Convolutional Autoencoders: Application in denoising of geophysical data

Deepak K. Gupta

January 1, 2019

1 Definition

1.1 Project Overview

The domain of geophysics deals with data acquired through several physics-based experiments. Several teams of geophysicists work together to clean and process the acquired data before it anything can be interpreted out of it. The traditional processing steps are extremely cumbersome, and among these, removal of noise is a very challenging task. The geological complexities inside the earth make the data so heterogeneous that it is extemely difficult to differentiate the noise from the signal component of the data. Moreover, denoising is a cumbersome task, and with traditional techniques, a significant amount of time needs to be spent on it.

The application of various machine learning techniques has been explored to automate the process of seismic data denoising. A direction of research along this line is to investigate the application of autoencoders on such problems. Autoencoders are a variant of the traditional machine learning methodology, where the input and output of the network are set to be the same [1]. During training, the network is thus forced to learn an internal representation of the input data in a different space. Recently, application of autoencoders has also been explored on geophysical data [8, 10].

In a recent work, the application of deep fully-connected autoencoders has been explored for the removal of noise and reconstruction of signal in geophysical data [2]. Remarkable reduction in noise could be achieved, which makes autoencoders a potential denoising tool. Although the reduction in noise is significant, there exist further possibilities of improvements. Moreover, due to the computational costs associated with the fully-connected layers, they are not so favorable for large datasets.

In the past two years, convolutional neural networks (CNN), a variant of the traditional neural networks, have proved to be enormously powerful tool for classification problems. During this course, the discussion on CNN has provoked my interest to explore its application for the purpose of denoising geophysical data. Some recent applications of CNN on geophysical problems are identification of geological features on seismic data [11], automated velocity model building [9], among others.

Convolutional autoencoders, a form of CNNs can be used for denoising. Convolutional autoencoders could possibly help to overcome the drawbacks of traditional autoencoders in the context of denoising. In particular, it is believed that the spatial pattern of the signal can be preserved, and noise could be removed at significantly lower computational costs. Thus, in this project, the application of convolutional autoencoders will be explored for removal of noise from synthetic and real geophysical images. The synthetic datasets can be generated using the physics-based forward model of the problem. In this project, this aspect will not be explored, rather existing datasets will be used. Similarly, we will use a real dataset which is available in public. To compare our results, the performance baselines will be set using wavelet filtering, bilateral filtering and variational filtering techniques.

1.2 Problem statement

The main objective of this project is to explore the application of convolutional autoencoders for the reduction of noise in geophysical images. The focus of this project will be to explore the reduction of incoherent noise in particular. The test problems will include synthetic images containing random, normal and salt-pepper noise as well as real geophysical datasets with added random noise.

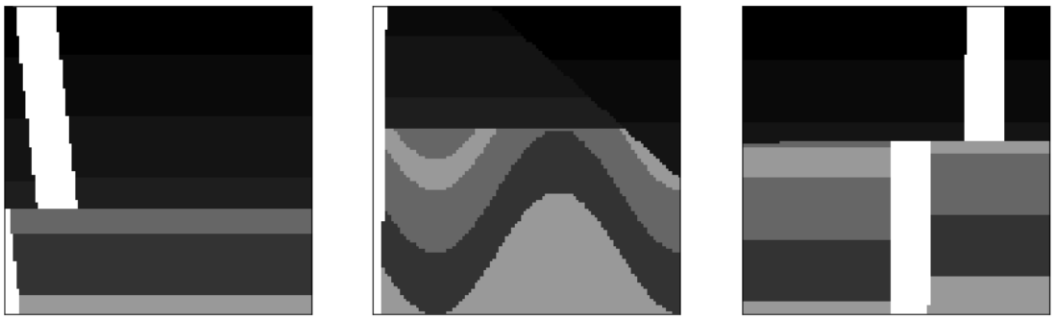

Figure 1: Synthetic geophysical images (with no added noise) comprising multiple geological features on a resolution of 100×100 pixels. These images have been adapted from those published at [5].

