Early stage noise removal using a convolutional autoencoder

**Introduction**

Noise is a major concern in seismic data and influences the processing and interpretability of seismic data at various steps. Denoising is usually a time-demanding process, where several methods are chained in order to remove special cases of noise. However, noise has a certain pattern, apart from random noise, which can be exploited by machine learning algorithms, that rose drastically in popularity within the last decade. This is particularly the case for neural network. Convolutional neural network work well on images. Seismic data can be treated as such images. Rising computational computing power, in particular GPUs, as well as the occurrence of widely available open source libraries such as Tensorflow (Abadi et al., 2016) and PyTorch (Paszke et al., 2019) have made neural networks achievable for all kinds of institutions, companies and research fields.

Therefore, in recent years, several works have tried to address noise with neural networks. Richardson and Feller (2019) use a U-Net (Ronneberger et al., 2015) autoencoder with a ResNet34 (He et al., 2016) as pre-trained encoder. However, they train the neural network on blended synthetic datasets with the goal of deblocking and transfer it to denoising tasks. Mandelli et al. (2019) aim to interpolate and denoise data simultaneously using synthetic data for training, which has issues translating to field data and shows significant residuals. Zhao et al. (2019) remove swell noise using an elaborate process to generate accurate synthetic training data. A strength of neural networks is a fast inference, when they are trained. The necessity of retraining them on newly modeled synthetic data that resembles the currently processed dataset contradicts this advantage. Furthermore, a lot of a priori processing and interpretation is required to obtain the required labels, which makes such an approach not feasible at an early processing step.

Aforementioned supervised methods, among others, often require available geological information to construct accurate training data that represent the geology of the target data or involved process, which are not always available and is time-consuming to obtain. In this work, we aim to remove random noise in the shot-gather domain without the necessity of generating labels for a supervised learning approach. Instead, we use an unsupervised approach. Furthermore, we focus on preserving the primary signal, rather than removing as much noise as possible. In order to achieve that, we use an autoencoder that resembles the U-Net structure but uses a ResNext50 (Xie et al., 2017) encoder variant, that adopts some improvements from the XResNet approach (He et al., 2019).

**Autoencoder**

The ResNet (He et al., 2016) mitigates the vanishing gradient problem of deep neural networks via identity shortcut connections within a ResNet block. Figure 1a shows the ResNeXt variation of the ResNet block used in this work. The input of the first block has 256 channels. A copy of the input is added to the output of three consecutive convolutional layers, that use cardinality. These convolutional layers try to estimate residuals between the input and output of the block. Cardinality is similar to the concept of the Inception model (Szegedy et al., 2017) and used to reduce the need of a wider or deeper architecture. In the Inception model, the filter are depth-wise concatenated, while in the XResNet, the resulting features are added. In case the input and output are the same, the convolutional layers do not need to learn the identity function, due to the skipping connection, unlike in the U-Net, which allows to train deeper networks (He et al., 2016).

Figure 1b shows a schematic of the architecture used. The blue box indicates the input stem, which consists of three stacked convolutional layers and a maxpooling layer. The gray boxes indicate the stacks of ResNeXt blocks used, while the factor within the block describes the number of consecutive stacked ResNeXt blocks, before the next pooling is applied. The encoder is the ResNeXt50 (Xie et al., 2017) architecture, apart from the last fully connected layer, which we left out. The decoder is the exact same architecture flipped. The pooling is substituted by transposed convolutional layers to achieve upsampling.
Figure 1: Schematic description of the autoencoder used in this work. The amount of ResNeXt blocks used per depth is described by the number inside. The input of a ResNeXt block propagates through the stack of three convolutional layers with a cardinality of 32 and 256 input feature maps. The merged output of the convolutional layers is added to a copy of the input that skipped the convolutional process. Therefore, the convolutional layers learn the residuals, which gives the Residual network (ResNet) its name.

