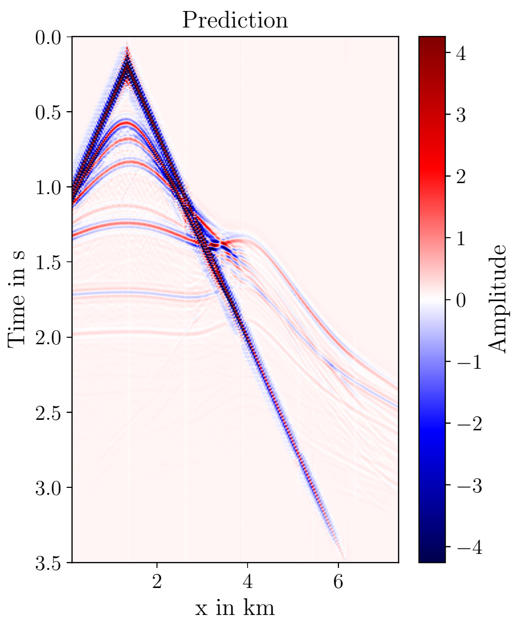
Trace normalization

In FWI data are usually normalized per trace rather than per whole shot gather (Louboutin et al., 2017). The extrapolated data are used for FWI, therefore, it might be beneficial to use a trace normalization for the network as well. Figure 3.14 shows the differences between normalizing shot-wise and trace-wise. The value range for both data sets is the same. Trace-by-trace normalization has the advantage that the natural amplitude decay at larger offsets is reversed, such that all traces contribute to the gradient in the same amount (Louboutin et al., 2017).

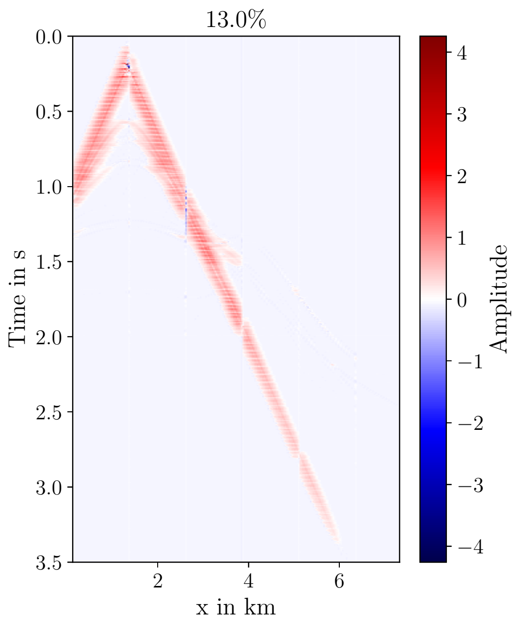
Figure 3.15 shows the predictions of a network trained on trace-normalized data. Similarly to the previous data, the NRMS value of 2.9 % is very good. Also, the prediction and the target look similar, except for the traces close to the source location, where a wrong bias can be seen as vertical line. The residual plot indicates that the single traces have different biases, which results in vertical stripes.



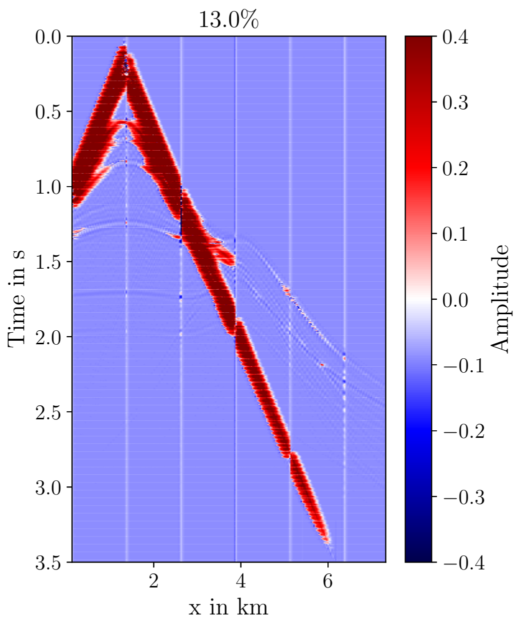
(a) Target



(b) Prediction

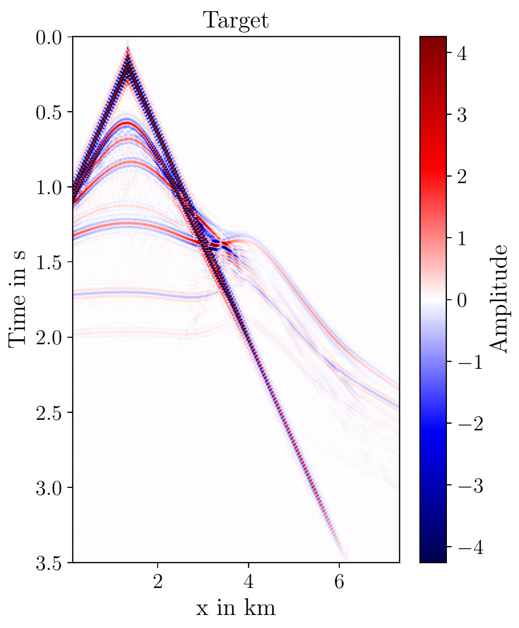


(c) Residual

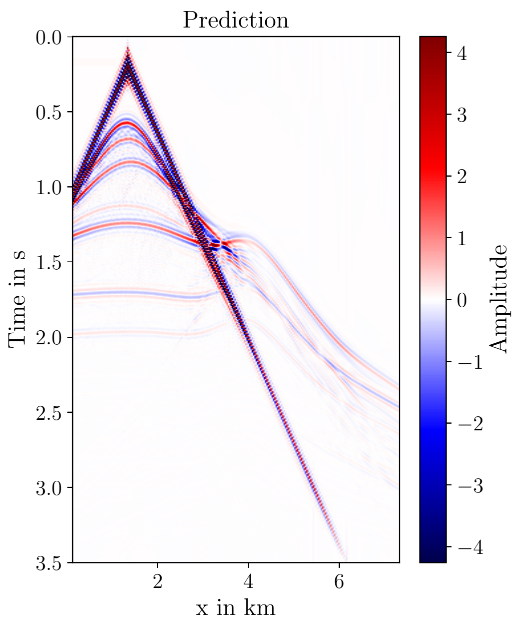


(d) Residual

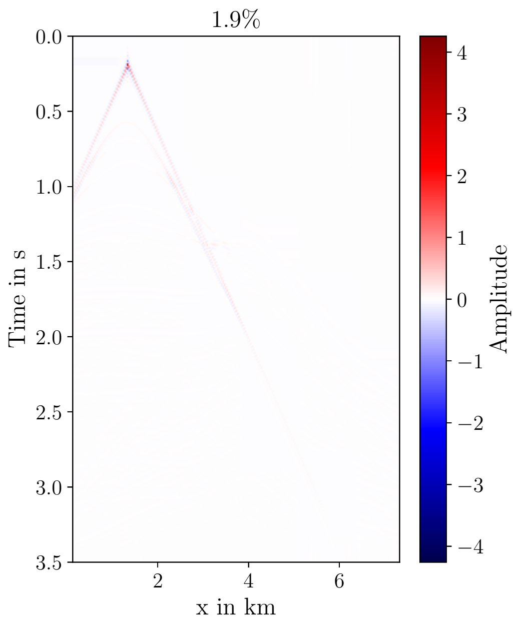
Figure 3.12.: Shot gathers of (a) target, (b) prediction and (c) residual of a network trained on per-shot normalized patches. (a)-(c) have the same clip, which is the 99-percentile of (a). (d) is the same as (c) but with a different clip to enhance the grid-like artifacts. The title of the residual plots gives the NRMS value, which indicates a good fit.



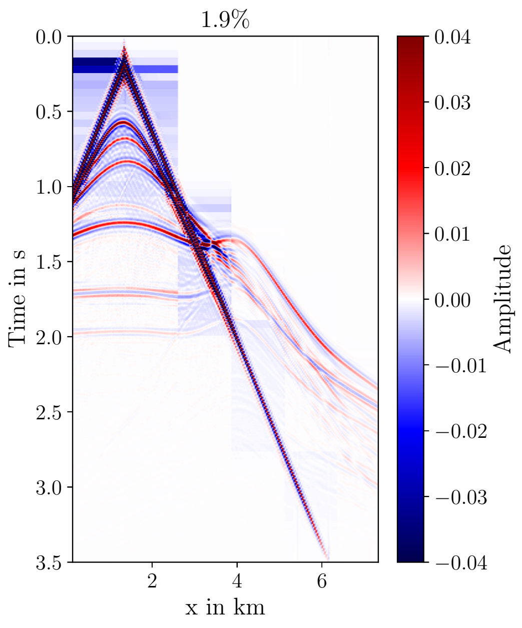
(a) Target



(b) Prediction



(c) Residual



(d) Residual

Figure 3.13.: Shot gathers of (a) target, (b) prediction and (c) residual of a network trained on per-patch normalized patches. (a)-(c) have the same clip, which is the 99-percentile of (a). (d) is the same as (c) but with a different clip to enhance the blocky artifacts. The title of the residual plots gives the NRMS value, which indicates a nearly perfect fit.

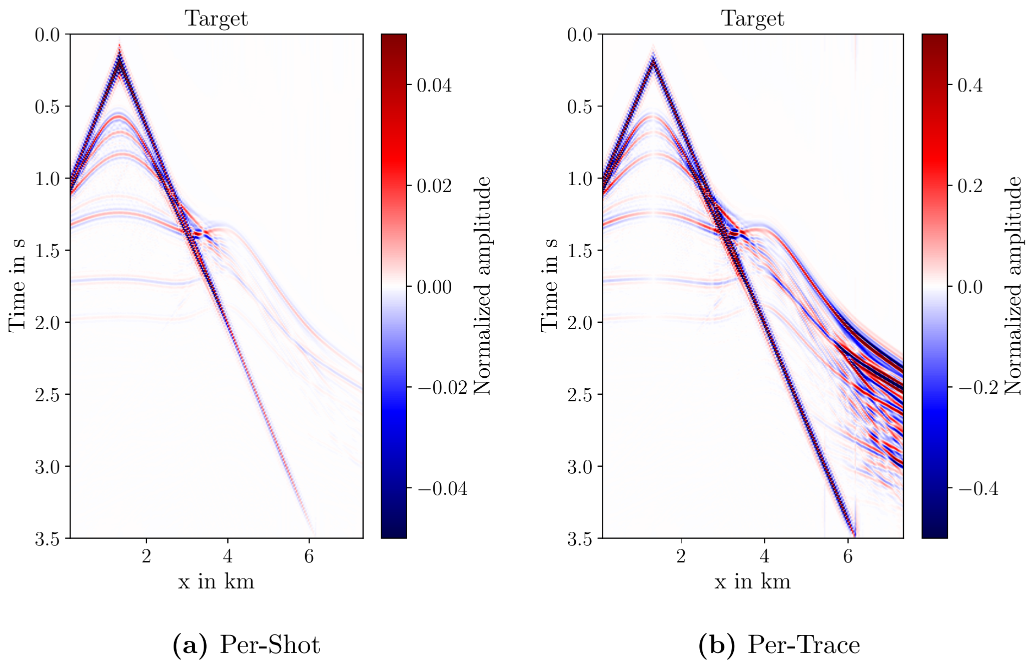


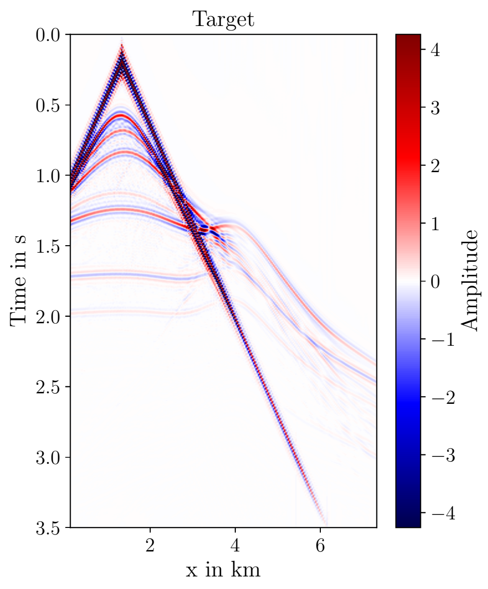
Figure 3.14.: Shot gathers normalized (a) per shot and (b) per trace. Note that the amplitudes in the far offsets are increased in case of the trace normalization.

Global normalization

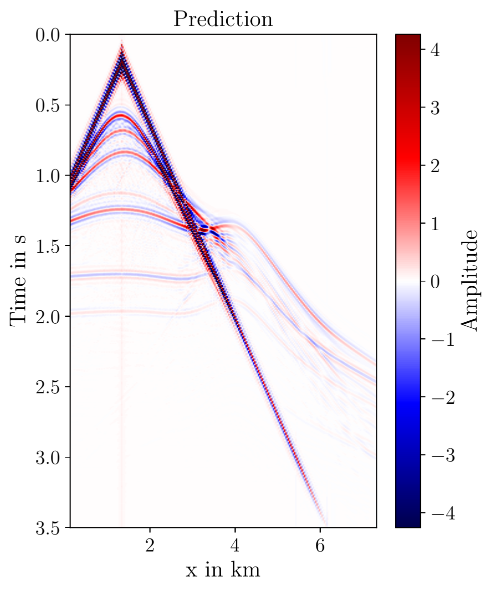
Another possibility is to normalize all data with the mean and standard deviation of the entire training data set. With this approach, all values remain between -1 and 1. The advantage of this approach is that every value of the original value range corresponds to one specific value in the output range. Furthermore, the number of parameters that need to be stored decreases significantly. Figure 3.16 depicts the results of a network trained on globally normalized data. The NRMS value of 33.6 % states a poorer fit than the previous results and the prediction has a strong bias. By comparing Figure 3.16 to Figure 3.12, it can be seen that the residual shows a similar low-frequency artifact around the direct wave. Furthermore, a strong grid-like artifact disturbs the resulting shot gather, which is further discussed in section 3.3.3.

3.3.3. Overlapping patches

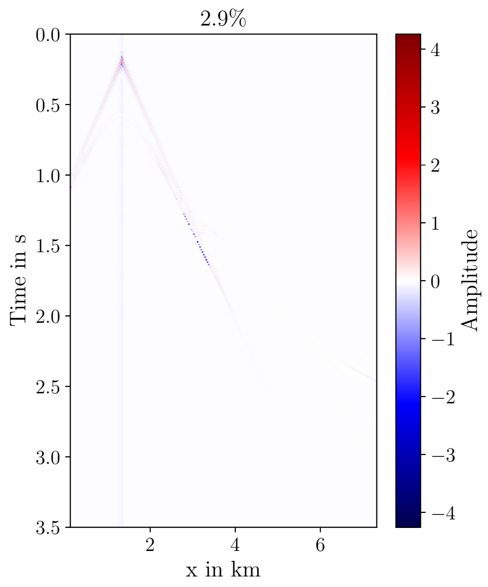
Applying the trained network on adjacent patches leads to an unwanted grid structure in the output, see Figure 3.16. Artifacts occur because convolutions and deconvolutions with zero-padding are used. These are suspected to introduce artifacts at the borders (Alsallakh et al., 2021). Data points on the borders of the input patch are involved in fewer convolutions than data points in the center (Alsallakh et al., 2021). This leads to less accurate results for the data points at the borders (Li et al., 2017). For larger input patches, the relative extent of the border effects decreases (Alsallakh et al., 2021). Therefore, the patch size is increased for the test data set. The input gather is first padded with zeros of half the training patch size and then overlapping patches of size 256×256 are used instead of adjacent patches of size 128×128. In the original U-Net paper (Ronneberger et al., 2015), overlapping tiles were also used. In this work, the 128×128 center part of the output is cropped such that the border parts with the artifacts can be neglected. Ovcharenko et al. (2020) also clip the border data points to address the edge effects of the convolution. The same network as before, still trained on adjacent patches, is used to predict the results on overlapping patches, which leads to the successful avoidance of the grid-like artifacts, see Figure 3.17.



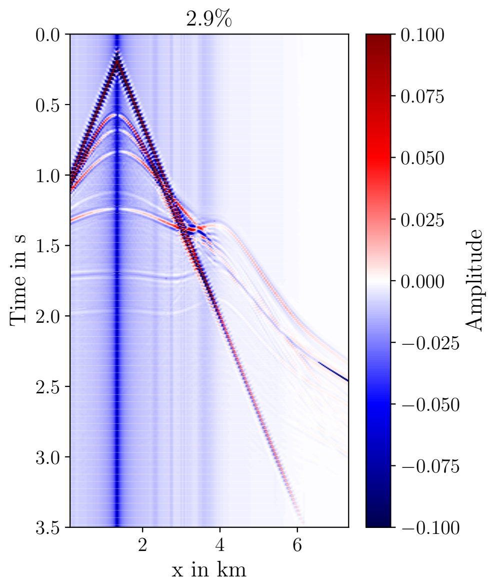
(a) Target



(b) Prediction

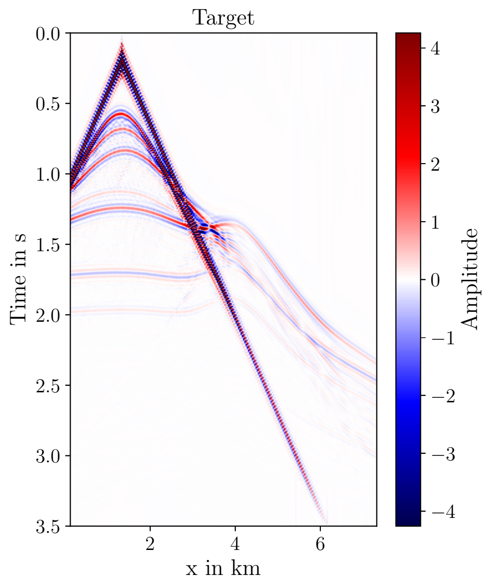


(c) Residual

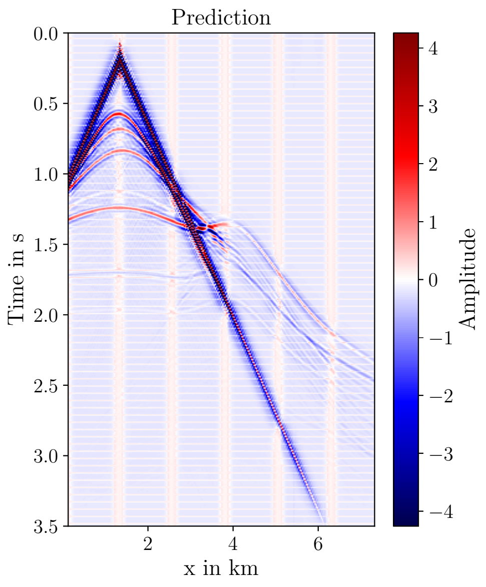


(d) Residual

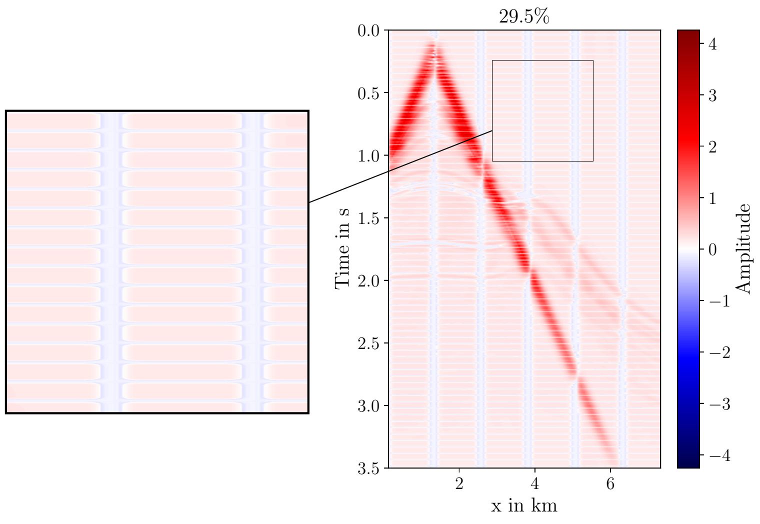
Figure 3.15.: Shot gathers of (a) target, (b) prediction and (c) residual of a network trained on per-trace normalized patches. (a)-(c) have the same clip, which is the 99-percentile of (a). (d) is the same as (c) but with a different clip to enhance the artifacts. The title of the residual plots gives the NRMS value, which indicates a very good fit.



