De-noising Seismic Data using Deep Learning

Abstract:
In this paper, deep learning is introduced as a method to reduce the noise in seismic data. Denoising of seismic data enables the professionals to understand the condition of Earth's crust under study and to give recommendations regarding drilling, conservation, etc. The algorithms traditionally used for denoising depend on signal models and their corresponding prior assumptions. Deep learning models consists of multiple layers of neural networks. In this paper, we will be using supervised learning in which the output denoised data is available. Convolutional auto-encoder model is used to learn the transformations required to denoise the data. During training, the model tunes its parameters to fit the training data. At the end of training, the model can denoise raw seismic data without the need for parameter tuning and optimal modelling of signal and noise.

Methodology:
Deep learning is a machine learning technique in which multiple neural networks are used for classification, image recognition, prediction, computer vision, etc. A deep learning model requires an input which is processed through the network and produces an output. In supervised learning, the output is labelled during the training period, thus the model tweaks the parameters to fit to the labelled training data. A convolutional neural network is a deep learning model used primarily for computer vision as it produces good results with images as input. An autoencoder model is the one in which the input layer and the output layer are of the same dimensions whereas the layers in the middle have lower dimensions. Thus, convolutional auto-encoder model is used because the denoised data will have the same dimensions as the raw data.

The raw seismic had the dimensions (351, 8910) in 2D form. After 3D transformations, the dimensions became (351,100,81). Deep learning models work well with large amounts of data. Thus, we break down the 3D cube into many smaller cubes to be used as input for the model. As the raw data cube does not have same dimensions in X, Y and Z axis, the smaller cubes can not have the same dimensions along the axes. (3,5,3), (9,10,9) and (27,22,27) are three possible input cube sizes. Slicing the input cubes along these dimensions do not disrupt the integrity of the raw seismic data as it cleaves them according to their present dimensions.As the dimension reduce, the number of input cubes increase and vice versa. For the input cube of the dimensions (3,5,3), 69497 input cubes are made. For the input cube of the dimensions (9,10,9), 3860 input cubes are made. For the input cube of the dimensions (27,22,27), 194 input cubes are made.

For (3,5,3) input cubes, 60000 cubes are in training set and 9497 cubes are in test set. For (9,10,9) input cubes, 3000 cubes are in training set and 860 cubes are in test set. For (27,22,27) input cubes, 130 cubes are in training set and 65 cubes are in test set. The smaller dataset for (27,22,27) input cube is compensated by the larger size of the individual input cube. The denoised data is of the dimensions (351, 8910) as well and is divisible into three possible cube

sizes of (3,5,3), (9,10,9) and (27,22,27). In this paper, supervised learning is used thus the denoised data is treated as labeled data. During training, the convolutional auto-encoder changes its parameters to fit the raw seismic input data to the denoised output data. A 48 core CPU was used for training purposes.

Means squared loss is used to determine how the model performs on test data. A very low loss shows that the model is working properly and denoising the test data significantly. There are multiple hyperparameters that can be tweaked to increase the performance of the model. Number of layers, filter size, training size, test size, validation set, epochs, and batch size are the hyperparameters that were tweaked to change the performance of the model. Number of layers increases the number of parameters in the model. If the model is too complex (high number of layers) then the model overfits to the data. The model also overfits to the if the training set is too small. One iteration of the model over training set is called one epoch. With increasing number of epochs, the model fits more and more to the training set. Batch normalization is used to reduce overfitting of the model on training data.

Results:

For simplicity purposes, this paper will discuss the results of only (27,22,27) sized cubes. Considering the nature of data, a denoised larger sized cube will provide more accurate results. The following tables contain the results for multiple convolutional auto-encoders.