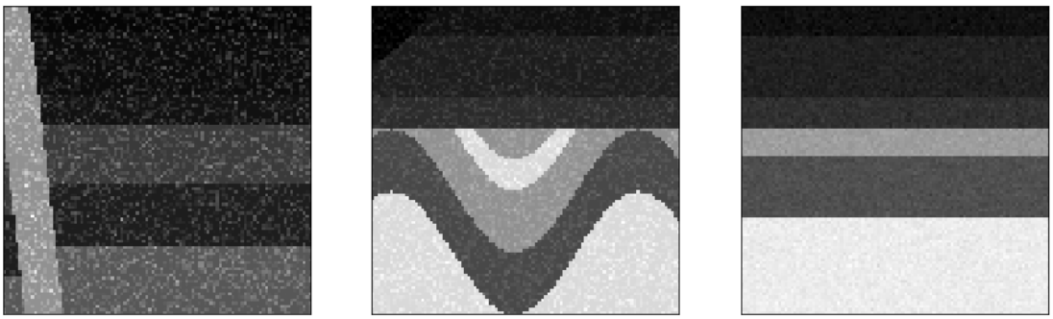

Figure 2: Synthetic geophysical images (with added normal noise) comprising multiple geological features on a resolution of 100×100 pixels. These images have been adapted by adding normal noise to the images available at [5].

1.3 Metrics

The problems to be studied here can be considered as regression problems in the sense that a good projection from noisy domain to noisefree domain needs to be learned. Thus, root mean square evaluation metric would suffice. In particular, root mean squared error (rmse) and mean absolute error (mae) are used as evaluation metrics.

2 Analysis

2.1 Data Exploration and Visualization

Th trained convolutional autoencoder needs to be able to remove incoherent noise from images, and for the task described in this work, the test images will comprise synthetic and real geophysical images. One possible choice could be to use the existing trained convolutional networks and transform them into autoencoders. However, most of the existing well established CNN models have been trained for classification problems and their training images differ from the images that are being dealt in this work. Moreover, for the purposes of learning, it has been decided to train a convolutional autoencoder. Due to limited compute capacity, the trained model is not expected be highly accurate.

To train the convolutional autoencoder, 2D synthetic images of 50×50 and 100×100 pixels are used. In this work, we use the set of synthetic images presented in [5] and studied in [4]. The resolution of original images was 200×200 pixels and the results reported in [4] have been obtained by training on K80 GPUs for a period of approximately 24 hours. However, since such resources are not available, the resolution of the training images has been reduced to 2 times and 4 times. The models will be trained on K1100 Tesla GPUs.

Fig. 1 shows three synthetic geophysical images comprising multiple geological features obtained on an image resolution of 100×100 pixels. These images have been adapted by coarsening the images available at

[5]. In all the three images, it can be seen that there are sharp contrasts in the images. These represent change in lithological properties inside the earth. In general, with the use of gradient-based optimization techniques, obtaining such contrasts requires very high resolution input images as well as deep neural networks. Since the compute capacity used for this work is very limited, low resolution outputs can be expected.

To train the autoencoder to denoise an image, it needs to be trained over noisy images as well. To accommodate this, noisy versions of the images are generated by adding random uniform/Gaussian or other types of noise. For example, Fig. 2 shows three images where uniform random noise of varying magnitudes have been added. The goal of a trained denoising autoencoder would be to remove the random noise from the images.

2.2 Algorithms and techniques

For denoising propose, *Convolutional Neural Networks (CNN)* are used. CNN is among the state of art algorithms used primarily for classification problems in machine learning. Compared to other training approaches, it needs larger sets of training data. However, in this study, the limitation is not caused by the amount of training data, rather the compute capacity itself forms the bottleneck. CNN is implemented in a autoencoder architecture, calling it convolutional autoencoder. Details related to the employed network architectures will be discussed in a later section.