(a) Target

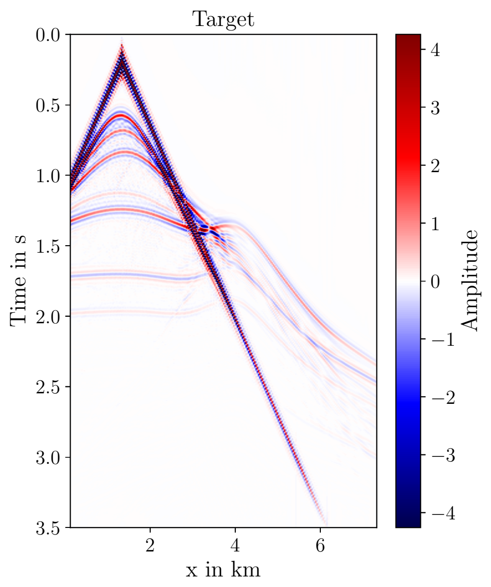


(b) Prediction

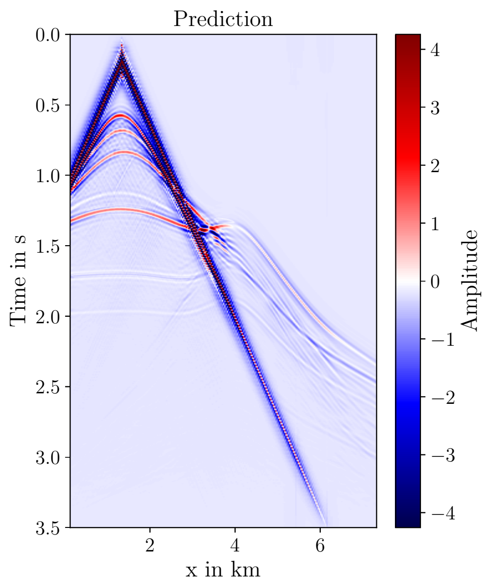


(c) Residual

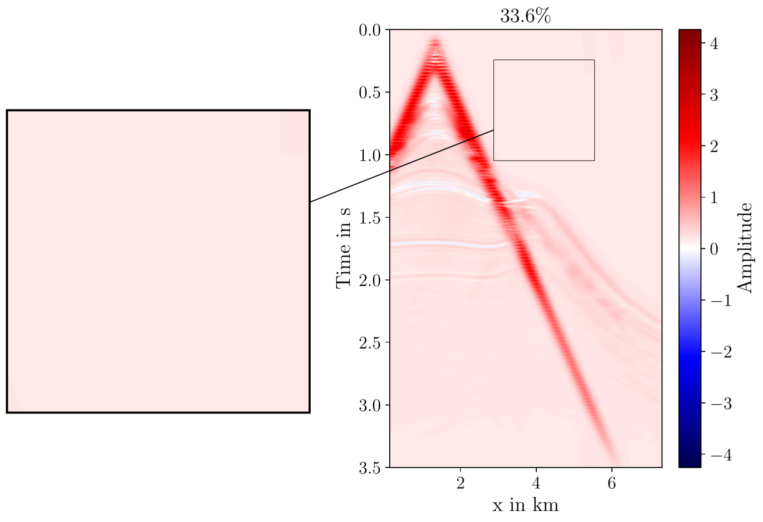
Figure 3.16.: Shot gathers of (a) target, (b) prediction and (c) residual of a network trained on globally normalized patches. The zoomed area enhances the grid-like artifacts. (a)-(c) have the same clip, which is the 99-percentile of (a). The title of the residual plot gives the NRMS value, which indicates a poorer fit than the previous results.



(a) Target



(b) Prediction



(c) Residual

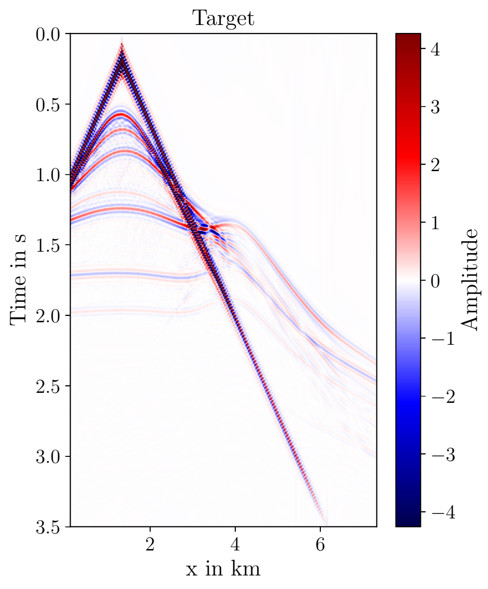
Figure 3.17.: Shot gathers of (a) target, (b) prediction and (c) residual of a network trained on globally normalized patches. For this test overlapping patches are used, which remove the grid-like artifacts, compare zoomed area with Figure 3.16. (a)-(c) have the same clip, which is the 99-percentile of (a). The title of the residual plot gives the NRMS value, which indicates a poorer fit than the previous results.

3.3.4. Comparison of the normalization techniques

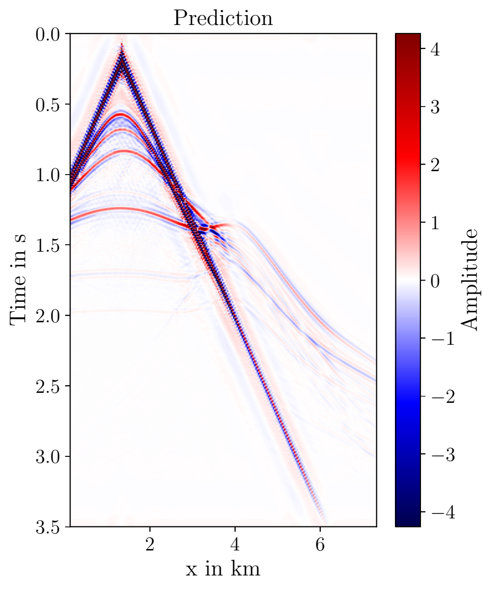
Even though training on full shot gathers gives good results, it is memory intensive. Therefore, this approach is not feasible in practice. All other trials introduced some artifacts, with low errors for the patch-wise and trace-wise normalization. However, the blocky artifacts produced by patch-wise normalization occur as artificial signals on the single traces. This might also affect the amplitude relations between real events occurring on one trace. Trace-wise normalization introduces a different bias for each trace. The artifacts for both methods are strongest on patches or traces with the largest amplitudes. Hence, if the network predicts all patches with a similar constant bias, this error will be boosted differently depending on the parameters needed for reconstructing the original amplitudes. The global normalization and the shot normalization show a similar behavior, as they are normalized in the same manner. Both results suffer from a bias and a low-frequency artifact along the direct wave. Here the complete shot gather is normalized, therefore, the reconstruction of the true amplitudes does not introduce additional artifacts, beside the grid-like structure. This artifact can be easily avoided by using overlapping patches for the test data set. However, the origin of the low-frequency artifact is not obvious. A high-pass filter with a 3 Hz corner frequency is applied to the globally normalized prediction to reduce the artifacts. Figure 3.18 depicts the resulting prediction and the residual. The NRMS value decreases to 7.0 %, and the bias decreases in amplitude as well. Filtering is an easy way to improve the data quality and for the global normalization only three parameters in total need to be stored. Therefore, the global normalization is my chosen method.

3.3.5. Final pre-processing workflow after reconstruction tests

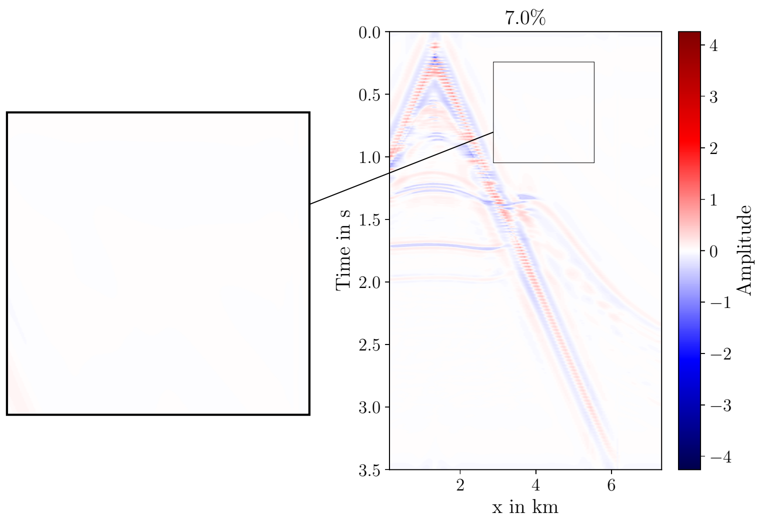
The resulting preprocessing workflow is summarized in Figure 3.19. The data are first scaled by 1×10^8, then a linear taper is applied from 3.0 s to 3.5 s to reduce the artifacts of the subsequent 1.5-integration. The input data are then high-pass filtered to minimize low-frequency artifacts introduced by the integration. Afterwards, noise is added. Subsequently, each trace is normalized with the global mean and standard deviation. A padding of 128 zeros is applied to the far offsets and the end of the traces. Next, the input data are split into patches of size 128×128. For test data an additional padding of 64 rows or columns of zeros along all sides of the shot gather is applied. Lastly, the test data are then split into overlapping patches of size 256×256.



(a) Target



(b) Prediction



(c) Residual

Figure 3.18.: Shot gathers of (a) target, (b) prediction and (c) residual of a network trained on globally normalized patches and high-pass filtered above 3 Hz. (a)-(c) have the same clip, which is the 99-percentile of (a). The title of the residual plot gives the NRMS value, which indicates a good fit.

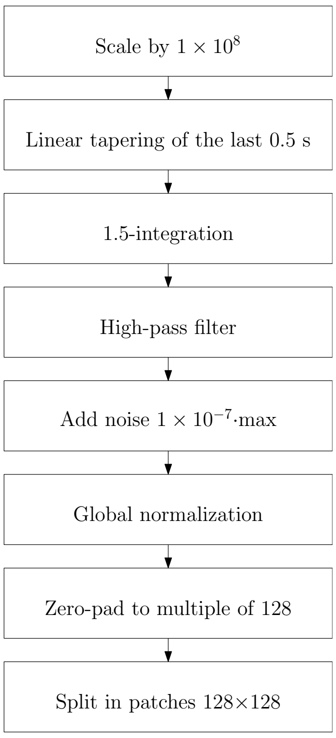


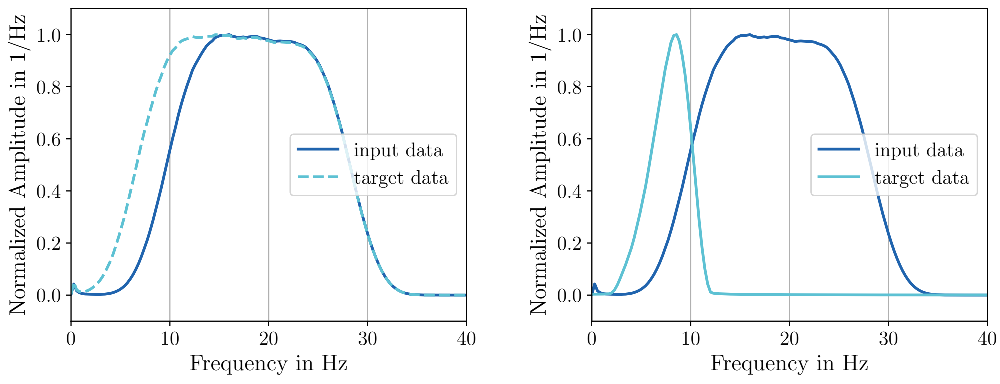
Figure 3.19.: Preliminary pre-processing workflow for training data after the reconstruction tests. For the test data a patch size of 256×256 is used.

4. Application of deep learning for low-frequency extrapolation

With a successful reconstruction test, the selected processing workflow can be applied to the actual problem. The task of the network is to predict low-frequency data from high-frequency data; therefore, different input and target data sets need to be created. Two different prediction possibilities are considered, see Figure 4.1. In the first approach, the whole bandwidth is extrapolated, which allows a good quality control on whether also the high frequencies are reconstructed correctly. In the second approach, only the low frequencies are predicted. This approach has the advantage that the network actually has to learn something completely new, and might not just focus on getting the high frequencies correct. In the related literature the latter approach seems to be more common, only Wang et al. (2020) and Nakayama and Blacquière (2021) use the entire bandwidth as target data. For creating the input data, the high-pass filter designed in section 3.2 is used. The low-frequency target data are set between 3 and 12 Hz. Hence, they leak into frequencies larger than 10 Hz to avoid ripples in the seismograms (Figure 4.1b). The low-frequency data are obtained using a bandpass filter that covers frequencies from 3 to 12 Hz. Lower frequencies are not considered as they are influenced by edge artifacts of the 1.5-integration.

4.1. Full-bandwidth extrapolation

In the first approach, the entire bandwidth is predicted. The same hyperparameters are chosen for the network as in the reconstruction tests, see section 3. Figure 4.2 shows the results after 100 epochs of training on the globally normalized data. The target and predicted data show high similarity. However, the same artifacts as in the reconstruction tests with global normalization are present. Around the high-amplitude values, low-frequency artifacts can be seen. The NRMS value is 17.0 %, which still indicates a good fit. To further investigate the results, the resulting spectrum and some predicted waveforms are studied (Figure 4.3). For high frequencies, the spectrum of the prediction fits very well with the target spectrum. For frequencies lower than approximately 13 Hz, the spectrum of the prediction follows the target spectrum only partly. Below 3 Hz, the low-frequencies amplitudes increase instead of going towards zero. For offsets up to around 2 km, the phases of the waveforms have a nearly perfect fit. The amplitudes are not entirely reconstructed correctly. For larger offsets, the extrapolated waveform does not follow the target wavelet but follows the input wavelet instead. Furthermore, a bias can be observed for long offsets. To remove the large DC component, a high-pass filter is applied to the frequencies between 0 to 3 Hz (Figure 4.4a). As a result, the residual of the filtered prediction and the target as well as the NRMS value are decreased (Figure 4.5). The NRMS value of 6 % can be considered a nearly perfect fit, and small offset traces up to approximately 2 km show a very good reconstruction of phases and amplitudes (Figure 4.3c). For larger offsets, the prediction still tends to copy the input rather than predicting the low frequencies (Figure 4.3d).



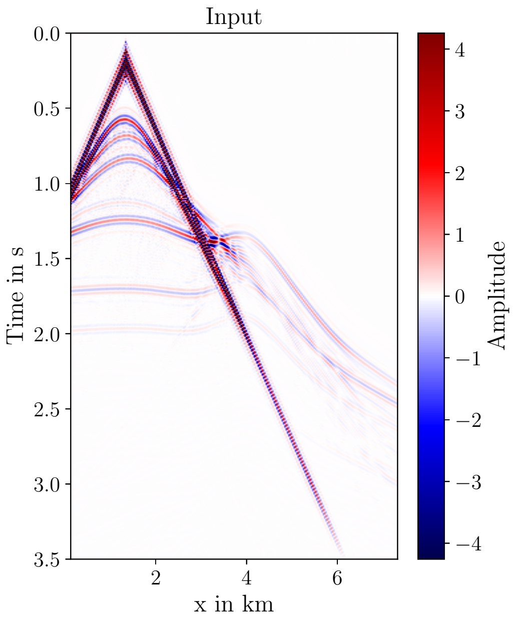
(a) Full-bandwidth prediction **(b)** Low-frequency prediction

Figure 4.1.: Amplitude spectra of a shot gather for input and target data for two different approaches. Note that the amplitudes of all spectra a normalized individually.

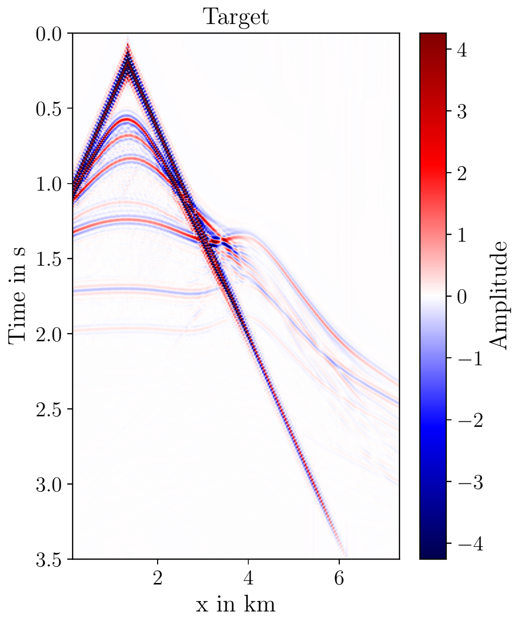
Furthermore, the approach of bandwidth extension allows to determine how well the input data is reconstructed. For that, the high-pass filter designed in section 3.2 is applied to the predicted data and the resulting spectrum is compared to the spectrum of the input data (Figure 4.6). Besides small amplitudes at frequencies larger than 35 Hz, both spectra match perfectly. This means that the network is able to transfer the complete information of the input layer to the output layer. To be able to compare the results of the bandwidth extension with the ones of the low-frequency approach, only the low-frequency data are extracted from the predicted data by applying the same filter as used to create the low-frequency target data. The resulting spectrum and waveforms show a good fit for the low frequencies (Figure 4.7). At smaller offsets, only small differences in the amplitudes can be observed, while the phase shows a good fit. For the largest offset displayed, the first arrival still matches reasonably well with the target data; however, at the arrival of the direct wave at around 2.3 s, both amplitude and phase are completely off for the side lobes, while the main peak is still reconstructed correctly. Figure 4.8 depicts the shot gathers for the low frequencies. The NRMS value of 16 % verifies the good fit. However, by comparing the shot gathers of target and prediction, it can be seen that the prediction is not able to reconstruct the low frequencies of the events that occur around 1.7 s and around 2.0 s. For the application of FWI on that data, these reflections are not important, as they will only play a role at later stages of the FWI. The low frequencies are rather used to fit a smooth background model than the single reflectors.

4.2. Low-frequency extrapolation

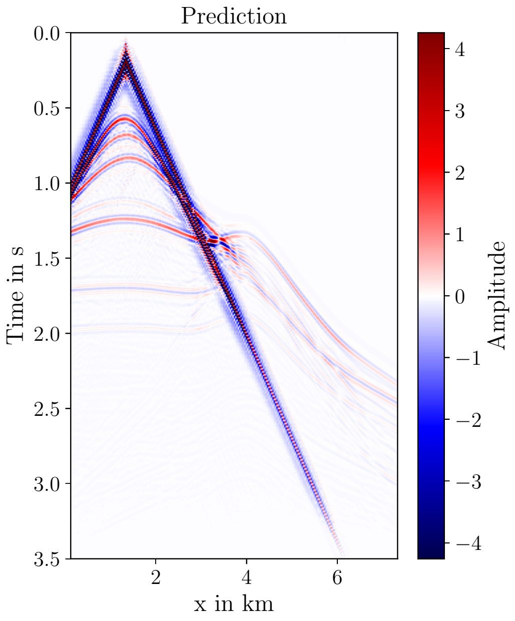
In the second approach only the low frequencies are used as target. The same hyperparameters as before are chosen and the network is trained for 100 epochs. Figure 4.9 depicts the true low frequencies, the extrapolated low frequencies, and the residual of a shot gather. The shot gathers of target and prediction already differ during the visual inspection. The NRMS value of 49.4 % indicates a poor fit, and the residual plot shows a strong bias in the shot gather. Looking at the spectrum in Figure 4.9a confirms the poorer fit. This approach suffers from a high DC component as well, which results in the strong bias. Furthermore, unexpected high-frequency content can be found in the data. The waveform (Figure 4.10b) is also affected by the bias. To properly quality control the data, the DC component is removed by applying a high-pass filter above 3 Hz. The resulting spectrum (Figure 4.12a) shows that with the removed very low frequencies the spectra of target and extrapolated data resemble each other and the bias is removed from the waveforms, see Figure 4.12.



