Incorporating acquisition geometry in deep learning-based full waveform inversion

Introduction

Full waveform inversion (FWI) is widely considered the best method for reconstructing high-resolution velocity models. Conventional (physics-based) FWI methods need the following inputs: shot-gathers, wavelet, initial model, and acquisition geometry. These methods face several challenges such as: a high dependency on the initial model, the complexity of building the initial model, the possibility of cycle skipping, and a high computational burden (Schuster 2017).

The FWI problem lends itself well for a data driven approach using a deep learning graph. Deep learning methods do not require an initial model. Although, the run-time to train a deep model is comparable to running a physics-based algorithm, the application of a trained model is very fast (Yang and Ma, 2019). Moreover, trained models have repeat potential meaning that trained models can be re-applied on projects with similar datasets. This feature of deep learning models is a potential game-changer as it brings FWI into the realm of standard seismic processing solutions.

Against this background, many researchers have in the last decade worked on finding a deep learning solution for the FWI problem. The challenges these methods need to overcome include solutions for: the overfitting problem, optimization of hyper-parameters set selection, and designing the optimal architecture for the given problem (Aggarwal, 2018). Arayapol et al. (2018) used a deep neural network to estimate the velocity model using semblance-based velocity spectrum as input. Yang and Ma (2019) used a U-Net for Velocity Model Building (VMB) from noise-contaminated Common shot-gathers. They found that increasing noise levels improve a network’s generalization capability. Zhang and Lin (2020) experimented with different loss functions when training Generative Adversarial Networks (GANs) on a dataset named “CurvedData”. Their ‘VlocityGAN’ model performed better than other models applied to this dataset, which they attribute to a regularization term that is derived from the data. Li et al. (2019) introduced “SeisInvNet”, a model that generates one feature map for each trace. All feature maps in SeisInvNet are combined into the final model. They claim that “SeisInvNet” outperforms “InversionNet”, another well-known model that was introduced by Wu and Lin (2019). Although data-driven results are promising, in operational settings, there are presently, to our knowledge, only a few data-driven solutions tested on synthetic data (Zhang and Lin, 2020; Li et al., 2019) can compete with conventional FWI solutions.

A possible explanation is that the training data in many studies is oversimplified. For example, most state-of-the-art, data-driven FWI methods use shot gathers that are modelled with a flat surface (Li et al., 2019; Zhang and Lin, 2020; Yang and Ma, 2019). This is not very realistic when dealing with onshore data.

Deep Neural Networks (DNNs) are capable of learning complex relationships in datasets with high variability in input - and target space. In seismic modelling, we stochastically vary many input parameters while others remain static. Examples of parameters that are typically remain static are: the seismic wavelet, the acquisition geometry, and the topography. If these static assumptions are not valid, we need to take action. In this paper the flat surface assumption is not valid. Moreover, we also assume that the acquisition geometry varies. We model data with extreme topography and varying acquisition geometries and we present DNN solutions that take both effects into account. In addition, we compare networks that are trained on data with – and without a flat surface / fixed acquisition assumption.

Method

We use simulated data for training, testing and validation. Our velocity models are stochastically varied grids of 200 x 200 cells with a grid size of 10 m. Each model contains 10 to 50 events grouped in sequences that are structurally deformed to simulate folding and erosion. Each layer's velocity is allowed to vary between 1000 to 5500 m/s, and generally exhibits an increasing trend with depth. A velocity variance of 1500 m/s is allowed, and each layer can change laterally with a variation of up to 1000 m/s.
To synthesise seismic data, we design an acquisition geometry with 100 receivers fixed at the surface and spaced 20 m apart. We simulate 10 shot gathers per model by shooting from 10 regularly spaced positions into the fixed receiver spread. The upper left image in Fig. 1 shows the acquisition geometry along the surface of one model with extreme topography. Shot locations in this image are represented by red dots while the yellow dots represent receiver locations. Shot gathers are constructed by propagating a Ricker wavelet with a dominant frequency between 15 to 25 Hz through the model using an 8\textsuperscript{th} order finite difference numerical solution of the acoustic wave equation. Each trace is 3 s with a temporal sampling rate of 3 ms (1000 samples). We apply gain - and normalization functions. Operating in this way, we create 1000 observation points. In our DNN experiments we use 872 of these examples for training, 96 for testing and 32 for validation.