2.3 Benchmark

The novelty associated with the applicability of a method cannot be justified if its performance is not compared to some of the state-of-art methods. In this case, the performance of convolutional autoencoders in the context of denoising needs to be compared to the existing denoising methods used in the field of geophysics. The results in this study are compared with those obtained from total variational filtering, bilateral filtering and wavelet filtering. The motivation to use these methods comes from the study presented in [5], where these methods have been used for benchmarking associated to a similar problem of geophysical denoising.

In this study, the details related to the benchmark methods are ignored. To implement these methods, the inbuilt functions within the `skimage` package under `skimage.restoration` have been used. The functions for total variational filtering, bilateral filtering and wavelet filtering are `denoise_tv_chambolle()`, `denoise_bilateral()` and `denoise_wavelet()`, respectively. For most of the parameters to be supplied to these functions, approximately optimal values have been chosen based on iterative tests.

3 Methodology

3.1 Data Preprocessing

For training the convolutional autoencoders, we use 2D synthetic and real images. As stated earlier, the base images for the synthetic cases have been taken from [4]. Preprocessing involved adding random uniform and normal noise to create noisy versions of these images. No additional preprocessing was involved in this study.

3.2 Implementation

The implementation of convolutional autoencoders for the task of denoising can be described in two steps:

1. Training the autoencoder for denoising
2. Application on noisy test data

3.2.1 Training the convolutional autoencoder

In this section, the training of the autoencoder is discussed in brief. Note that this study does not attempt to find an optimal choice of autoencoder for denoising. Rather, only one network configuration for each image

Layer (type)	Output Shape	Param #
input_1 (InputLayer)	(None, 50, 50, 1)	0
conv2d_1 (Conv2D)	(None, 50, 50, 16)	160
max_pooling2d_1 (MaxPooling2	(None, 25, 25, 16)	0
conv2d_2 (Conv2D)	(None, 25, 25, 8)	1160
max_pooling2d_2 (MaxPooling2	(None, 13, 13, 8)	0
conv2d_3 (Conv2D)	(None, 13, 13, 8)	584
max_pooling2d_3 (MaxPooling2	(None, 7, 7, 8)	0
conv2d_4 (Conv2D)	(None, 7, 7, 8)	584
up_sampling2d_1 (UpSampling2	(None, 14, 14, 8)	0
conv2d_5 (Conv2D)	(None, 14, 14, 8)	584
up_sampling2d_2 (UpSampling2	(None, 28, 28, 8)	0
conv2d_6 (Conv2D)	(None, 26, 26, 16)	1168
up_sampling2d_3 (UpSampling2	(None, 52, 52, 16)	0
conv2d_7 (Conv2D)	(None, 50, 50, 1)	145

Total params: 4,385
Trainable params: 4,385
Non-trainable params: 0

Figure 3: Representation of a convolutional autoencoder model for the images of size 50×50 pixels, built using Keras. The encoder part of the network downsamples the image to 7×7 pixels and 8 channels, and the decoder upsamples it further to the original size and a single channel.

Layer (type)	Output Shape	Param #
input_1 (InputLayer)	(None, 50, 50, 1)	0
conv2d_1 (Conv2D)	(None, 50, 50, 16)	160
max_pooling2d_1 (MaxPooling2	(None, 25, 25, 16)	0
conv2d_2 (Conv2D)	(None, 25, 25, 8)	1160
max_pooling2d_2 (MaxPooling2	(None, 13, 13, 8)	0
conv2d_3 (Conv2D)	(None, 13, 13, 8)	584
max_pooling2d_3 (MaxPooling2	(None, 7, 7, 8)	0
conv2d_4 (Conv2D)	(None, 7, 7, 8)	584
up_sampling2d_1 (UpSampling2	(None, 14, 14, 8)	0
conv2d_5 (Conv2D)	(None, 14, 14, 8)	584
up_sampling2d_2 (UpSampling2	(None, 28, 28, 8)	0
conv2d_6 (Conv2D)	(None, 26, 26, 16)	1168
up_sampling2d_3 (UpSampling2	(None, 52, 52, 16)	0
conv2d_7 (Conv2D)	(None, 50, 50, 1)	145

Total params: 4,385
Trainable params: 4,385
Non-trainable params: 0

Figure 4: Representation of a convolutional autoencoder model for the images of size 100×100 pixels, built using Keras. The encoder part of the network downsamples the image to 7×7 pixels and 8 channels, and the decoder upsamples it further to the original size and a single channel.