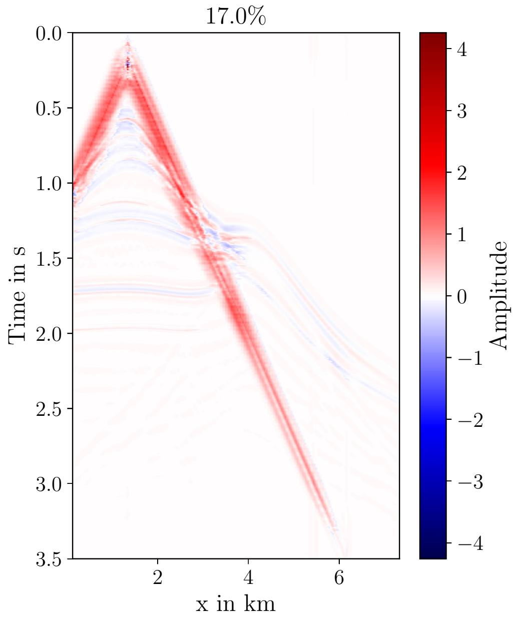
(a) Input



(b) Target

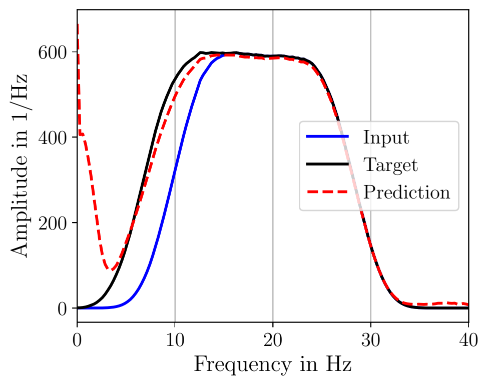


(c) Prediction

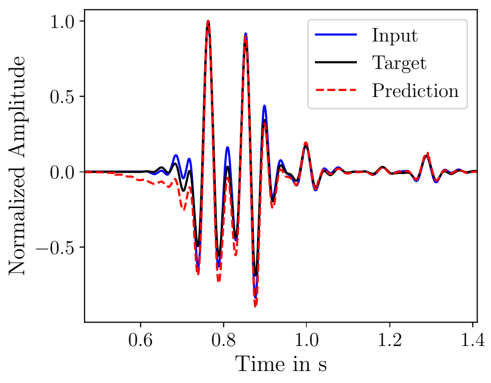


(d) Residual

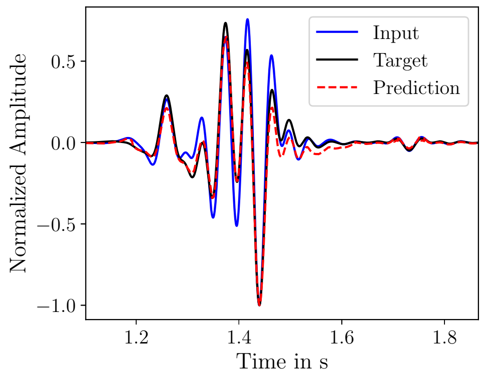
Figure 4.2.: Shot gathers of (a) input, (b) target, (c) prediction and (d) residual of a network trained to extrapolate the full bandwidth. (a)-(d) have the same clip, which is the 99-percentile of (b). The title of the residual plot gives the NRMS value, which indicates a good fit.



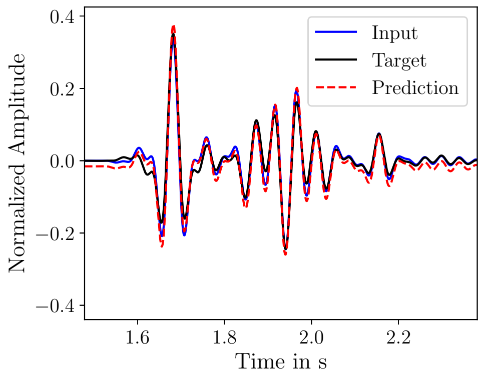
(a) Spectrum



(b) Snippet of waveform

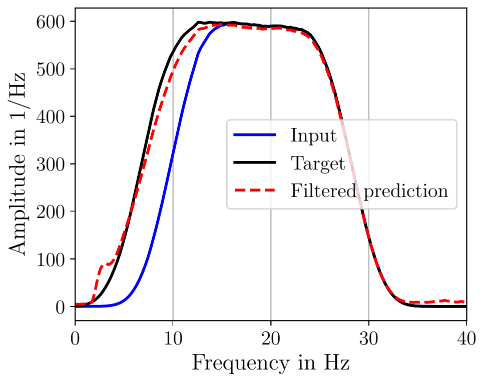


(c) Snippet of waveform

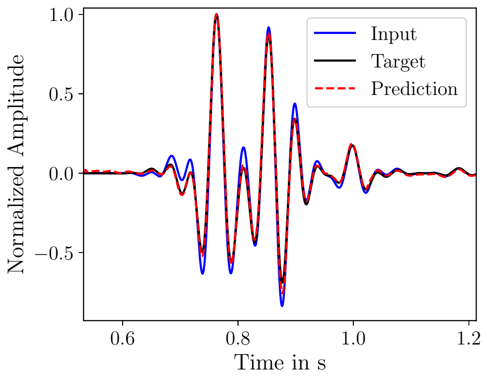


(d) Snippet of waveform

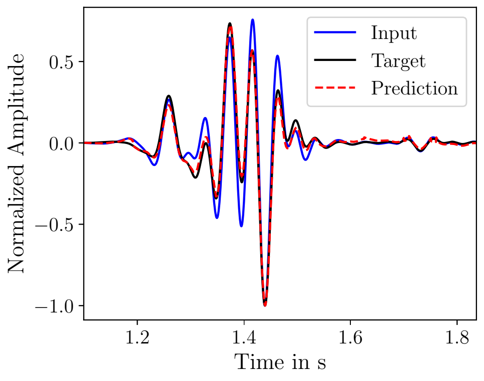
Figure 4.3.: (a) Amplitude spectra of a shot gather for true and extrapolated full-bandwidth data. (b) Snippet of a waveform at 655 m offset. (c) Snippet of a waveform at 1945 m offset. (d) Snippet at 3945 m offset.



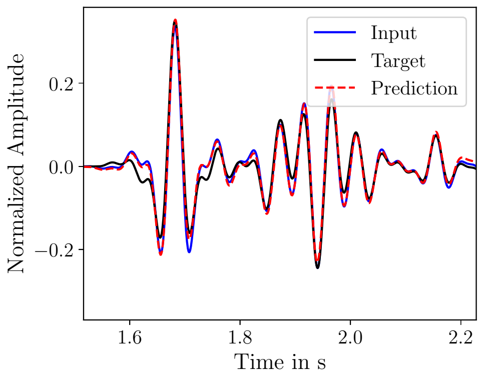
(a) Spectrum



(b) Snippet of waveform

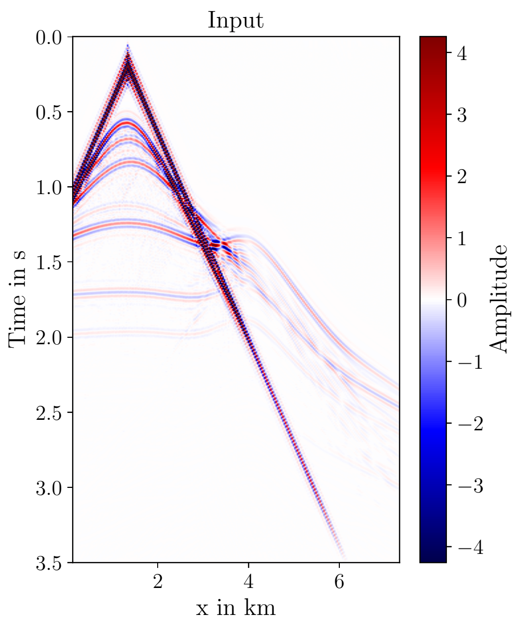


(c) Snippet of waveform

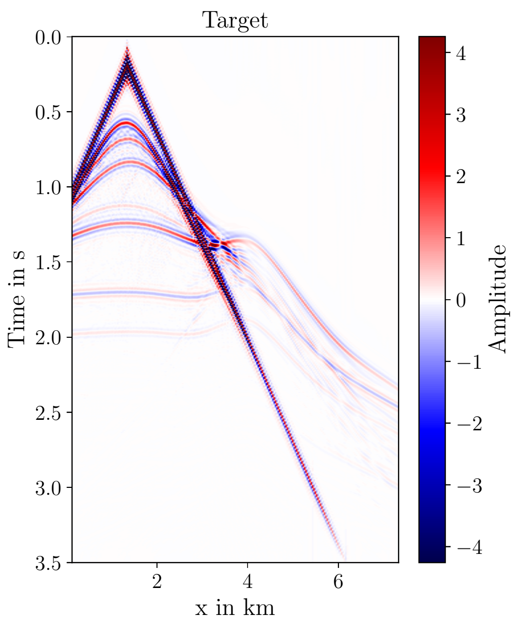


(d) Snippet of waveform

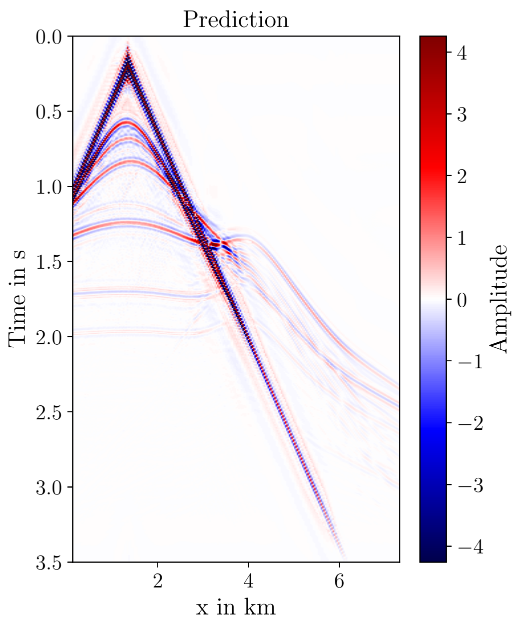
Figure 4.4.: (a) Amplitude spectra of a shot gather for true and extrapolated full-bandwidth data which were high-pass filtered afterwards. (b) Snippet of a waveform at 655 m offset. (c) Snippet of a waveform at 1945 m offset. (d) Snippet at 3945 m offset. Note that compared to Figure 4.3, the DC component and the very low amplitudes are removed.



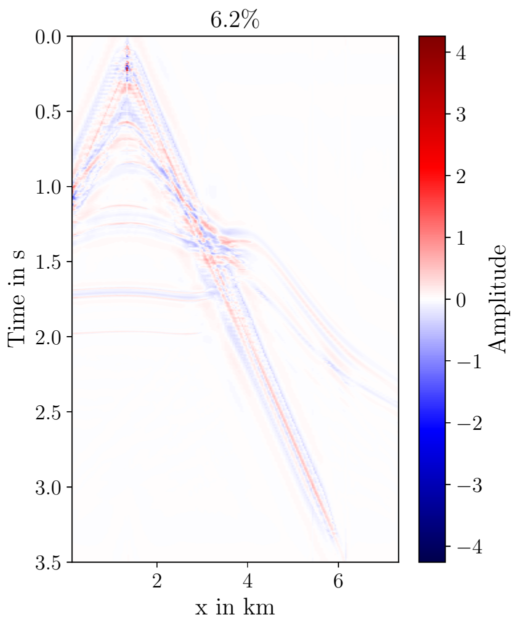
(a) Input



(b) Target



(c) Prediction



(d) Residual

Figure 4.5.: Shot gathers of (a) input, (b) target, (c) prediction and (d) residual of a network trained to extrapolate the full bandwidth. (a)-(d) have the same clip, which is the 99-percentile of (b). The title of the residual plot gives the NRMS value, which indicates a good fit. Compared to the results in Figure 4.2, frequencies below 3 Hz are removed.

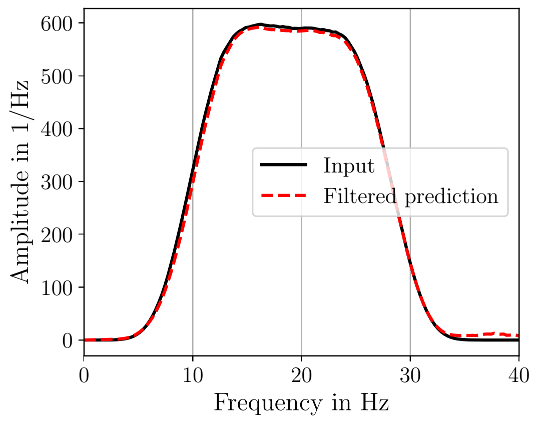


Figure 4.6.: Amplitude spectra of a shot gather with the input data (solid line) and the high-pass filtered predictions (dashed line).

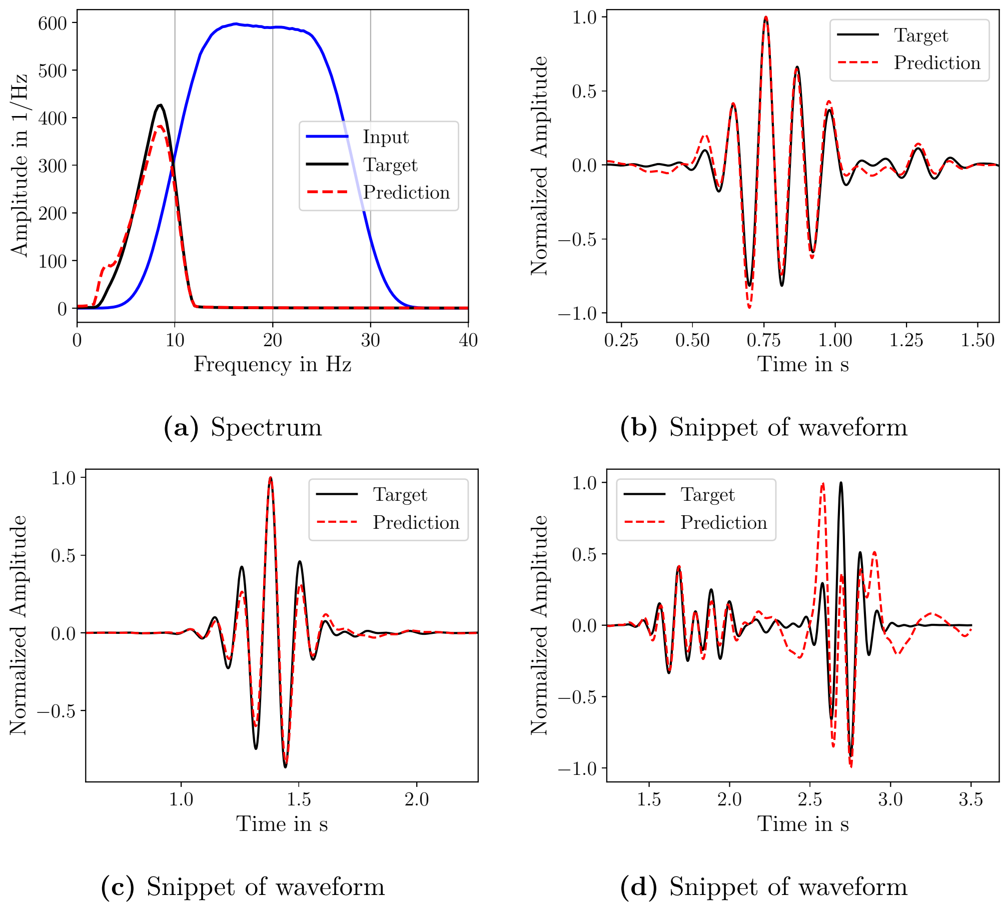
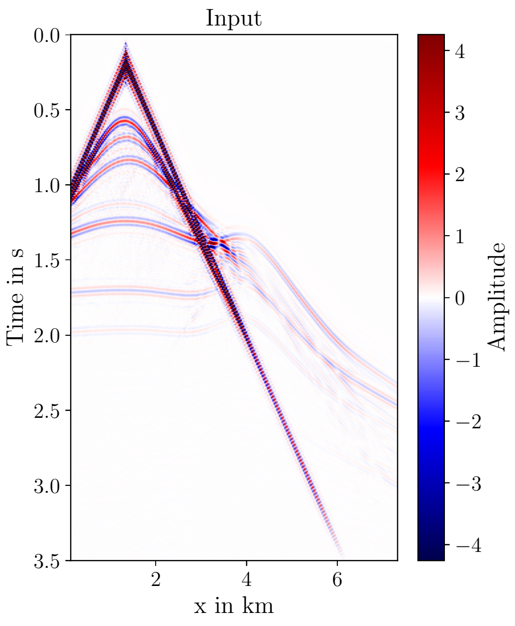
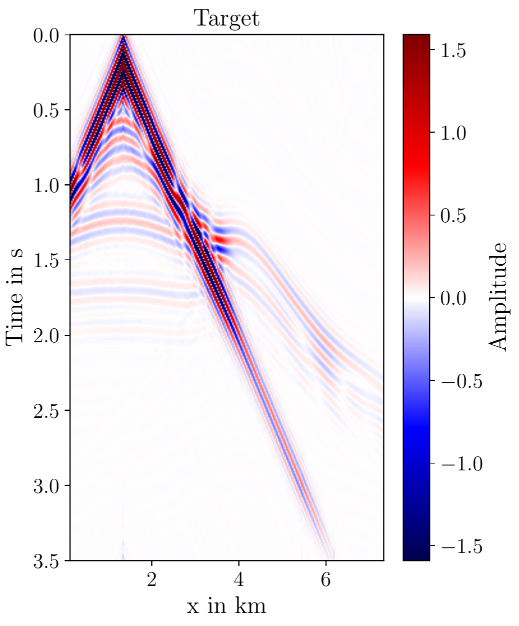


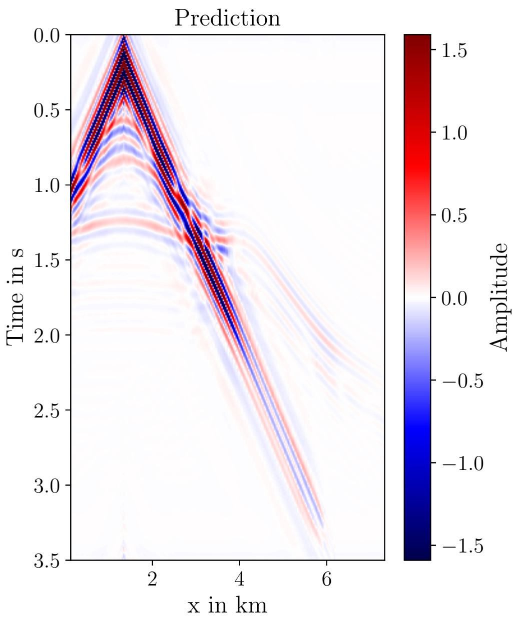
Figure 4.7.: (a) Amplitude spectra of a shot gather for true and extrapolated full-bandwidth data. (b) Snippet of a waveform at 655 m offset. (c) Snippet of a waveform at 1945 m offset. (d) Snippet at 3945 m offset. Note that compared to Figure 4.3, the predicted data are low-pass filtered.



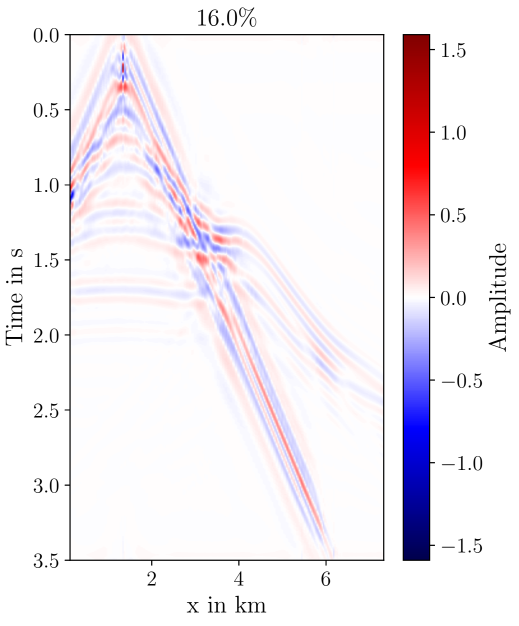
(a) Input



(b) Target



(c) Prediction



(d) Residual

Figure 4.8.: Shot gathers of (a) input, (b) target, (c) prediction and (d) residual of a network trained to extrapolate the full bandwidth. The full-bandwidth data are low-pass filtered to show the fit of the low frequencies only. (a)-(d) have the same clip, which is the 99-percentile of (b). The title of the residual plot gives the NRMS value, which indicates a good fit.