We train two networks with different architectures. The first network, hereinafter Network 1, includes the acquisition geometry data as additional input to the shot gathers (Fig. 1). The second network (Network 2), is similar to Network 1, except that is operates on shot gathers only. The first input layer in both networks is a ten-channel image input of size 1000*100. This is followed by, three convolutional blocks, each of which contains a convolutional layer with ten filters, a ReLU layer, and a max-pooling layer, which are applied to enhance traces with local data. This design is inspired by the work of Li et al. (2019). At this point the coordinates of shot and receivers are added in Network 1. This is done through a concatenation layer with one hot coding. In Network 2 we skip this layer. Next, the information is passed through another convolutional block with stride, and an encoder/decoder convolutional network with a depth of three. The number of filters in this part is 32, 64, and 128, respectively. Finally, we add a block with two transposed convolutional layers. The size of the feature maps and the output model are adjusted by cropping and by the transposed convolutional layers. Finally, we add a convolutional layer to transform the feature maps into the final result. We use MSE as loss function, and the ADAM method to update the weights. The hyper-parameters and loss function values of the training phase are given in Table 1.
Figure 1 Architecture and input/output data regarding to Network 1

Table 1 Hyper-parameters and final loss function values of the training phase

<table>
<thead>
<tr>
<th>Hyper-parameter</th>
<th>Mini Batch Size</th>
<th>Initial Learning Rate</th>
<th>Learn Rate Drop Period</th>
<th>Learn Rate Drop Factor</th>
<th>L2 Regularization</th>
<th>Number of Epochs</th>
<th>The final loss function for the Validation dataset (Net1)</th>
<th>The final loss function for the Validation dataset (Net2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value</td>
<td>32</td>
<td>5e-3</td>
<td>20</td>
<td>0.8</td>
<td>1e-3</td>
<td>400</td>
<td>3.1E+8</td>
<td>2.5E+9</td>
</tr>
</tbody>
</table>

Example

Figure 2 shows three results from our test data set. The true models are shown on the left. The shot point locations are indicated by white asterisks. Next to that we show the model predicted by Network 1 and next to that the result of Network 2. In the right-most column, we show a comparison of three velocity profiles at the location of the black line in the left-most column. We compare the true profile (blue) with the predicted profile by Network 1 (brown) and the predicted profile by Network 2 (yellow).

As expected, we observe that Network 2 is unable to reconstruct the velocity model for models with extreme topography (middle - and lowest rows). However, if there is no topography (upper row), Network 2 performs better than Network 1. A possible explanation is the following: approximately one-quarter out of all models have a flat surface. Apparently, Network 2 has learned to recognize flatness as a common feature resulting in excellent predictions when the surface is indeed flat. When the surface is not flat, the model fails to make reasonable predictions.

The final value of MSE metric for Network 1 is nearly the same for all three models. Also, final errors in the loss functions are nearly the same for all models. This implies that the performance of Network 1 is independent from the surface topography of the models. The network does quite a good job in
reconstructing velocity models with extreme topography but when the models are flat, the fit is visually less appealing. A possible explanation for this apparent mis-fit is the limited size of our training data set.

![Figure 2](image)

**Figure 2** Comparison of reconstructed models from different architecture with the original models from the test dataset

**Conclusion**

We presented a deep learning method for FWI in areas with varying acquisition geometry and topography. We train a DNN on two inputs: simulated shot gathers and acquisition geometry inputs. Our tests show that this new model is capable of predicting velocity models with extreme topographies.

**References**


Schuster, G.T. [2017] Seismic Inversion investigations in geophysics no. 20, SEG, Tulsa, USA.