Table 1: Optimization related details for the various convolutional autoencoder models used in this study.

CNN model	Image size	Training samples	No. of noisy samples	Batch size	Epochs
ca_100_1_1	100×100	17920	4480	200	100
ca_50_1_3	50×50	35840	8960	200	100
ca_50_1_5	50×50	44800	11200	200	100
ca_seismic_1	50×50	211442	52861	400	120

Table 2: Performance comparison of convolutional autoencoder models with various filtering techniques for the removal of noise from various input images.

Case/Method	-	Input		Conv. Autoenc.		Tot. Var. Filt.		Bilat. Filt.		Wavelet Filt.	
	Sample	rmse	mae	rmse	mae	rmse	mae	rmse	mae	rmse	mae
ca_100_1_1	82	0.176	0.145	0.034	0.022	0.148	0.145	0.149	0.142	0.148	0.145
	1000	0.113	0.074	0.044	0.020	0.078	0.074	0.078	0.072	0.079	0.066
	1424	0.093	0.067	0.029	0.017	0.068	0.067	0.069	0.065	0.071	0.067
ca_50_1_3	320	0.349	0.286	0.078	0.039	0.297	0.286	0.296	0.277	0.290	0.264
	968	0.066	0.040	0.057	0.028	0.048	0.042	0.046	0.040	0.052	0.037
	1200	0.216	0.186	0.065	0.030	0.189	0.186	0.187	0.181	0.184	0.163
ca_50_1_5	2267	0.158	0.138	0.051	0.022	0.142	0.138	0.140	0.135	0.135	0.123
	820	0.125	0.091	0.04	0.012	0.084	0.051	0.083	0.058	0.093	0.051
	2000	0.147	0.127	0.056	0.03	0.132	0.128	0.130	0.124	0.125	0.108

resolution (50×50 and 100×100) has been used. These networks can be seen in Figs. 3 and 4. Table 1 lists details related to training various convolutional autoencoder models used in this study. For all the synthetic cases used in this study, a batch-size of 200 samples is used and the optimization is run for a total of 100 epochs. For the real test case, batch-size of 400 is used and the optimization problem is run for 120 epochs. The mean squared error function is used to compute the loss, and the values of other parameters are kept to the default values specified in Keras.

As stated earlier, K1100 Tesla GPU card is used and the models are trained using Keras and Tensorflow.

3.2.2 Application on noisy test data

Once the autoencoder models had been trained, the goal was to use them for the purpose of denoising and assess their performance with respect to the chosen benchmarks. The test data generally comprised a series of images with random noise. The results are presented below in the results section.

4 Results

4.1 Synthetic examples

For the various models developed in the study, almost similar CNN architectures were used. This is described in Figs. 3 and 4. For the first test, the model was trained to accept input images of size 100×100 pixels, and the resultant trained model is referred to as `ca_100_1_1` in Tables 1 and 2. For this model, a total of 17920 images, comprising noisefree as well as noisy images, was used for training purpose. Half of the total images used were noisefree and half of them contained Gaussian normal noise. During optimization, batch size of 200 images was used and the optimization problem was run for 100 epochs. Further the model was tested on a total of 4480 images.