For smaller offsets, the main peak and the surrounding side lobes are recovered fairly well. However, a high-frequency jitter can be seen, e.g., at 1.1 s in Figure 4.12b. The fit of larger offsets is again poorer. Figure 4.12d shows that the first arrival cannot be reconstructed, neither amplitudes nor phases are correct. However, the direct wave is reconstructed well, which is the opposite behavior to the bandwidth extension approach. Figure 4.11 depicts the target data and the predictions. The bias is removed and the NRMS value is reduced to 25.5 %. The poor results in the large offsets are clearly visible in the shot gathers as well. Instead of clearly defined arrivals, the wavefront arrivals in the predicted data are highly disturbed for larger offsets.

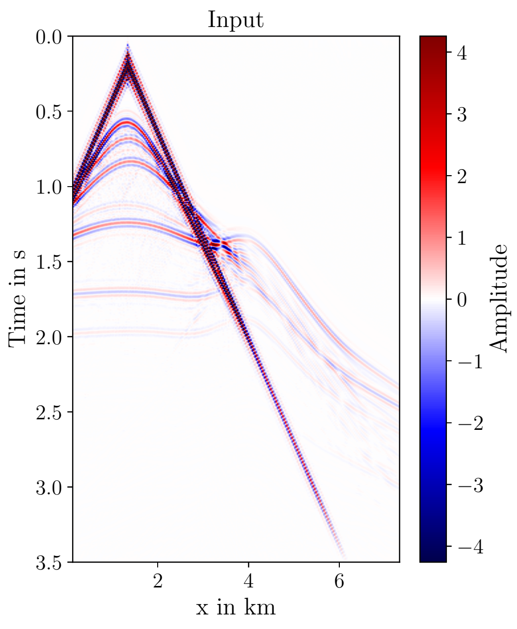
4.3. Comparison of the extrapolation approaches

Both approaches are able to predict the low frequencies reasonably well. The traces match well for small offsets, especially for the bandwidth extension approach. Comparing the low frequencies only, the bandwidth approach seems to be able to predict the phase correctly for the entire waveform, while the low-frequency extrapolation approach gets out of phase. The high-frequency artifacts of the low-frequency approach should not be taken into account for the comparison, as the other data set is low-pass filtered. Using the full bandwidth as target data seems to give better results than training on only the low frequencies. Therefore, the first approach is preferred and used in the following.

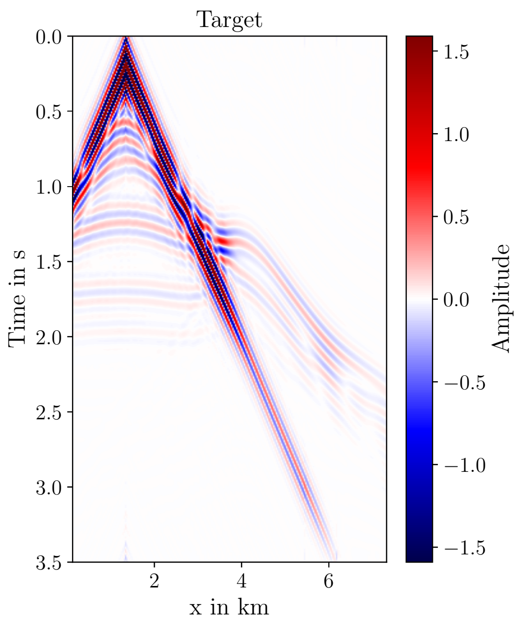
For comparing the resulting NRMS values with the work of Ovcharenko et al. (2020), only the comparisons of the low-frequency content should be considered, as keeping the high-frequency content correctly is not the actual aim of this work. In their work, Ovcharenko et al. (2020) present results for full-offset data ranging from around 25 to 41 %. So the resulting 16.0 % and 25.5 % of this work are in a similar but slightly lower range. Similar to my findings, also the accuracy is deteriorated at larger offset for the results of Ovcharenko et al. (2020, 2017), Sun and Demanet (2019), and Fang et al. (2020). Reasons for a lower accuracy at far-offsets might be that the training data contains more near-offset samples than far-offset samples. Furthermore, near-offset samples have higher amplitudes than far-offset samples, such that strong amplitudes are fitted before smaller amplitudes.

4.4. Parameter testing

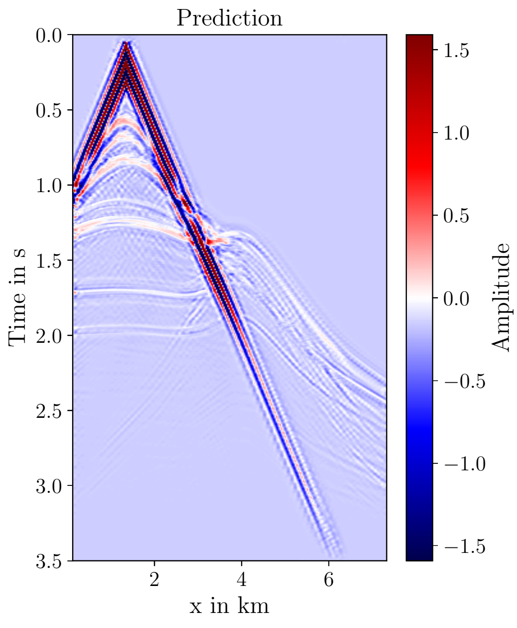
With the aim to improve the fit especially in the larger offsets, different choices of hyper-parameters are tested. With this the influence of the design of the network is analyzed. The high-frequency content of the target data is already given in the input data; therefore, it makes sense to train on residuals and use ResNet blocks as discussed in section 2.1.5.2. Furthermore, a SELU activation function can be advantageous. Lastly, different depths of the network will be tested. At first, the ResNet blocks and the SELU activation function are introduced, while the remaining parameters are kept as before. After training for 100 epochs, the results show a decreased NRMS value of 10.2 % and an overall improved fit, see Figure 4.13c. Rather than the low-frequency artifact along the direct wave, a constant bias for the background can be seen. This could be removed via filtering. As expected, the ResNet blocks and the SELU activation function are suitable for this task and help to improve the result. Lastly, networks with different numbers of layers and numbers of ResNet blocks per layer are tested. To easily reference the different networks, the following notation is introduced: If a network is, e.g., of the structure (2,2,2), it means that its encoder and decoder each consist of three layers with two ResNet blocks per layer. In the following, networks with a (3,3), (3,3,3) and (2,2,2,2) architecture are tested. All architectures give reasonably good results with NRMS values below 20 % (Figure 4.13). The deeper networks perform worse than the shallower networks. The deeper networks have more trainable parameters and might not be fully trained yet, meaning they would have needed more epochs of training. Furthermore, more ResNet blocks per layer increase the performance.



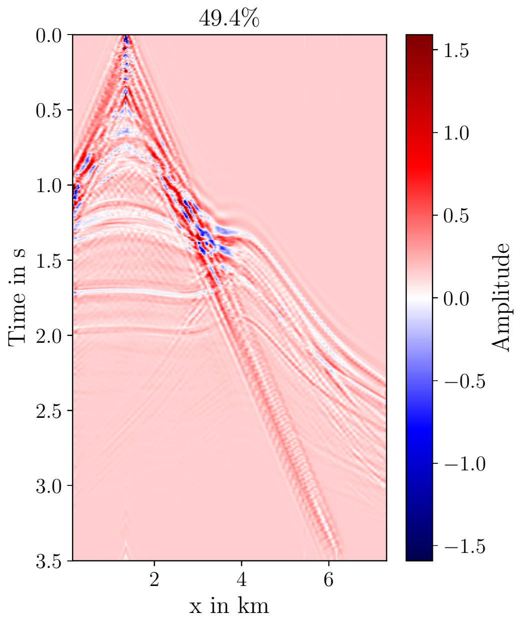
(a) Input



(b) Target

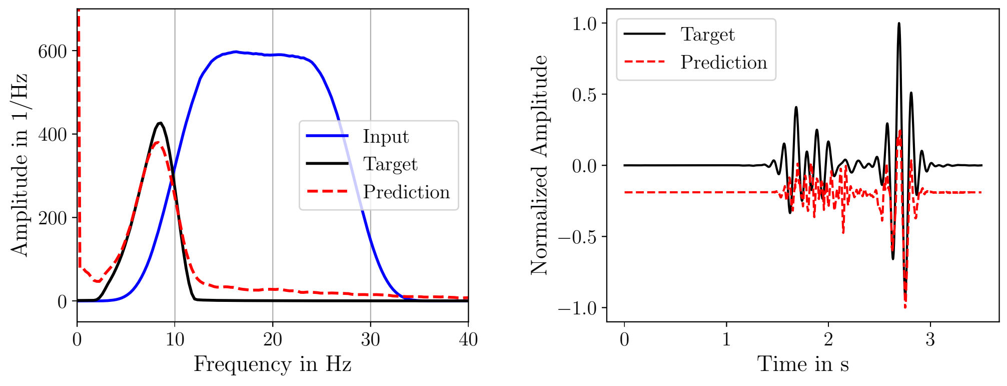


(c) Prediction



(d) Residual

Figure 4.9.: Shot gathers of (a) input, (b) target, (c) prediction and (d) residual of a network trained to extrapolate only the low frequencies. (a)-(d) have the same clip, which is the 99-percentile of (b). The title of the residual plot gives the NRMS value, which indicates a poor fit.



(a) Spectrum **(b)** Snippet of waveform

Figure 4.10.: (a) Amplitude spectra of a shot gather for true and extrapolated low-frequency data. Here, the network is trained on predicting only the low frequencies. (b) Snippet of a waveform at 3945 m offset. The large DC component leads to a strong bias on the trace.

4.5. Loss functions

Finally, different loss functions are tested. So far, a simple mean squared error (MSE) was used, which is defined as

$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}(x_i - y_i)^2 \ , \tag{4.1}$$

where n is the number of data points, x_i the true target data and y_i the predicted data. The mean squared error (MSE) is an L2 norm and usually the go-to loss function in optimization, as its Hessian and gradient are easy to calculate (Wang and Bovik, 2009). Among others, Hu et al. (2021), Fabien-Ouellet (2020b), and Sun and Demanet (2020a) use the MSE as loss function for their networks. However, the MSE over-penalizes large errors (Zhao et al., 2016). Instead, a mean absolute error (MAE) can be used,

$$\mathrm{MAE} = \frac{1}{n}\sum_{i=1}^{n}|x_i - y_i| \ , \tag{4.2}$$

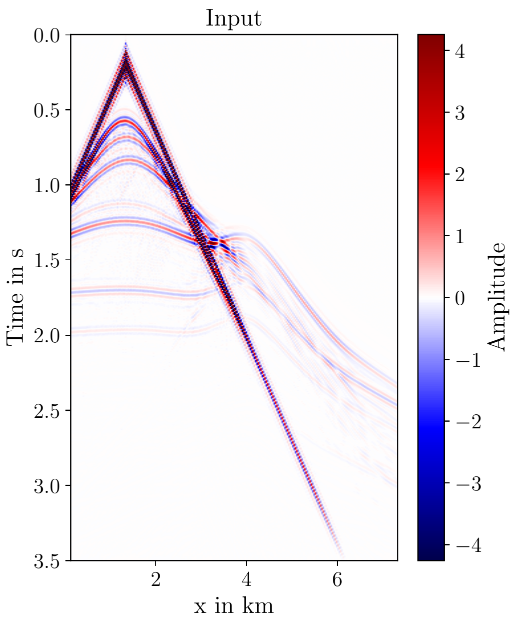
which also has a derivative that is easy to calculate (Zhao et al., 2016). Such an L1 norm was used, for example, by Ovcharenko et al. (2022) and Aharchaou and Baumstein (2020) for training their networks for low-frequency extrapolation. The Huber loss is a loss function that lies between the MSE and the MAE (Huber, 1964). For large errors the Huber loss penalizes just like an L1 norm, while smaller errors are treated with an L2 norm:

$$\mathrm{Huber} = \begin{cases} \frac{1}{2}(x_i - y_i)^2 , & \text{for } |x_i - y_i| \leq \delta \\ \delta \cdot (|x_i - y_i| - \frac{1}{2}\delta), & \text{otherwise} \end{cases} \tag{4.3}$$

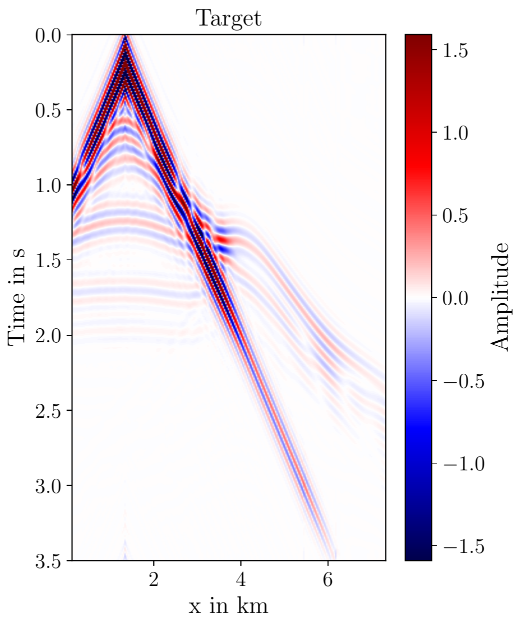
Introducing the structural similarity index measure (SSIM; Wang et al., 2004) allows us to keep the statistical similarity between target and prediction large by comparing luminance, contrast, and structure. It is defined as

$$\mathrm{SSIM}(x, y) = \frac{(2\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)} \ , \tag{4.4}$$

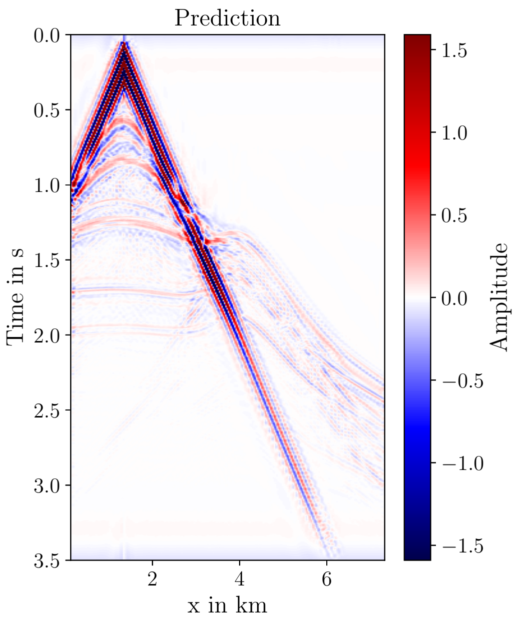
where μ is the mean and σ the standard deviation. The small coefficients C_1 and C_2 ensure stability (Wang et al., 2004). SSIM values are in the range of (-1,1], with 1 meaning $x = y$.



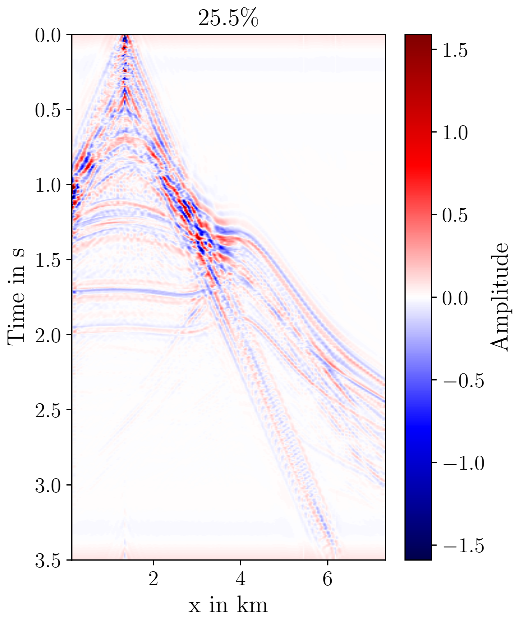
(a) Input



(b) Target

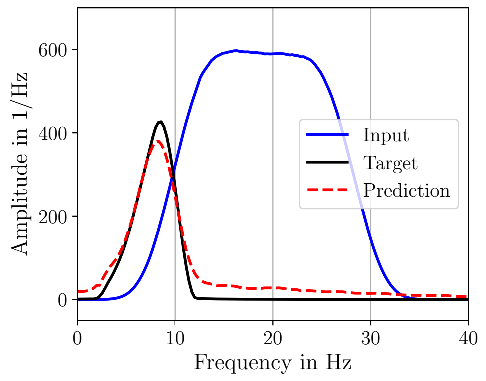


(c) Prediction

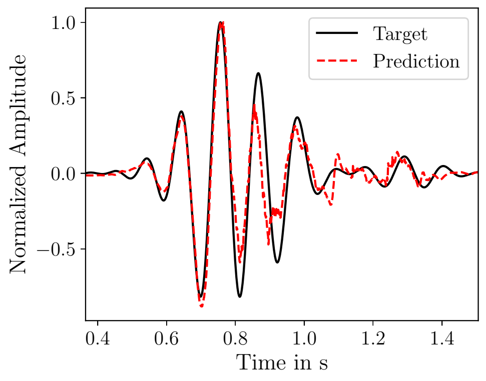


(d) Residual

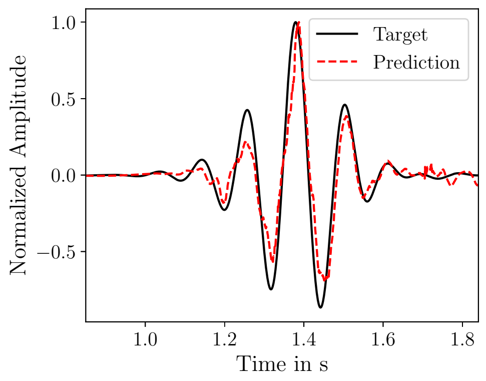
Figure 4.11.: Shot gathers of (a) input, (b) target, (c) prediction and (d) residual of a network trained to extrapolate the full bandwidth. (a - d) have the same clip, which is the 99-percentile of (b). The title of the residual plots gives the NRMS value, which indicates a poor fit. Compared to the results in Figure 4.9, frequencies below 3 Hz are removed.



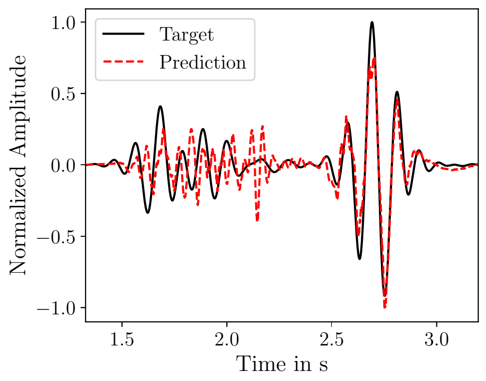
(a) Spectrum



(b) Snippet of waveform

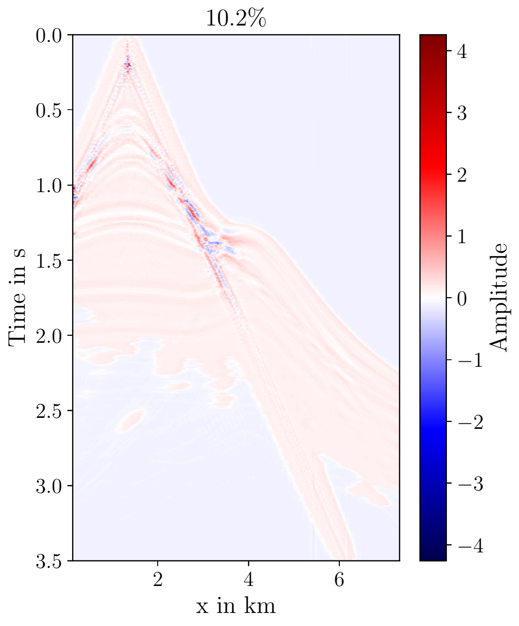


(c) Snippet of waveform

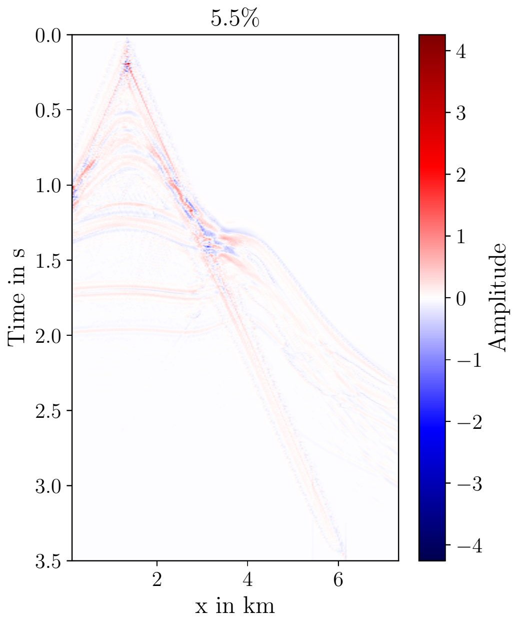


(d) Snippet of waveform

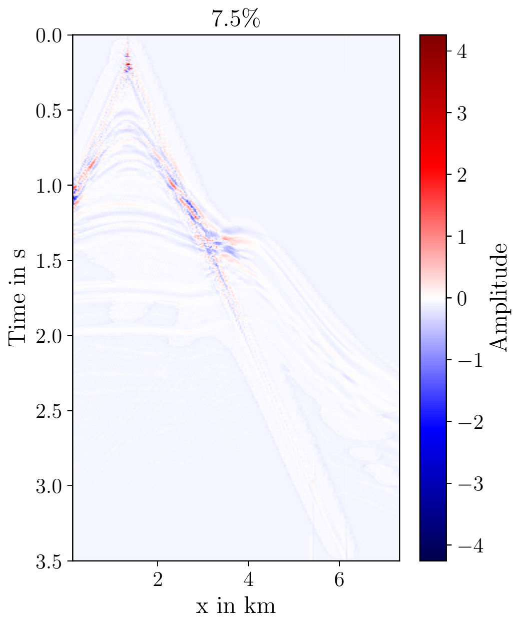
Figure 4.12.: (a) Amplitude spectra of a shot gather for true and extrapolated low-frequency data. Here, the network is trained to predict only the low frequencies. Furthermore a high-pass filter is applied to remove the DC component and the very low frequencies. (b) Snippet of a waveform at 655 m offset. (c) Snippet of a waveform at 1945 m offset. (d) Snippet at 3945 m offset.