- **Model 1**

Cube Size	27,22,27
Training Set	130
Test Set	65
Number of layers	4
Filter Size	3
Epochs	20
Batch Normalization	Yes
Batch Size	1
Number of trainable Parameters	4285
Loss Function	Means Square Loss
Training Loss	0.0226

Test Loss	0.0230

- **Model 2**

Cube Size	27,22,27
Training Set	130
Test Set	65
Number of layers	6
Filter Size	3
Epochs	5
Batch Normalization	Yes
Batch Size	1
Number of trainable Parameters	16,039
Loss Function	Means Squared Loss
Training Loss	0.0224
Test Loss	0.027

- **Model 3**

Cube Size	27,22,27
Training Set	130
Test Set	65
Number of layers	6
Filter Size	3
Epochs	30
Batch Normalization	Yes
Batch Size	1

Number of trainable Parameters	16,039
Loss Function	Means Squared Loss
Training Loss	0.0218
Test Loss	0.026

- **Model 4**

Cube Size	27,22,27
Training Set	130
Test Set	65
Number of layers	4
Filter Size	5
Epochs	10
Batch Normalization	Yes
Batch Size	1
Number of trainable Parameters	19,573
Loss Function	Means Squared Loss
Training Loss	0.0265
Test Loss	0.02

- **Model 5**

Cube Size	27,22,27
Training Set	130
Test Set	65
Number of layers	8
Filter Size	3

Epochs	10
Batch Normalization	Yes
Batch Size	1
Number of trainable Parameters	33,400
Loss Function	Means Squared Loss
Training Loss	0.0232
Test Loss	0.0287

- **Model 6**

Cube Size	27,22,27
Training Set	130
Test Set	65
Number of layers	6
Filter Size	5
Epochs	10
Batch Normalization	Yes
Batch Size	1
Number of trainable Parameters	73,663
Loss Function	Means Squared Loss
Training Loss	0.0268
Test Loss	0.0227

- **Model 7**

Cube Size	27,22,27
Training Set	130

Test Set	65
Number of layers	65
Filter Size	8
Epochs	5
Batch Normalization	Yes
Batch Size	1
Number of trainable Parameters	153,646
Loss Function	Means Squared Loss
Training Loss	0.0231
Test Loss	0.025

- **Model 8**

Cube Size	27,22,27
Training Set	130
Test Set	65
Number of layers	8
Filter Size	5
Epochs	25
Batch Normalization	Yes
Batch Size	1
Number of Trainable Parameters	153,646
Loss Function	Means Squared Loss
Training Loss	0.02
Test Loss	0.028

All the models mentioned above show a training and test loss of **~2%**. The low test loss shows that the models are not overfitting the training data. The low training loss shows that all the models are complex enough to fit to the data.