Among the various test images, 3 were randomly chosen. The performance related details for these 3 test images are shown in Table 2. Further, the resultant images obtained from the model as well as with the baseline techniques are shown in Figs. 5, 6 and 7. From the root mean squared error (rmse) and

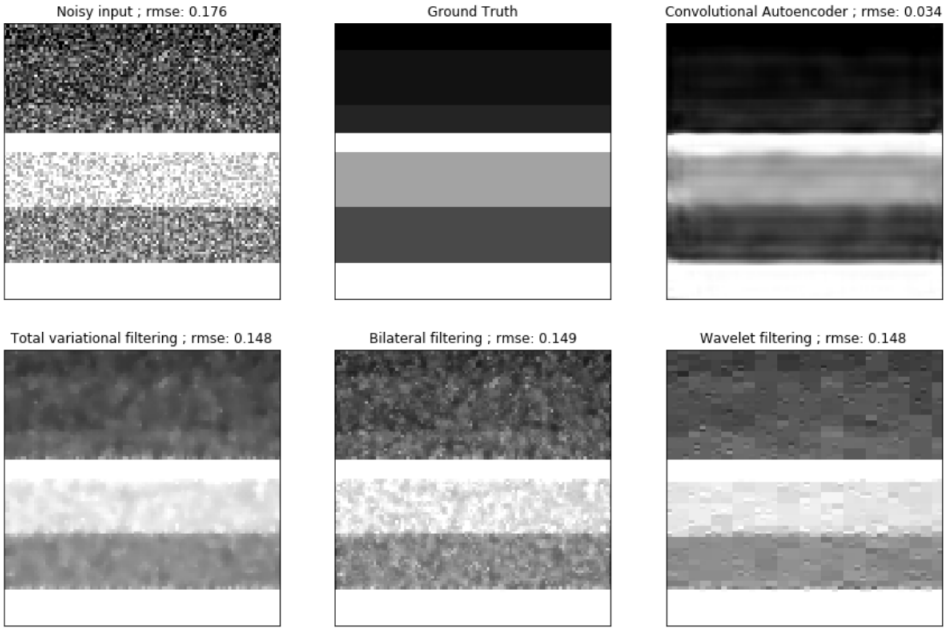

Figure 5: Noisy test image, the ground truth version, and the denoised results obtained using convolutional autoencoder, total variational filtering, bilateral filtering and wavelet filtering for test sample 82. The resolution of these images is 100×100.pixels and the autoencoder model used here is `ca_100_1_1`.

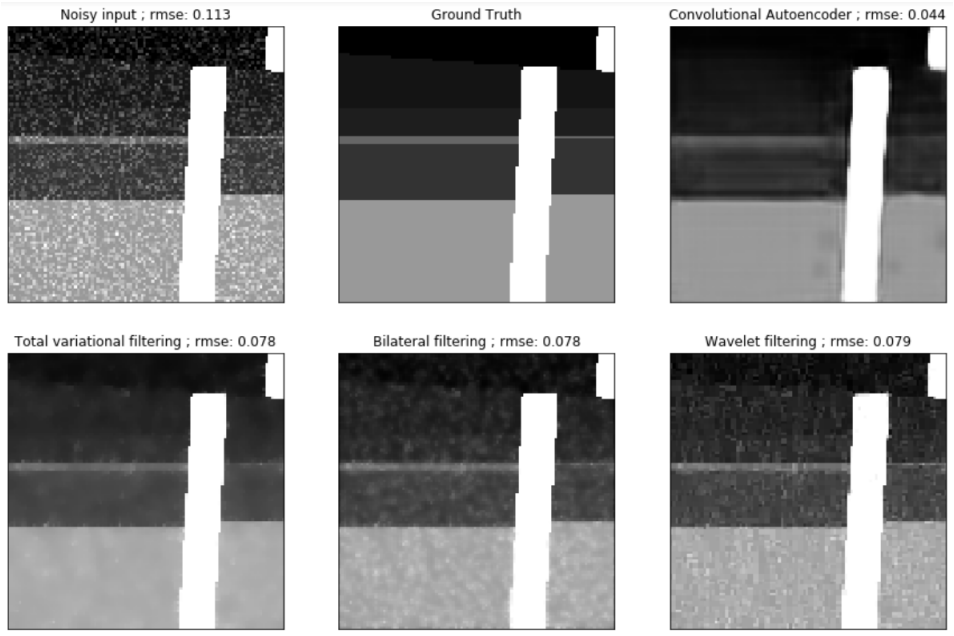

Figure 6: Noisy test image, the ground truth version, and the denoised results obtained using convolutional autoencoder, total variational filtering, bilateral filtering and wavelet filtering for test sample 1000. The resolution of these images is 100×100 pixels and the autoencoder model used here is `ca_100_1_1`.