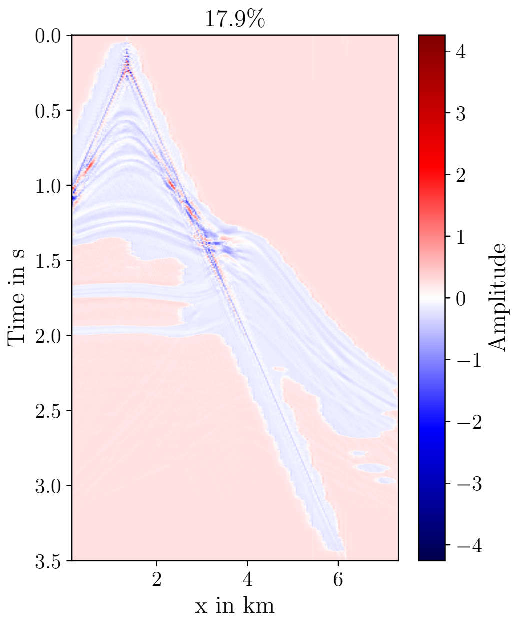
(a) ResNet+SELU+(2,2,2)



(b) ResNet+SELU+(3,3)



(c) ResNet+relu+(3,3,3)



(d) ResNet+SELU+(2,2,2,2)

Figure 4.13.: Residual plots for bandwidth extension approaches with different network architectures. The title of the residual plots shows the NRMS value.

Sun and Demanet (2022) also use the SSIM to evaluate the similarity of their extrapolated and true low-frequency data. The MSE and the SSIM functions can be combined by weighting them:

$$\mathrm{LOSS}(x, y) = \alpha \mathrm{MSE}(x, y) - (1 - \alpha)\mathrm{SSIM}(x, y) \ , \tag{4.5}$$

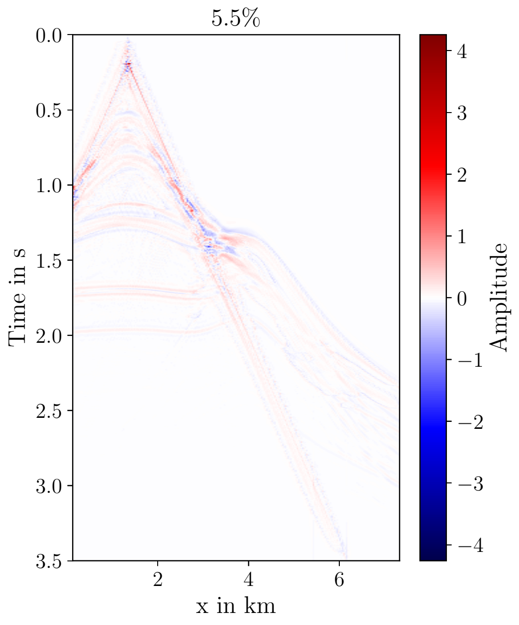
where α is a weighting factor. This forces the network to get a low MSE while at the same time keeping the statistical similarity high. For the application, the SSIM has to be scaled to the interval (0,1] in order to get meaningful loss values. Similarly, the MAE and the SSIM loss can be combined.

To test how well the individual loss functions work on the data set, networks are trained for 100 epochs with a (3,3) architecture, SELU activation functions and different loss functions. Figure 4.14 depicts the resulting residuals for different loss functions. It can be seen that reasonable results with NRMS values smaller than 10 % can be obtained with all loss functions. The MSE loss and the combination of MSE and SSIM loss show similar results with only small errors in the amplitudes. Also the difference between MAE on the one hand and MAE in combination with SSIM loss on the other hand is small. The SSIM loss needs additional calculations; therefore, only the MAE or MSE loss alone will be considered. Training with the Huber loss introduces low-frequency artifacts again, but the overall fit is good. A closer look at the residuals of the MAE and MSE loss reveals, that the MAE loss shows a better performance than the MSE loss at far offsets. Also, the reflection events at around 1.7 s and 2.0 s are better resolved with the MAE loss. Even though, the MAE loss has larger errors along the direct wave, a better reconstruction of the far-offset arrivals and later reflections is preferred. Therefore, the MAE loss is the chosen loss function for the final network.

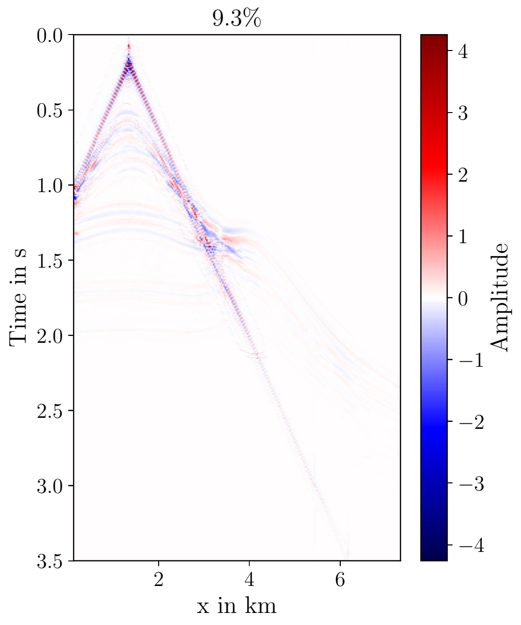
4.6. Final results

With the results from the previous tests, a UNet with ResNet blocks, SELU activation function, (3,3) architecture and an MAE loss is taken as final network. Figure 4.15 shows the target and the predicted gather for a single shot and their residual. The NRMS value of 9.3 % indicates a good fit for the full bandwidth. The two shot gathers are very similar. Only the direct wave suffers from larger errors. In Figure 4.16 the spectrum and the waveforms for different offsets are compared to the true data. In contrast to the tests, the spectrum does not fit the spectrum of the target completely and is off by a small factor. Looking at the normalized traces, it can be seen that the waveforms match, so indeed the network is predicting the values wrong by a constant factor. Compared to previous results as, e.g., depicted in Figure 4.7, especially the far-offset trace has improved significantly. Instead of following the input, the extrapolated trace matches with the target trace. The trace at 3945 m offset has some low-amplitude, high-frequency artifacts at around 1.2 s and 1.6 s. This is not a problem for FWI, as in FWI only the lower frequencies will be used anyway.

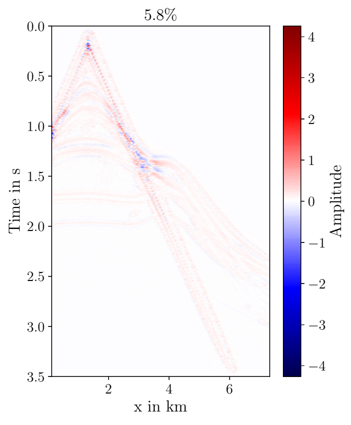
Furthermore, the extrapolated low frequencies are compared separately by applying the same low-pass filter as in section 4.1. Figure 4.17 shows a good fit for the low frequencies as well. However, the large amplitudes at around 4 km are not correctly recovered. Furthermore, the later reflections at around 1.7 s and 2.0 s are of very low amplitude. The resulting NRMS value of 13.9 % verifies the good fit. Compared to the results achieved by Ovcharenko et al. (2020), the network presented in this work performed better. However, this comparison is not fair and the results should not be used blindly to judge the two different approaches in general, as the used data sets differ a lot in complexity. In this work, acoustic data is used, while Ovcharenko et al. (2020) use elastic land data that contain a strong ground roll.



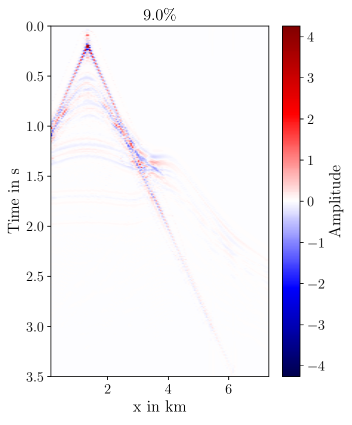
(a) MSE loss



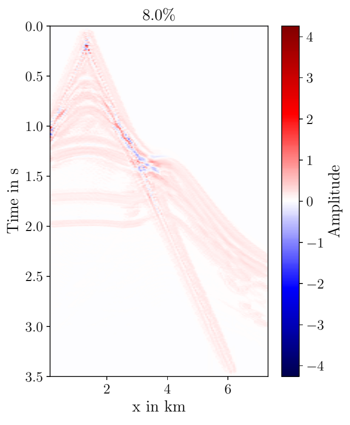
(b) MAE loss



(c) MSE+SSIM loss

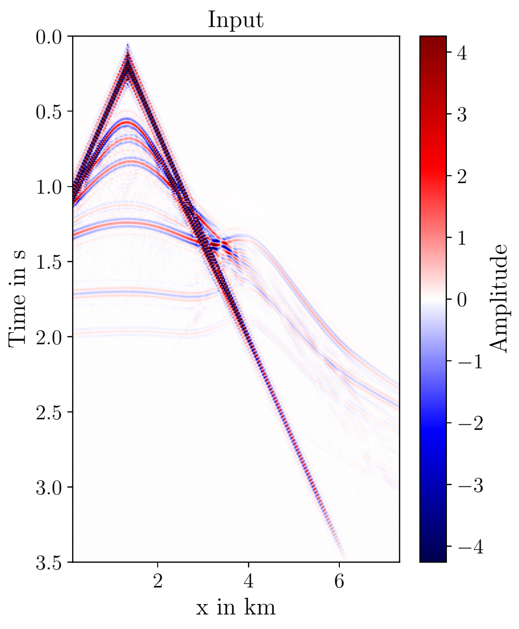


(d) MAE+SSIM loss

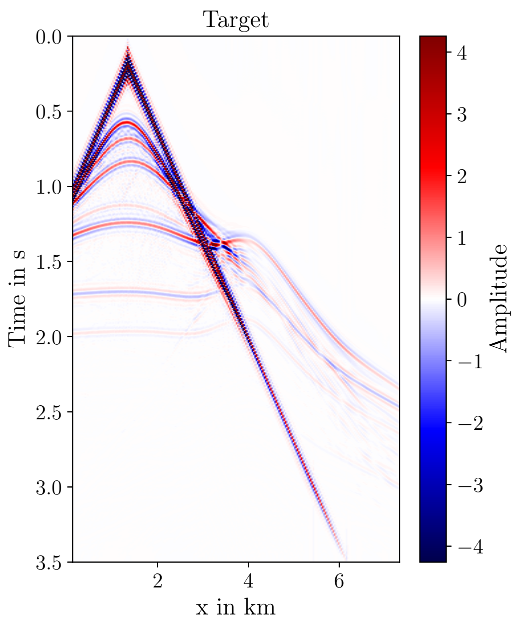


(e) Huber loss

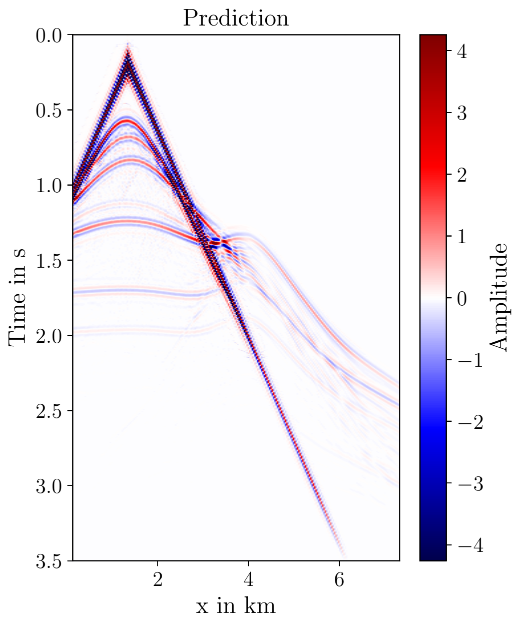
Figure 4.14.: Residual plots between target and prediction for (a) an MSE, (b) an MAE, (c) and MSE+SSIM, (d) and MAE+SSIM, and (e) a Huber loss function. The percentage above each plot is the NRMS value.



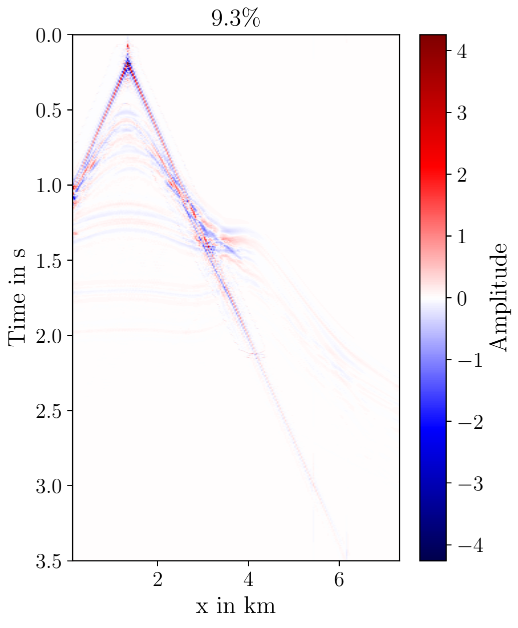
(a) Input



(b) Target

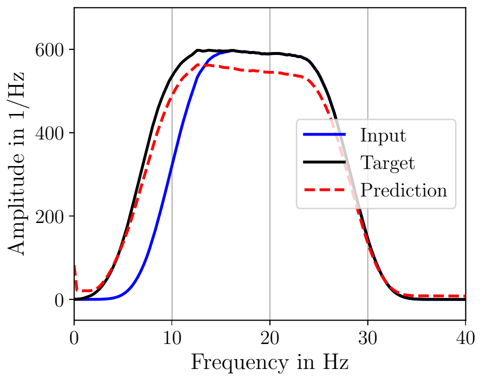


(c) Prediction

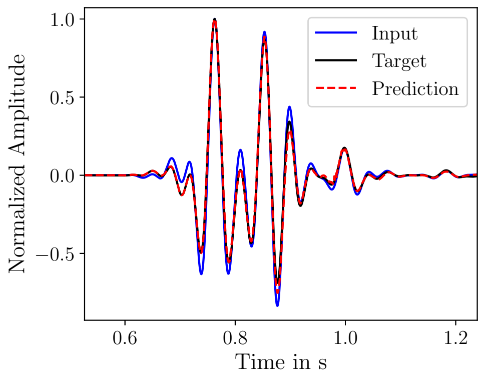


(d) Residual

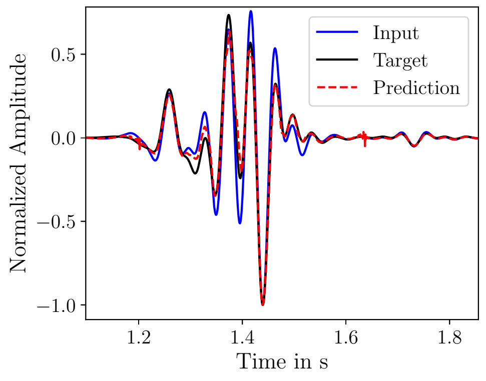
Figure 4.15.: Shot gathers of (a) input, (b) target, (c) prediction and (d) residual of the final network trained to extrapolate the full bandwidth. (a)-(d) have the same clip, which is the 99-percentile of (b). The title of the residual plot gives the NRMS value, which indicates a good fit.



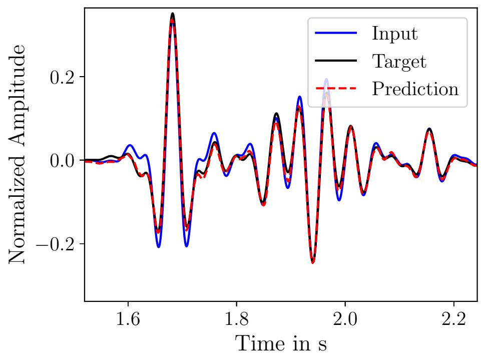
(a) Spectrum



(b) Snippet of waveform



(c) Snippet of waveform

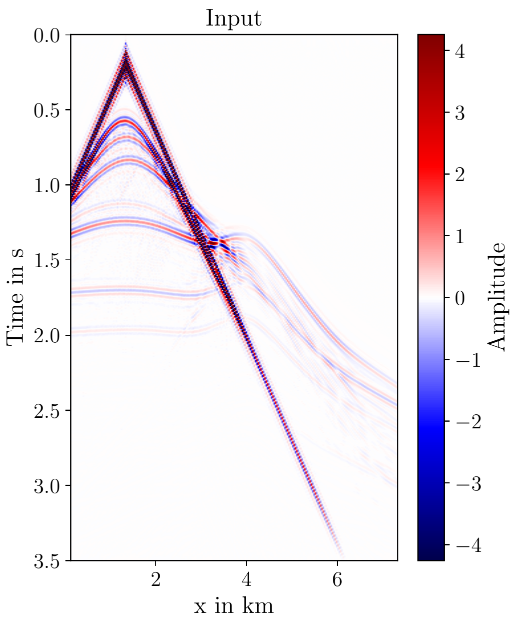


(d) Snippet of waveform

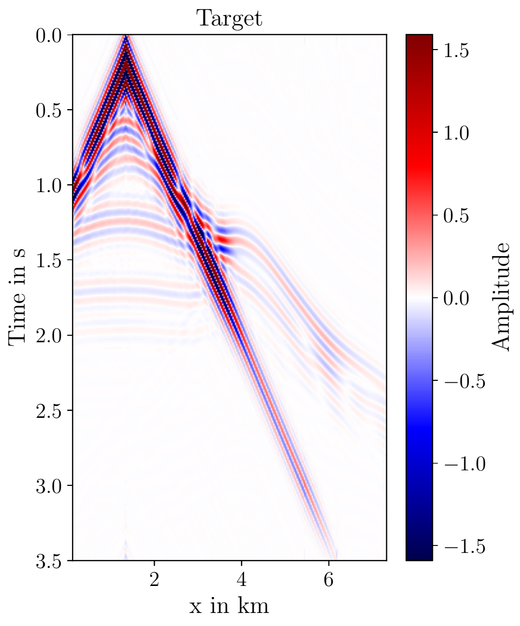
Figure 4.16.: (a) Amplitude spectra of a shot gather for true and extrapolated full-bandwidth data of the final network. (b) Snippet of a waveform at 655 m offset. (c) Snippet of a waveform at 1945 m offset. (d) Snippet at 3945 m offset.