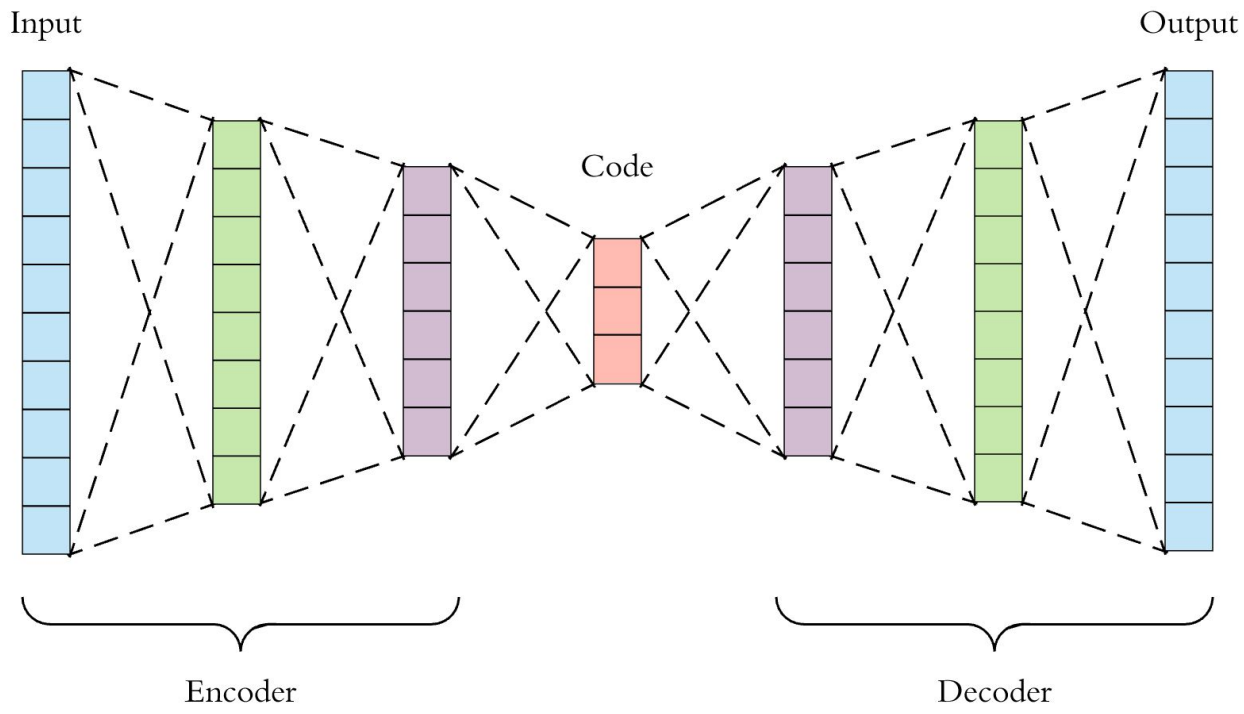
Conclusion:

In this paper, we have discussed multiple convolutional auto-encoder models which denoise raw seismic data. All the discussed models show significant denoising in the results. The limitations of this paper are that these models are only tested on a single dataset and the dataset is small. The models will become more robust and generalizable as they are trained on larger training datasets.

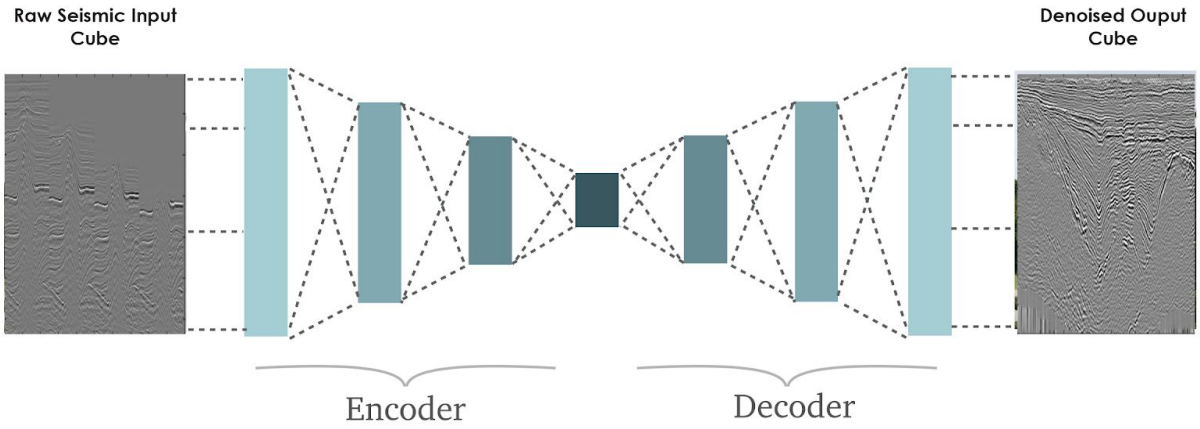
This is only the first step in automated seismic interpretation. After denoising, deep learning models can achieve human level interpretation if the models are trained on large labelled datasets.

Appendix:

- Basic structure of an auto-encoder



- Convolutional Autoencoder


Raw Seismic Input Cube
Denoised Ouput Cube
Encoder
Decoder