Training and Validation

We use a large dataset in comparison to the GPU memory available. We use a NVIDIA GeForce GTX 1080, which has a memory of 8 GB. Furthermore, we use a deep neural network, that has slightly more than 100 layers. In our case, it is not feasible to use a batch of complete shot gathers. Hence, we split a shot gather of 2501 samples and 576 traces into subimages of 128 samples by 128 traces. Therefore, we extend the shot gathers to a size of 2560 samples and 640 traces by zero padding. Afterwards, we split the shot gather into the aforementioned subimages. This process is repeated for each shotgather. Our approach does not only solve the memory issue but also helps us during the training. Neighboring shots are quite similar. Therefore, using a complete shot gather as a batch leads to less exploration of the loss function in the neural network. The loss function fluctuates less than using smaller batches, which differ from another from the simple fact, that they are from different offsets and traveltimes.

In this study, we use field data that contains 1077 shots for training with a maximum offset of 7338 m. We split the shots such that we have 270 shots for validation and 807 shots for training. Our loss function is the mean squared error between the raw data and the reconstructed data of the decoder. The encoder can be used with pre-trained weights that are publicly available. However, we choose to train the full network ourselves in order to investigate the training process in more detail and allow for changes in the architecture and the input shapes, particularly reduce their size to fit the GPU memory. Our training is unsupervised, meaning the goal of the autoencoder is to reproduce the input. We use the Adam optimizer (Kingma and Ba, 2014), a learning rate of $10^{-3}$ and train the network for 1000 epochs. As a loss function, we use the Huber loss (Huber, 1964). Additionally, we perturbate the convolutional layers with random noise in the order of $10^{-5}$ to $10^{-4}$ during the training process. This forces slight changes of the weights in order to explore more of the search space. The noise on the weights acts as a regularization and reduces overfitting, which is also addressed by batch normalization (Ioffe and Szegedy, 2015) after each convolutional layer. The activation function used is the common rectified linear unit (ReLU).

Results

We apply the neural network to a dataset acquired by TGS in the Levantine Basin in the eastern part of the Mediterranean Sea, offshore Israel. Figure 3 shows a shot gather that was not part of the training data. The first reflection is from the sea floor, followed by reflections from dipping horizons. The pronounced reflection near the bottom at roughly 2.8 s stems from the top of a salt body. The difference plot 2c...
shows some residual of the sea floor reflection, which the network could not sufficiently fit. In the area of interest below, we observe almost no primary residuals but steeply dipping noise, that is removed, among other noise.

Figure 5 shows a common-offset gather of a near offset. In this excerpt, some parts of the data were used during training, which means we have to be careful evaluating those results, as we cannot exclude overfitting. However, the results of the shot gather in Figure 3 suggests, that overfitting is not a major issue here. In the common-offset gather in Figure 5, we can observe a similar pattern as in the shot gather, namely residuals of the sea floor and removed noise below. Particularly in the noisiest part of the dataset, between 2.0 s and 2.5 s, substantial amounts of noise are removed.

Figure 3: Denoising result of a shot gather, that is not used in the training data. The raw shot is shown at the top, the reconstructed shot gather of the autoencoder in the middle and their difference at the bottom.

Figure 5: Denoising result of a common-offset gather for a near offset. The raw data is shown at the top, the reconstructed data of the autoencoder in the middle and their difference at the bottom. Apart from the sea floor reflection, most primaries are well reconstructed. Most noise is removed in a slump complex between 2.0 s and 2.5 s.
Conclusions

In this work, we used a state-of-the-art autoencoder to denoise seismic data in the shot gather domain at an early stage of the processing sequence. We apply the network unsupervised, which removes the necessity to model synthetic data or other appropriate labels. We do not rely on the usually time consuming task, that adds additional bias to the training. We chose an architecture, that is able to approximate the seismic data well, such that the signal is preserved. This is true for most of the data. However, the high amplitude sea floor shows removed primary signal. This could be addressed by a special consideration of outliers within the training process, e.g. in the loss function. Furthermore, we trained the network on parts of the shown dataset. For a better generalization, more divers datasets would be required.

Acknowledgements

The work was supported by the sponsors of the Wave Inversion Technology (WIT) consortium and the Federal Ministry for Economic Affairs and Energy of Germany (project number 03SX504B). We thank Google for providing the Tensorflow API. Furthermore we thank TGS for providing the field data. We would like to thank the Applied Seismics Group Hamburg for continuous discussions.

References