To further study the performance of the network, the low-frequency spectrum and waveforms are investigated in Figure 4.18. Still, the extrapolated data show smaller amplitudes in the spectrum, but the trend is similar to the true data. The extrapolated waveforms match well with the target data. For all traces some peaks are not entirely correct in amplitude, but the phases match very well. The surprisingly good fit at far offsets is worth noting, as the previous results of this work showed increasing errors for far-offset traces.

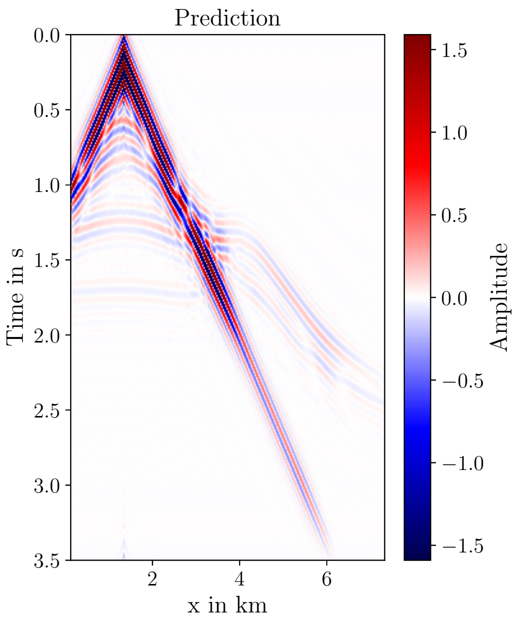
Additional shot gathers and traces are analyzed in section C in the appendix. The NRMS value for all reviewed shots is around 9.5 %, see Figure C.2. Most of the traces show a similar good fit (Figure C.4). However, for offsets larger than 6900 m the DC component becomes strong enough to disturb the waveform. Furthermore, for a few traces at various offsets, the prediction follows partly the input and not the target. The network still shows a great overall performance. For only the low-frequency band, the NRMS is in a range of 13 to 16 % (Figure C.3). The traces match well; however, some arrivals have a significantly lower amplitude than in the true trace (Figure C.5). Furthermore, some phases are not correctly reconstructed. Nevertheless, the overall impression is still good.



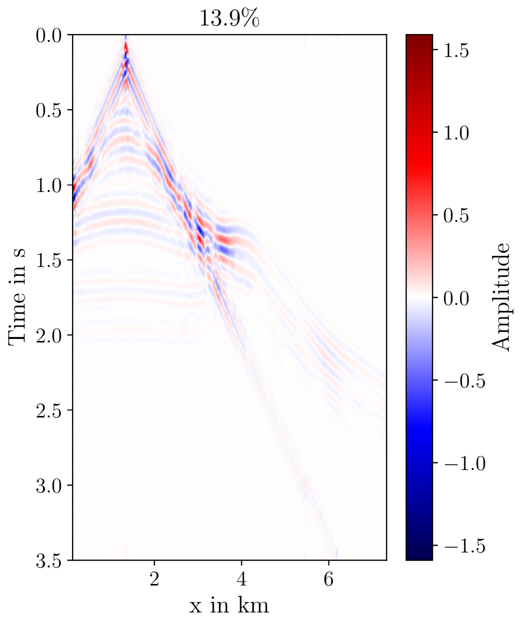
(a) Input



(b) Target

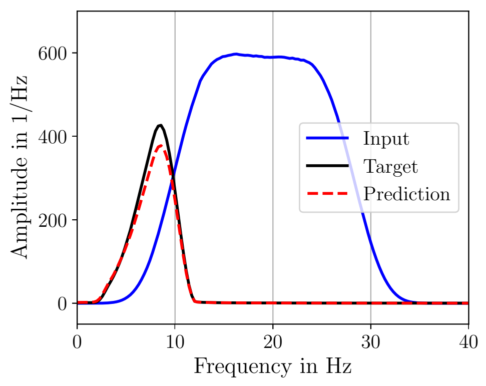


(c) Prediction

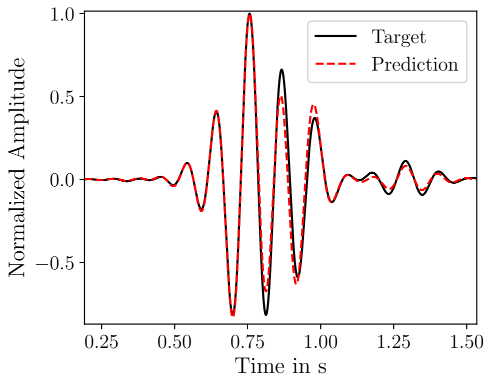


(d) Residual

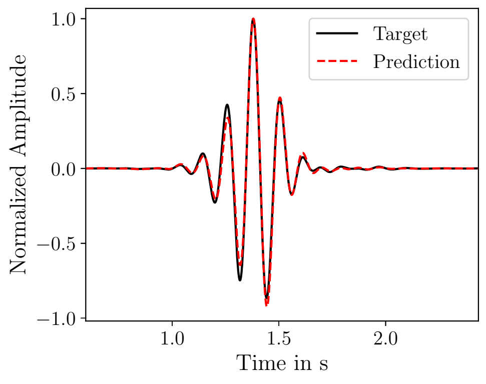
Figure 4.17.: Shot gathers of (a) input, (b) target, (c) prediction and (d) residual of the final network trained to extrapolate the full bandwidth. The data is low-pass filtered to assess the fit of the low frequencies. (a - d) have the same clip, which is the 99-percentile of (b). The title of the residual plots gives the NRMS value, which indicates a good fit.



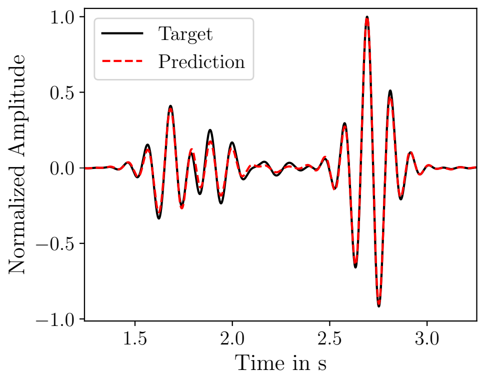
(a) Spectrum



(b) Snippet of waveform



(c) Snippet of waveform



(d) Snippet of waveform

Figure 4.18.: (a) Amplitude spectra of a shot gather for true and extrapolated full-bandwidth data of the final network. (b) Snippet of a waveform at 655 m offset. (c) Snippet of a waveform at 1945 m offset. (d) Snippet at 3945 m offset. Note that the data are the low-pass filtered version of the data in Figure 4.16.

5. Conclusions

This thesis aimed to investigate the potential of neural networks for the task of low-frequency extrapolation. This is motivated by the fact that low frequencies can improve FWI results by overcoming the cycle-skipping problem. The development of a deep learning workflow for low-frequency extrapolation consisted of several steps: generating subsurface models, simulating seismic data, developing a pre-processing workflow by reconstruction tests and the actual extrapolation of low frequencies.

To be able to model training data, velocity models were generated on the basis of the subsurface structures around the Asse II salt mine. The shape of the salt body as well as the number and shape of additional interfaces was randomly varied. Additionally, a water column was taken as first layer. The resulting velocity models were simple and geologically plausible. With the velocity models, synthetic seismic data were simulated in acoustic media. A flat-spectrum wavelet was used as source signal and a high-pass filter was designed for the input data.

To develop a suitable pre-processing workflow, reconstruction tests were performed. For that purpose, the output of the network should be as close to the input as possible. A U-Net as proposed by Ronneberger et al. (2015) was utilized for this task. The essential steps of the pre-processing include the 1.5-integration of the data, normalizing and standardizing the data with a global mean and standard deviation, and splitting the data into smaller patches.

For extrapolating the low frequencies, two approaches were investigated. The full-bandwidth approach was able to predict the phases correctly while extrapolating solely the low frequencies resulted in slightly wrong phases. To further improve the results, the influence of the network architecture was studied. Extrapolating the full bandwidth instead of only the low frequencies, which is the usual approach in the related literature, provides the possibility to utilize ResNet blocks for the task. Furthermore, the performance with different loss functions was assessed. While adding a structural similarity index measure to the loss function did not result in any great changes, the change to a mean absolute error improved the results. The results of the final network design were assessed by comparing the target and prediction for the entire frequency range and the low-frequency range. The extrapolation of the missing frequencies was successful. Besides small errors in the amplitudes, the network can predict the low frequencies successfully. The fit, particularly in the far offsets, has improved compared to the first try of full-bandwidth extrapolation especially in the low-frequency band. An entirely correct extrapolation is not needed for using the data in FWI, as the low frequencies are only intended to improve the model for FWI not a correct imaging of the low-frequency content. Thus, small errors in the low frequencies do not make a large difference.

The validity of the results is limited in several ways because this thesis only had the intention to study the potential of such networks. For acoustic data from layers above a salt dome, the network shows good performance. However, field data is neither purely acoustic nor do they have a good signal-to-noise ratio compared to the utilized synthetic data. Therefore, it is important to test the ability of the network to generalize the extrapolation to different settings, such as elastic data, data with a lower signal-to-noise ratio, or data from subsurface models with completely different structures. First, the already trained network can be confronted with such new data and the generalization performance of it can be studied. Then, in case of failure, the network can also be trained from scratch with the new data sets to improve the performance. To improve the noise robustness, also data augmentation during training could be used to add different levels and kinds of noise to the same set of data.

The extrapolated frequencies of this work could be utilized to improve FWI. I suggest performing FWI in multiple steps on the extrapolated full-bandwidth data for frequencies below 8 Hz. Afterwards, the original data can be used to continue the FWI on an improved starting model. It might also be possible to perform the FWI solely on the extrapolated full-bandwidth data, but this comes with the risk of performing FWI on data that might contain artifacts.

The machine learning approach presented in this thesis does not consider whether the results are physically plausible. Therefore, it might be worth to investigate the potential of combining physical equations with neural networks for the task of low-frequency extrapolation. Such an approach can be taken, e.g., with physics-informed (Raissi et al., 2019) or physics-guided neural networks (Daw et al., 2017). These networks constrain the predictions by either an additional penalty on the loss function or a combination of such a penalty and a calculation of additional input features with a physics based model, respectively.

All in all, the utilized neural network successfully extrapolates low frequencies for synthetic data in a controlled setup. The results might deteriorate for very large offsets. Nevertheless, the predicted data is still close to the true target, which will improve the results of FWI.

Acknowledgements

First of all, I want to thank Prof. Dr. Thomas Bohlen for giving me the freedom to choose a topic in the field of machine learning and for the support with SOFI concerning the acquisition design and setup. Furthermore, I want to acknowledge Prof. Dr. Dirk Gajewski for accepting to be the second reviewer of my thesis.

Next, many thanks go to Thomas Hertweck, Lars Houpt and Jan Walda for their supervision, countless fruitful discussions, great input, taking the time to help with any technical problems, and proofreading my thesis. I further want to acknowledge Jan Walda for providing me with his machine learning framework and Thomas Hertweck for his Python package for the in- and output of seismic data.

Furthermore, I want to thank more people who helped me in different ways during the process of this thesis. Here I want to name Sonia Șortan for helping me getting started with SOFI and providing me with the subsurface model and Marie Gärtner for proofreading and for all the evenings spent cooking together and playing board games. Many thanks go as well to my fellow students and friends for the great time we had together during lunch breaks and all the evening and weekend activities.

In addition, I want to thank my flatmate Josephine for enduring countless conversations about the progress of my thesis and surprising me with a cake or some food every now and then to keep the motivation up. Last but not least, I want to thank my family for supporting me and my studies. Especially Pauline helped me by proofreading parts of my thesis and giving me tips and tricks concerning figures and tables.

Bibliography

Abadi, M., Agarwal, A., et al. (2015). *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems.* URL: https://www.tensorflow.org (visited on 11/10/2022).

Aharchaou, M., Baumstein, A., Vdovina, T., Lu, R., and Neumann, E. (2021). "Deep Learning of Bandwidth Extension from Seabed Seismic". In: *82nd EAGE Conference and Exhibition Extended Abstracts.* European Association of Geoscientists & Engineers. DOI: 10.3997/2214-4609.202011111.

Aharchaou, M. and Baumstein, A. (2020). "Deep learning-based artificial bandwidth extension: Training on ultrasparse OBN to enhance towed-streamer FWI". *The Leading Edge* 39.10, pp. 718–726. DOI: 10.1190/tle39100718.1.

Alsallakh, B., Kokhlikyan, N., Miglani, V., Yuan, J., and Reblitz-Richardson, O. (2021). "Mind the Pad – CNNs Can Develop Blind Spots". In: *International Conference on Learning Representations.* Vienna, Austria. URL: https://openreview.net/forum?id=m1CD7tPubNy (visited on 11/10/2022).

Beyreuther, M., Barsch, R., Krischer, L., Megies, T., Behr, Y., and Wassermann, J. (2010). "ObsPy: A Python Toolbox for Seismology". *Seismological Research Letters* 81.3, pp. 530–533. DOI: 10.1785/gssrl.81.3.530.

Bishop, C. M. (2006). *Pattern Recognition and Machine Learning.* New York: Springer. DOI: 10.1007/978-0-387-45528-0.

Bohlen, T., De Nil, D., Köhn, D., and Jetschny, S. (2016). *SOFI2D – Seismic modeling with finite differences: 2D elastic and viscoelastic version (updated 2022).* URL: https://git.scc.kit.edu/GPIAG-Software/SOFI2D (visited on 11/10/2022).

Bohlen, T. and Saenger, E. H. (2006). "Accuracy of heterogeneous staggered-grid finite-difference modeling of Rayleigh waves". *Geophysics* 71.4. DOI: 10.1190/1.2213051.

Bourlard, H. and Kamp, Y. (1988). "Auto-association by multilayer perceptrons and singular value decomposition". *Biological Cybernetics* 59.4, pp. 291–294. DOI: 10.1007/BF00332918.

Bunks, C., Saleck, F. M., Zaleski, S., and Chavent, G. (1995). "Multiscale seismic waveform inversion". *Geophysics* 60.5, pp. 1457–1473. DOI: 10.1190/1.1443880.

Choi, Y. and Alkhalifah, T. (2012). "Application of multi-source waveform inversion to marine streamer data using the global correlation norm". *Geophysical Prospecting* 60.4, pp. 748–758. DOI: 10.1111/j.1365-2478.2012.01079.x.

Clevert, D. A., Unterthiner, T., and Hochreiter, S. (2016). "Fast and accurate deep network learning by exponential linear units (ELUs)". In: *International Conference on Learning Representations.* San Juan, Puerto Rico. DOI: 10.48550/arXiv.1511.07289.

Cohen, J. K. and Stockwell, J. J. W. (2022). *CWP/SU: Seismic Unix Release No. 44: a free package for seismic research and processing.* Center for Wave Phenomena, Colorado School of Mines. URL: https://github.com/JohnWStockwellJr/SeisUnix (visited on 11/10/2022).

Courant, R., Friedrichs, K., and Lewy, H. (1928). "Über die partiellen Differenzengleichungen der mathematischen Physik". *Mathematische Annalen* 100, pp. 32–74. DOI: 10.1007/BF01448839.

Crameri, F. (2021). *Scientific colour maps.* Version 7.0.1. DOI: 10.5281/zenodo.5501399.

Daw, A., Karpatne, A., Watkins, W., Read, J., and Kumar, V. (2017). “Physics-guided Neural Networks (PGNN): An Application in Lake Temperature Modeling”. *arXiv.* DOI: 10.48550/arXiv.1710.11431.

Dumoulin, V. and Visin, F. (2016). “A guide to convolution arithmetic for deep learning”. *arXiv.* DOI: 10.48550/arXiv.1603.07285.

Fabien-Ouellet, G. (2020a). “Generating seismic low frequencies with a deep recurrent neural network for full waveform inversion”. In: *1st EAGE Conference on Seismic Inversion.* European Association of Geoscientists & Engineers. DOI: 10.3997/2214-4609.202037023.

Fabien-Ouellet, G. (2020b). “Low frequency generation and denoising with recursive convolutional neural networks”. In: *SEG Technical Program Expanded Abstracts*, pp. 870–874. DOI: 10.1190/segam2020-3428270.1.

Fang, J., Zhou, H., Li, Y. E., Zhang, Q., Wang, L., Sun, P., and Zhang, J. (2020). “Data-driven low-frequency signal recovery using deep-learning predictions in full-waveform inversion”. *Geophysics* 85.6, A37–A43. DOI: 10.1190/geo2020-0159.1.

Forbriger, T., Groos, L., and Schäfer, M. (2014). “Line-source simulation for shallow-seismic data. Part 1: Theoretical background”. *Geophysical Journal International* 198.3, pp. 1387–1404. DOI: 10.1093/gji/ggu199.

Glorot, X., Bordes, A., and Bengio, Y. (2011). “Deep Sparse Rectifier Neural Networks”. In: *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics.* Ed. by G. Gordon, D. Dunson, and M. Dudík. Vol. 15. Fort Lauderdale, FL, USA: PMLR, pp. 315–323. URL: https://proceedings.mlr.press/v15/glorot11a.html (visited on 11/10/2022).

Gogoi, M. and Begum, S. A. (2017). “Image Classification Using Deep Autoencoders”. In: *2017 IEEE International Conference on Computational Intelligence and Computing Research (ICCIC).* DOI: 10.1109/ICCIC.2017.8524276.

Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep learning.* MIT Press. URL: http://www.deeplearningbook.org (visited on 11/10/2022).

Guerrero, R., Qin, C., Bowles, C., Chen, L., Joules, R., Wolz, R., Valdés-Hernández, M., Dickie, D., Wardlaw, J., and Rueckert, D. (2018). “White matter hyperintensity and stroke lesion segmentation and differentiation using convolutional neural networks”. *NeuroImage: Clinical* 17, pp. 918–934. DOI: 10.1016/j.nicl.2017.12.022.

Harris, C. R., Millman, K. J., et al. (2020). “Array programming with NumPy”. *Nature* 585.7825, pp. 357–362. DOI: 10.1038/s41586-020-2649-2.

He, K., Zhang, X., Ren, S., and Sun, J. (2015). “Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification”. In: *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1026–1034. DOI: 10.48550/arXiv.1502.01852.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). “Deep Residual Learning for Image Recognition”. In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778. DOI: 10.1109/CVPR.2016.90.

Hu, W. (2014). “FWI without low frequency data – beat tone inversion”. In: *SEG Technical Program Expanded Abstracts.* Society of Exploration Geophysicists, pp. 4172–4177. DOI: 10.1190/segam2014-0978.1.

Hu, W., Chen, J., Liu, J., and Abubakar, A. (2018). “Retrieving Low Wavenumber Information in FWI: An Overview of the Cycle-Skipping Phenomenon and Solutions”. *IEEE Signal Processing Magazine* 35.2, pp. 132–141. DOI: 10.1109/MSP.2017.2779165.

Hu, W., Jin, Y., Wu, X., and Chen, J. (2020). “Physics-guided self-supervised learning for low frequency data prediction in FWI”. In: *SEG Technical Program Expanded Abstracts.* Society of Exploration Geophysicists, pp. 875–879. DOI: 10.1190/segam2020-3423396.1.

Hu, W., Jin, Y., Wu, X., and Chen, J. (2021). “Progressive transfer learning for low frequency data prediction in full waveform inversion”. *Geophysics* 86.4, R369–R382. DOI: 10.1190/geo2020-0598.1.

Huang, K.-Y., Liu, W. H., and Chang, I. C. (1989). “Hopfield model of neural networks for detection of bright spots”. In: *SEG Technical Program Expanded Abstracts*. Society of Exploration Geophysicists, pp. 444–446. DOI: 10.1190/1.1889690.

Huber, P. J. (1964). “Robust Estimation of a Location Parameter”. *The Annals of Mathematical Statistics* 35.1, pp. 73–101. DOI: 10.1214/aoms/1177703732.

Ioffe, S. and Szegedy, C. (2015). “Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift”. In: *Proceedings of the 32nd International Conference on Machine Learning*. Ed. by F. Bach and D. Blei. Vol. 37. Lille, France: PMLR, pp. 448–456. URL: https://proceedings.mlr.press/v37/ioffe15.html (visited on 11/10/2022).

Jin, Y., Hu, W., Wu, X., and Chen, J. (2018). “Learn Low Wavenumber Information in FWI via Deep Inception Based Convolutional Networks”. In: *SEG Technical Program Expanded Abstracts*. Society of Exploration Geophysicists, pp. 2091–2095. DOI: 10.1190/segam2018-2997901.1.

Jin, Y., Hu, W., Wu, X., and Chen, J. (2021). “Efficient progressive transfer learning for low-frequency reflection seismic data prediction”. In: *First International Meeting for Applied Geoscience & Energy Expanded Abstracts*. Society of Exploration Geophysicists, pp. 777–781. DOI: 10.1190/segam2021-3594767.1.

Kazei, V., Ovcharenko, O., Plotnitskii, P., Peter, D., Zhang, X., and Alkhalifah, T. (2021). “Mapping Full Seismic Waveforms to Vertical Velocity Profiles by Deep Learning”. *Geophysics* 86.5, R711–R721. DOI: 10.1190/geo2019-0473.1.

Kingma, D. P. and Ba, J. L. (2015). “Adam: A method for stochastic optimization”. In: *3rd International Conference on Learning Representations*. San Diego, CA, USA. DOI: 10.48550/arXiv.1412.6980.

Klambauer, G., Unterthiner, T., Mayr, A., and Hochreiter, S. (2017). “Self-Normalizing Neural Networks”. In: *Advances in Neural Information Processing Systems*. Ed. by I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett. Curran Associates, Inc., pp. 971–980. URL: https://proceedings.neurips.cc/paper/2017/file/5d44ee6f2c3f71b73125876103c8f6c4-Paper.pdf (visited on 11/10/2022).

Köhn, D. (2011). “Time domain 2D elastic full waveform tomography”. PhD thesis. Kiel University. URL: https://macau.uni-kiel.de/receive/diss_mods_00006786 (visited on 11/10/2022).

Komatitsch, D. and Martin, R. (2007). “An unsplit convolutional perfectly matched layer improved at grazing incidence for the seismic wave equation”. *Geophysics* 72.5, pp. 155–167. DOI: 10.1190/1.2757586.

Kragh, E. and Christie, P. (2002). “Seismic repeatability, normalized rms, and predictability”. *The Leading Edge* 21.7, pp. 640–647. DOI: 10.1190/1.1497316.

Lay, T. and Wallace, T. C. (1995). *Modern global seismology*. Academic Press.

Levander, A. R. (1988). “Fourth-order finite-difference P-SV seismograms”. *Geophysics* 53.11, pp. 1425–1436. DOI: 10.1190/1.1442422.

Li, W., Wang, G., Fidon, L., Ourselin, S., Cardoso, M. J., and Vercauteren, T. (2017). “On the compactness, efficiency, and representation of 3D convolutional networks: Brain parcellation as a pretext task”. In: *Information Processing in Medical Imaging, 25th International Conference*. Ed. by M. Niethammer, M. Styner, S. Aylward, H. Zhu, I. Oguz, P. Yap, and D. Shen. Cham: Springer, pp. 348–360. DOI: 10.1007/978-3-319-59050-9_28.

Li, Y. E. and Demanet, L. (2016). “Full-waveform inversion with extrapolated low-frequency data”. *Geophysics* 81.6, R339–R348. DOI: 10.1190/geo2016-0038.1.

Liu, N., He, T., Tian, Y., Wu, B., Gao, J., and Xu, Z. (2020). “Common-azimuth seismic data fault analysis using residual UNet”. *Interpretation* 8, SM25–SM37. DOI: 10.1190/INT-2019-0173.1.

Liu, X., Xue, P., and Li, Y. (1989). “Neural network method for tracing seismic events”. In: *SEG Technical Program Expanded Abstracts.* Society of Exploration Geophysicists, pp. 716–718. DOI: 10.1190/1.1889749.

Louboutin, M., Guasch, L., and Herrmann, F. J. (2017). “Data Normalization Strategies for Full-Waveform Inversion”. In: *79th EAGE Conference and Exhibition Extended Abstracts.* European Association of Geoscientists & Engineers. DOI: 10.3997/2214-4609.201700720.

Ma, Y., Hale, D., Gong, B., and Meng, Z. J. (2012). “Image-guided sparse-model full waveform inversion”. *Geophysics* 77.4, R189–R198. DOI: 10.1190/geo2011-0395.1.

Maas, A. L., Hannun, A. Y., and Ng, A. Y. (2013). “Rectifier nonlinearities improve neural network acoustic models”. In: *Proceedings of the 30th International Conference on Machine Learning.* Vol. 28. URL: http://robotics.stanford.edu/~amaas/papers/relu_hybrid_icml2013_final.pdf (visited on 11/10/2022).

Masci, J., Meier, U., Cireşan, D., and Schmidhuber, J. (2011). “Stacked convolutional auto-encoders for hierarchical feature extraction”. In: *Artificial Neural Networks and Machine Learning – ICANN 2011.* Ed. by T. Honkela, W. Duch, M. Girolami, and S. Kaski. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, pp. 52–59. DOI: 10.1007/978-3-642-21735-7_7.

Moczo, P., Kristek, J., and Halada, L. (2004). *The Finite-Difference Method for Seismologists. An Introduction.* Comenius University, Bratislava. URL: ftp://ftp.nuquake.sk/pub/Papers (visited on 11/10/2022).

Nakayama, S. and Blacquière, G. (2021). “Machine-learning-based data recovery and its contribution to seismic acquisition: Simultaneous application of deblending, trace reconstruction, and low-frequency extrapolation”. *Geophysics* 86.2, P13–P24. DOI: 10.1190/geo2020-0303.1.

Odena, A., Dumoulin, V., and Olah, C. (2016). “Deconvolution and Checkerboard Artifacts”. *Distill.* DOI: 10.23915/distill.00003.

Ovcharenko, O., Kazei, V., Alkhalifah, T. A., and Peter, D. B. (2022). “Multi-Task Learning for Low-Frequency Extrapolation and Elastic Model Building From Seismic Data”. *IEEE Transactions on Geoscience and Remote Sensing* 60, pp. 1–17. DOI: 10.1109/TGRS.2022.3185794.

Ovcharenko, O., Kazei, V., Kalita, M., Peter, D., and Alkhalifah, T. (2019). “Deep learning for low-frequency extrapolation from multioffset seismic data”. *Geophysics* 84.6, R989–R1001. DOI: 10.1190/geo2018-0884.1.

Ovcharenko, O., Kazei, V., Peter, D., and Alkhalifah, T. (2017). “Neural network based low-frequency data extrapolation”. In: *3rd SEG FWI workshop: What are we getting?* Society of Exploration Geophysicists.

Ovcharenko, O., Kazei, V., Peter, D., and Alkhalifah, T. (2018a). “Variance-based model interpolation for improved full-waveform inversion in the presence of salt bodies”. *Geophysics* 83.5, R541–R551. DOI: 10.1190/geo2017-0575.1.

Ovcharenko, O., Kazei, V., Peter, D., Silvestrov, I., Bakulin, A., and Alkhalifah, T. (2021). “Dual-band generative learning for low-frequency extrapolation in seismic land data”. In: *SEG Technical Program Expanded Abstracts.* Society of Exploration Geophysicists, pp. 1345–1349. DOI: 10.1190/segam2021-3579442.1.

Ovcharenko, O., Kazei, V., Peter, D., Zhang, X., and Alkhalifah, T. (2018b). “Low-frequency data extrapolation using a feed-forward ANN”. In: *80th EAGE Conference and Exhibition Extended Abstracts.* European Association of Geoscientists & Engineers. DOI: 10.3997/2214-4609.201801231.

Ovcharenko, O., Kazei, V., Plotnitskiy, P., Peter, D., Silvestrov, I., Bakulin, A., and Alkhalifah, T. (2020). “Extrapolating low-frequency prestack land data with deep learning”. In: *SEG Technical Program Expanded Abstracts.* Society of Exploration Geophysicists, pp. 1546–1550. DOI: 10.1190/segam2020-3427522.1.

Plessix, R. E. (2006). “A review of the adjoint-state method for computing the gradient of a functional with geophysical applications”. *Geophysical Journal International* 167.2, pp. 495–503. DOI: 10.1111/j.1365-246X.2006.02978.x.

Pollok, L., Saßnowski, M., Kühnlenz, T., Gundelach, V., Hammer, J., and Pritzkow, C. (2018). “Geological exploration and 3D model of the Asse salt structure for SE expansion of the Asse II mine”. In: *Mechanical Behavior of Salt IX (SaltMechIX)*, pp. 753–763.

Qian, F., Yin, M., Liu, X.-Y., Wang, Y.-J., Lu, C., and Hu, G.-M. (2018). “Unsupervised seismic facies analysis via deep convolutional autoencoders”. *Geophysics* 83.3, A39–A43. DOI: 10.1190/geo2017-0524.1.

Raissi, M., Perdikaris, P., and Karniadakis, G. E. (2019). “Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations”. *Journal of Computational Physics* 378, pp. 686–707. DOI: 10.1016/j.jcp.2018.10.045.

Richardson, A. (2018). “Seismic Full-Waveform Inversion Using Deep Learning Tools and Techniques”. *arXiv*. DOI: 10.48550/arXiv.1801.07232.

Ronneberger, O., Fischer, P., and Brox, T. (2015). “U-Net: Convolutional Networks for Biomedical Image Segmentation”. In: *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*. Ed. by N. Navab, J. Hornegger, W. M. Wells, and A. F. Frangi. Cham: Springer, pp. 234–241. DOI: 10.1007/978-3-319-24574-4_28.

Rumelhart, D. E. (1988). “The architecture of mind: A connectionist approach”. In: *Mind Readings: Introductory Selections on Cognitive Science*. Ed. by P. Thagard. MIT Press. Chap. 8, pp. 207–238.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). “Learning Representations by Back-Propagating Errors”. *Nature* 323, pp. 533–536. DOI: 10.1038/323533a0.

Șortan, S.-A. (2022). “Effects of seismic anisotropy and attenuation on first-arrival waveforms recorded at the Asse II nuclear repository”. MA thesis. Karlsruhe Institute of Technology (KIT). DOI: 10.5445/IR/1000151635.

Springenberg, J. T., Dosovitskiy, A., Brox, T., and Riedmiller, M. (2015). “Striving for simplicity: The all convolutional net”. In: *3rd International Conference on Learning Representations, ICLR 2015 – Workshop Track Proceedings*. DOI: 10.48550/arXiv.1412.6806.

Sun, B. and Alkhalifah, T. (2020). “ML-Misfit: Learning a Robust Misfit Function for Full-Waveform Inversion Using Machine Learning”. In: *82nd EAGE Annual Conference and Exhibition Extended Abstracts*. European Association of Geoscientists & Engineers. DOI: 10.3997/2214-4609.202010466.

Sun, H. and Demanet, L. (2018). “Low-frequency extrapolation with deep learning”. In: *SEG Technical Program Expanded Abstracts*. Society of Exploration Geophysicists, pp. 2011–2015. DOI: 10.1190/segam2018-2997928.1.

Sun, H. and Demanet, L. (2019). “Extrapolated full waveform inversion with convolutional neural networks”. In: *SEG Technical Program Expanded Abstracts*. Society of Exploration Geophysicists, pp. 4962–4966. DOI: 10.1190/segam2019-3197987.1.

Sun, H. and Demanet, L. (2020a). “Elastic full waveform inversion with extrapolated low-frequency data”. In: *SEG Technical Program Expanded Abstracts*. Society of Exploration Geophysicists, pp. 855–859. DOI: 10.1190/segam2020-3428087.1.

Sun, H. and Demanet, L. (2020b). “Extrapolated full-waveform inversion with deep learning”. *Geophysics* 85.3, R275–R288. DOI: 10.1190/geo2019-0195.1.

Sun, H. and Demanet, L. (2022). “Deep Learning for Low-Frequency Extrapolation of Multicomponent Data in Elastic FWI”. *IEEE Transactions on Geoscience and Remote Sensing* 60. DOI: 10.1109/TGRS.2021.3135790.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Dumitru, E., Vanhoucke, V., and Rabinovich, A. (2015). “Going Deeper with Convolutions”. In: *2015 IEEE*

Conference on Computer Vision and Pattern Recognition (CVPR). DOI: 10.1109/CVPR.2015.7298594.

Tarantola, A. (1984a). "Inversion of seismic reflection data in the acoustic approximation". *Geophysics* 49.8, pp. 1259–1266. DOI: 10.1190/1.1441754.

Tarantola, A. (1984b). "Linearized Inversion of Seismic Reflection Data". *Geophysical Prospecting* 32.6, pp. 998–1015. DOI: 10.1111/j.1365-2478.1984.tb00751.x.

Ten Kroode, F., Bergler, S., Corsten, C., de Maag, J. W., Strijbos, F., and Tijhof, H. (2013). "Broadband seismic data – The importance of low frequencies". *Geophysics* 78.2, WA3–WA14.

Thorbecke, J. (2021). *FDELMODC: 2D Finite-Difference Wavefield Modelling*. URL: https://github.com/JanThorbecke/OpenSource/blob/master/doc/fdelmodcManual.pdf (visited on 11/10/2022).

Van den Oord, A., Dieleman, S., Zen, H., Simonyan, K., Vinyals, O., Graves, A., Kalchbrenner, N., Senior, A., and Kavukcuoglu, K. (2016). "WaveNet: A Generative Model for Raw Audio". *arXiv*. DOI: 10.48550/arXiv.1609.03499.

Van Leeuwen, T. and Herrmann, F. J. (2013). "Mitigating local minima in full-waveform inversion by expanding the search space". *Geophysical Journal International* 195.1, pp. 661–667. DOI: 10.1093/gji/ggt258.

Virieux, J. and Operto, S. (2009). "An overview of full-waveform inversion in exploration geophysics". *Geophysics* 74.6. DOI: 10.1190/1.3238367.

Virieux, J. (1986). "P-SV wave propagation in heterogeneous media: Velocity-stress finite-difference method". *Geophysics* 51.4, pp. 889–901. DOI: 10.1190/1.1442147.

Virtanen, P., Gommers, R., et al. (2020). "SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python". *Nature Methods* 17, pp. 261–272. DOI: 10.1038/s41592-019-0686-2.

Wang, M., Xu, S., and Zhou, H. (2020). "Self-supervised learning for low frequency extension of seismic data". In: *SEG Technical Program Expanded Abstracts*. Society of Exploration Geophysicists, pp. 1501–1505. DOI: 10.1190/segam2020-3427086.1.

Wang, Z. and Bovik, A. C. (2009). "Mean Squared Error: Love it or Leave it? A new look at signal fidelity measures". *IEEE Signal Processing Magazine* 26.1, pp. 98–117. DOI: 10.1109/MSP.2008.930649.

Wang, Z., Bovik, A. C., Sheikh, H. R., and Simoncelli, E. P. (2004). "Image quality assessment: From error visibility to structural similarity". *IEEE Transactions on Image Processing* 13.4, pp. 600–612. DOI: 10.1109/TIP.2003.819861.

Warner, M. and Guasch, L. (2016). "Adaptive waveform inversion: Theory". *Geophysics* 81.6, R429–R445. DOI: 10.1190/GEO2015-0387.1.

Wehner, D., Landrø, M., and Amundsen, L. (2019). "On low frequencies emitted by air guns at very shallow depths – An experimental study". *Geophysics* 84.5, P61–P71. DOI: 10.1190/geo2018-0687.1.

Wu, R. S., Luo, J., and Wu, B. (2013). "Ultra-low-frequency information in seismic data and envelope inversion". In: *SEG Technical Program Expanded Abstracts*. Society of Exploration Geophysicists, pp. 3078–3082. DOI: 10.1190/segam2013-0825.1.

Xiao, X., Lian, S., Luo, Z., and Li, S. (2018). "Weighted Res-UNet for High-Quality Retina Vessel Segmentation". In: *9th International Conference on Information Technology in Medicine and Education, ITME 2018*, pp. 327–331. DOI: 10.1109/ITME.2018.00080.

Yang, F. and Ma, J. (2019). "Deep-learning inversion: A next-generation seismic velocity model building method". *Geophysics* 84.4, R583–R599. DOI: 10.1190/geo2018-0249.1.

Yu, S. and Ma, J. (2021). "Deep Learning for Geophysics: Current and Future Trends". *Reviews of Geophysics* 59.3, pp. 1–36. DOI: 10.1029/2021RG000742.

Zhang, Z.-D., Alkhalifah, T., Naeini, E. Z., and Sun, B. (2018). "Multiparameter elastic full waveform inversion with facies-based constraints". *Geophysical Journal International* 213.3, pp. 2112–2127. DOI: 10.1093/gji/ggy113.

Zhao, H., Gallo, O., Frosio, I., and Kautz, J. (2016). “Loss Functions for Image Restoration With Neural Networks”. *IEEE Transactions on Computational Imaging* 3.1, pp. 47–57. DOI: 10.1109/tci.2016.2644865.

Zhou, Y. T. and Chellappa, R. (1988). “Computation of optical flow using a neural network”. In: *1988 IEEE International Conference on Neural Networks*, pp. 71–78. DOI: 10.1109/icnn.1988.23914.

Zhu, W. and Beroza, G. C. (2019). “PhaseNet: A deep-neural-network-based seismic arrival-time picking method”. *Geophysical Journal International* 216.1, pp. 261–273. DOI: 10.1093/gji/ggy423.

Appendix

A. Used software and hardware

For this thesis I used the following software packages and hardware, which I would like to acknowledge here:

- Python packages: NumPy (Harris et al., 2020) and SciPy (Virtanen et al., 2020) for generating the subsurface models, TensorFlow (Abadi et al., 2015) for the machine learning framework and ObsPy (Beyreuther et al., 2010) for small data processing steps, Thomas Hertweck's seismics I/O package for reading in the .su files, and Jan Walda's framework for setting up and training encoder-decoder networks.
- Seismic Unix (Cohen and Stockwell, 2022) for data processing and quick quality controls.
- SOFI2D (Bohlen et al., 2016) for finite-difference forward modeling.
- Scientific color map "roma" (Crameri, 2021) for plots of the subsurface models.
- Four Nvidia A100 GPUs with 40 GB RAM each.

B. U-Net for the reconstruction tests

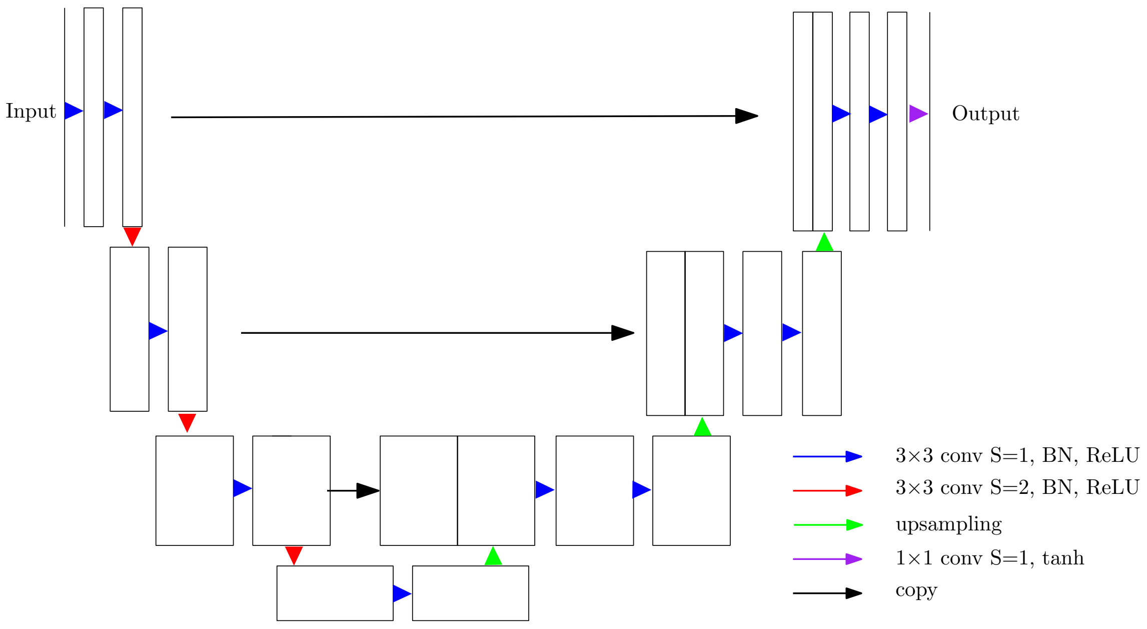
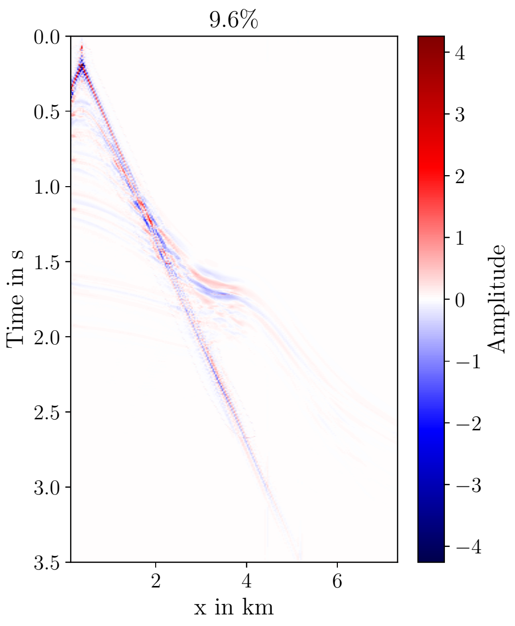
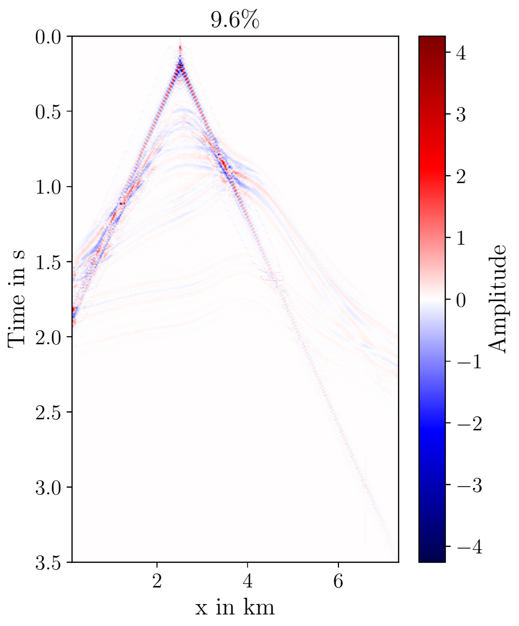


Figure B.1.: Architecture of the U-Net used for the reconstruction tests. Compared to the original U-Net by Ronneberger et al. (2015, s. Figure 2.6), the operations of down- and up-sampling have changed. Here, $S = 2$ stands for a convolution with stride 2. Batch normalization layers (BN) are used before the activation function. The black arrows depict the characteristic skip-connections.

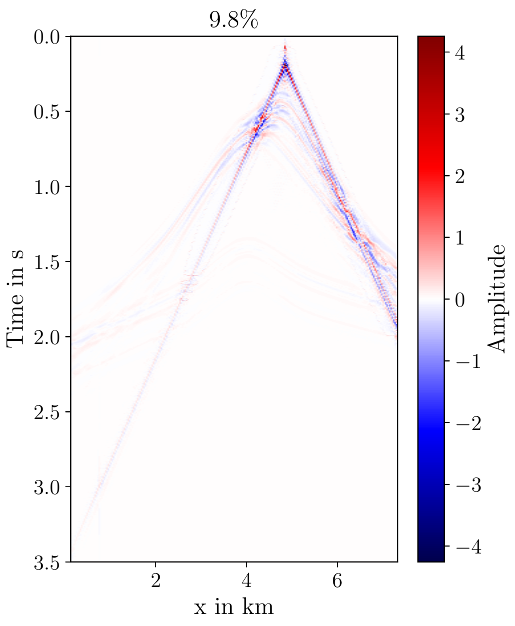
C. Further predictions of the final network



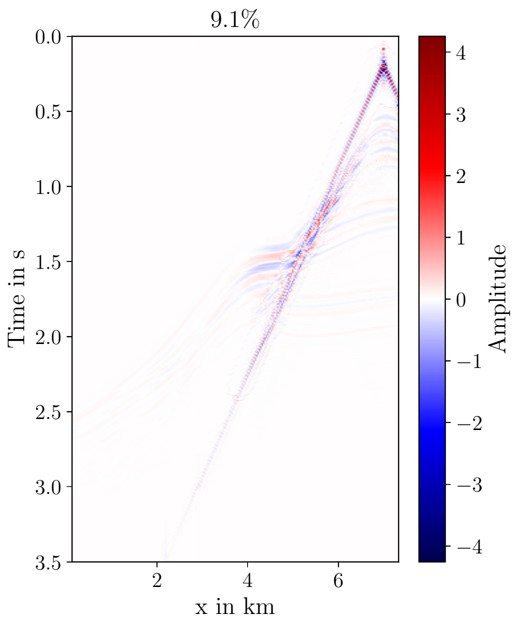
(a) Shot 2



(b) Shot 13

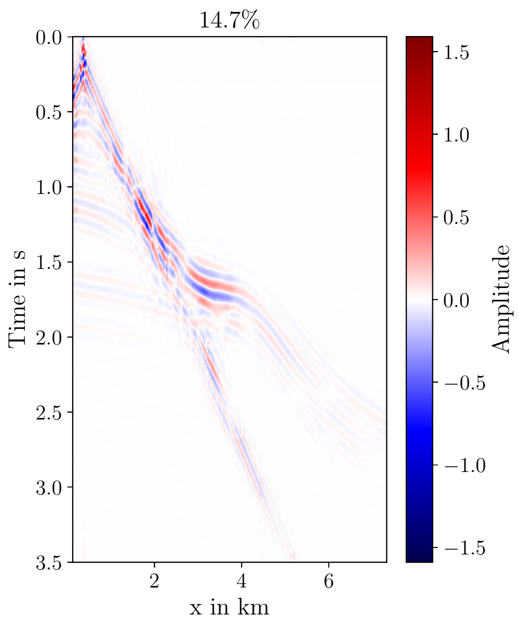


(c) Shot 25

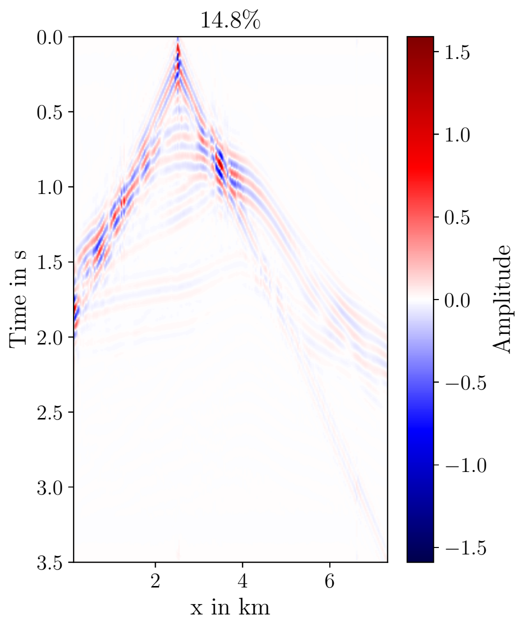


(d) Shot 36

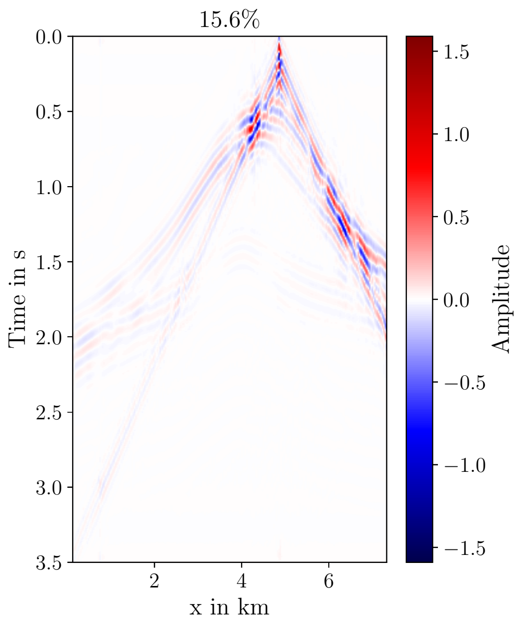
Figure C.2.: Residual plots between target and full-bandwidth prediction of (a) shot 2, (b) shot 13, (c) shot 25, and (d) shot 36. The title of the plots shows the NRMS value, which indicates a similar fit for all shot locations.



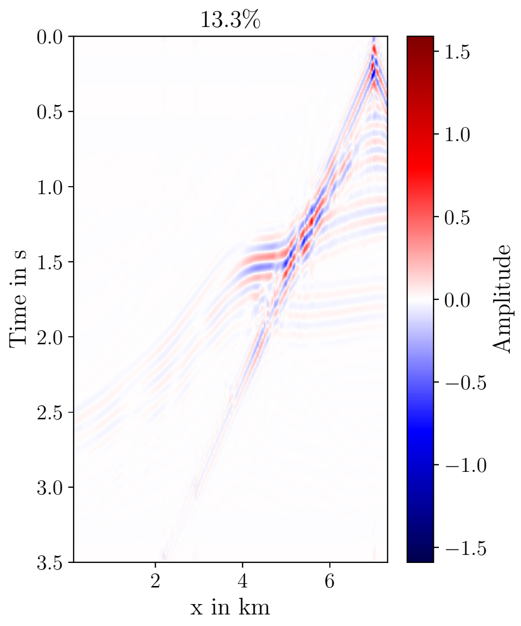
(a) Shot 2



(b) Shot 13

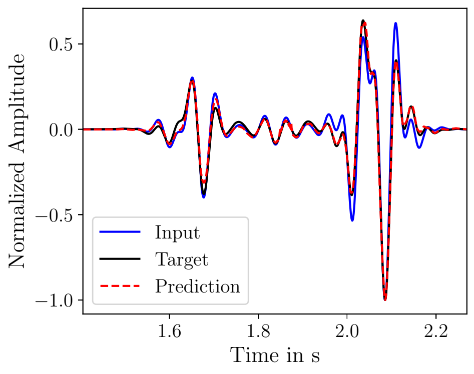


(c) Shot 25

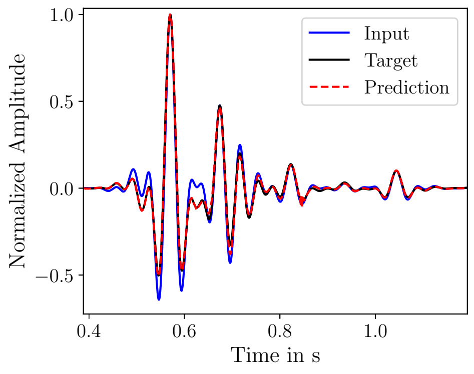


(d) Shot 36

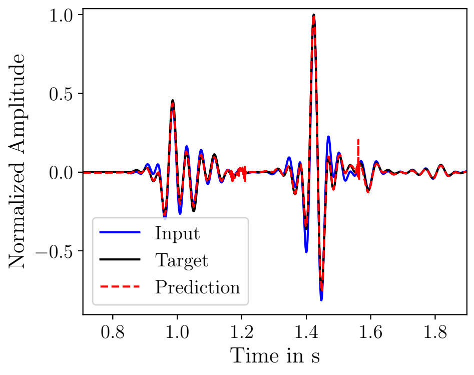
Figure C.3.: Residual plots between target and low-pass filtered full-bandwidth prediction of (a) shot 2, (b) shot 13, (c) shot 25, and (d) shot 36. The title of the plots shows the NRMS value, which indicates a similar fit for all shot locations.



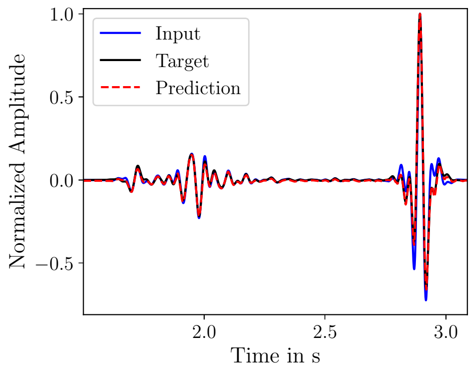
(a) Shot 2, Trace 300



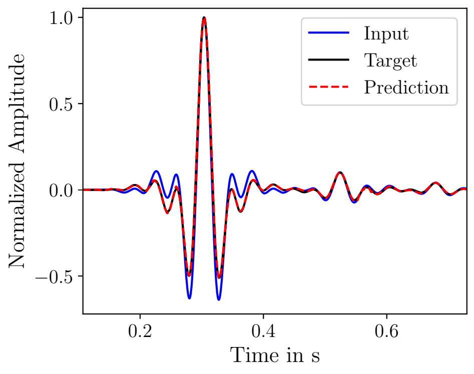
(b) Shot 13, Trace 300



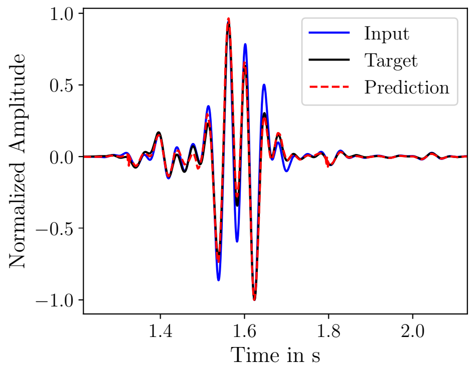
(c) Shot 25, Trace 300



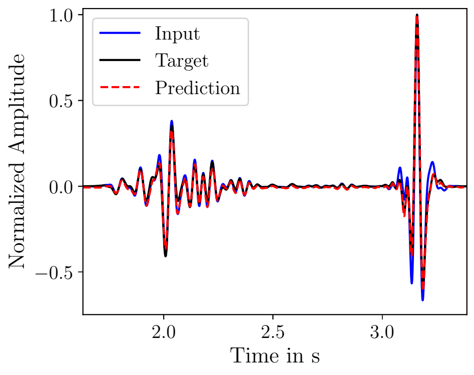
(d) Shot 36, Trace 300



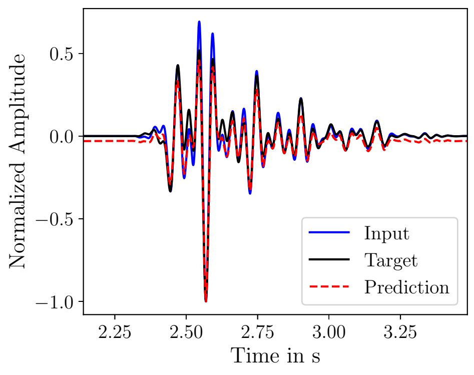
(e) Shot 2, Trace 40



(f) Shot 13, Trace 40

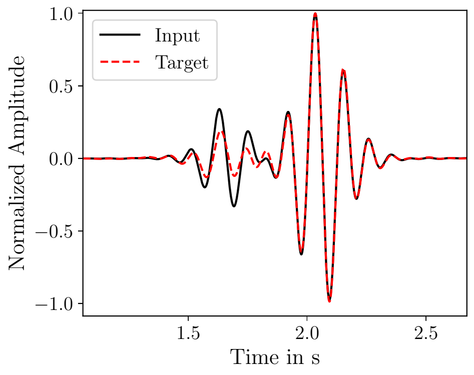


(g) Shot 25, Trace 40

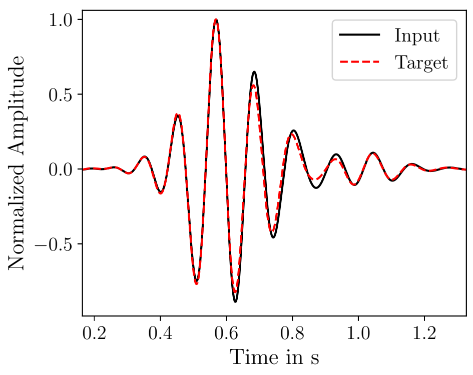


(h) Shot 36, Trace 40

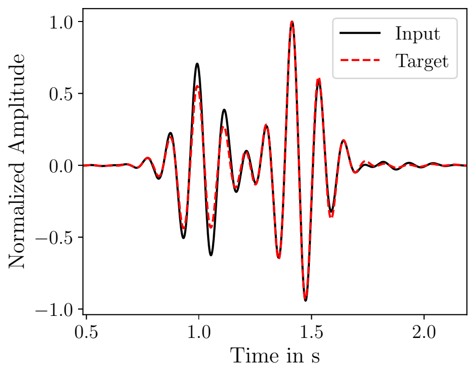
Figure C.4.: Waveform comparisons for the predicted full-bandwidth data for trace 300 (a-d) and 40 (e-h), for the shots 2 (a, e), 13 (b, f), 25 (c, g), and 36 (d, h). Note that in (a) the prediction follows partly the input, in (c) some high-frequency artifacts can be seen and in (h) the bias disturbs the good fit. The remaining traces show a nearly perfect fit.



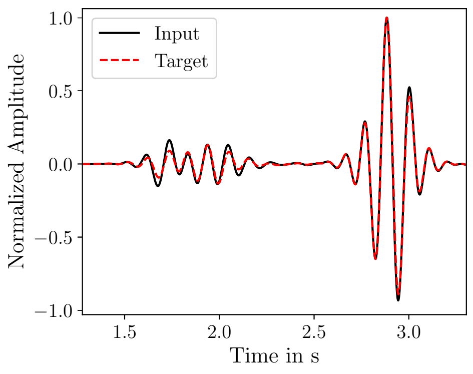
(a) Shot 2, Trace 300



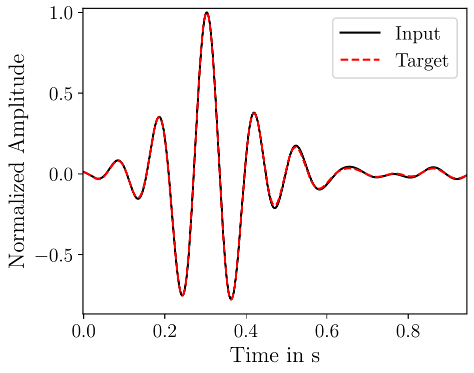
(b) Shot 13, Trace 300



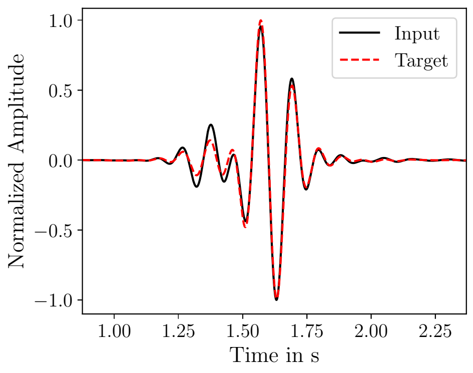
(c) Shot 25, Trace 300



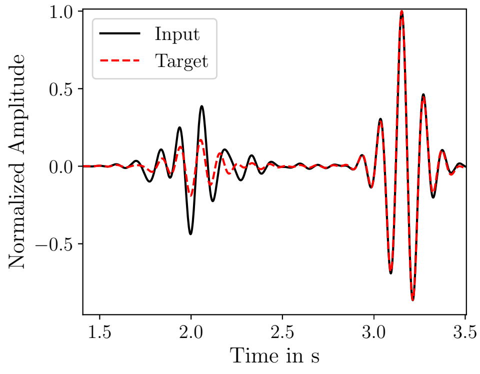
(d) Shot 36, Trace 300



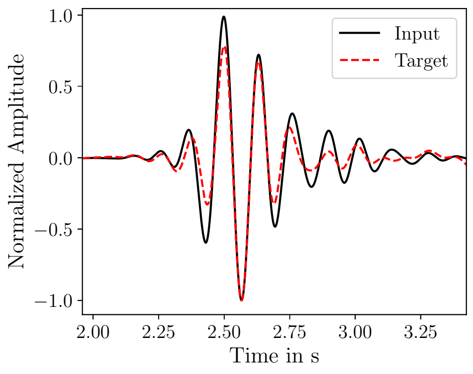
(e) Shot 2, Trace 40



(f) Shot 13, Trace 40



(g) Shot 25, Trace 40



(h) Shot 36, Trace 40

Figure C.5.: Waveform comparisons for the predicted full-bandwidth data for trace 300 (a-d) and 40 (e-h), for the shots 2 (a, e), 13 (b, f), 25 (c, g), and 36 (d, h). Note that in (a, f-h) the amplitudes are too small and phases are not entirely correct. The remaining traces show a nearly perfect fit.