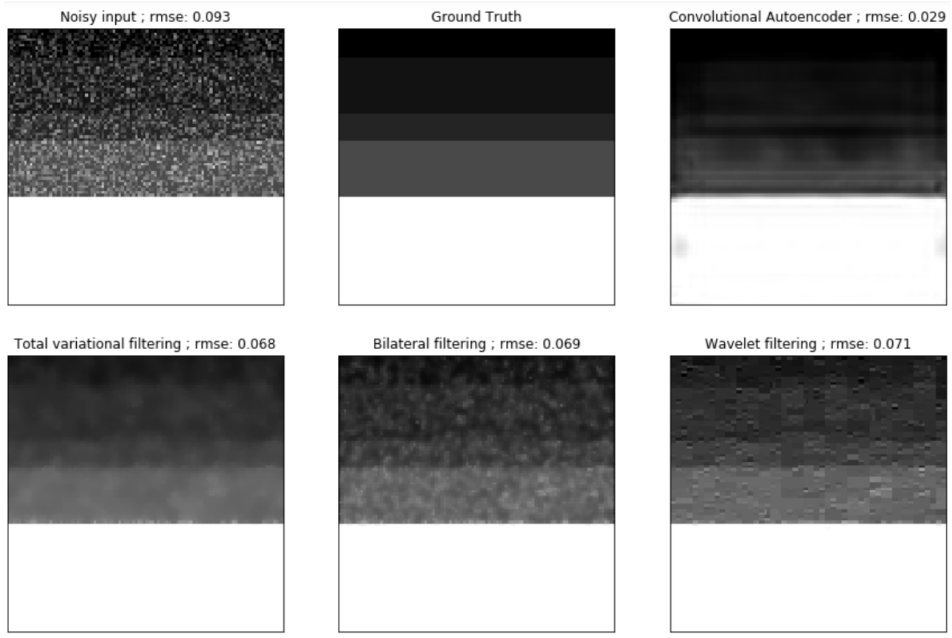

Figure 7: Noisy test image, the ground truth version, and the denoised results obtained using convolutional autoencoder, total variational filtering, bilateral filtering and wavelet filtering for test sample 1424. The resolution of these images is 100×100 pixels and the autoencoder model used here is `ca_100_1_1`.

mean absolute error (mae) values, it can be seen that the autoencoder models when trained for denoising, outperform the filtering based denoising methods. While at certain occasions, the filtering methods regularize the image without any significant error reduction, autoencoder model reduces the error almost for all the cases, as is also seen in the 3 randomly chosen samples.

However, a limitation is observed for a few samples, and this has been shown in Sample 1424 (Fig. 7). Several undesired artefacts arise in the denoised image which do not exist in the actual noisefree version. Moreover, although the noise on average is reduced, the relevant geophysical boundaries seem to be less prominent in the autoencoder results, when compared with those obtained using the filtering methods. Clearly, the autoencoder model has learned to reduce the error at a global level, however, it still needs further training to under the more localized representations.

Thus, as motivated in the earlier paragraph, the autoencoder model needs needs more data for training. However, since limited compute capacity was available, it was decided to reduce the resolution of each and increase the number of noisy samples used in training. Along this direction, further experiments used images of resolution 50×50 pixels. Two different instances of 50×50 size images have been considered. For both the cases, batch size was set to 200 and 100 epochs were run.

For the first case, a total of 35840 images were used for training, and out of this, 75% of the images comprised added random normal noise. Further the trained model was used on 8960 test images. For visualization, 3 test samples were randomly chosen and their performance was studied. The details related to the network as well as the model performance are mentioned in Tables 1 and 2. Figs. 11, 12 and 13 show the noisy input, ground truth, the results obtained using various filtering techniques and the results of the convolutional autoencoder.

From Fig. 11, it can be clearly seen that the convolutional autoencoder significantly outperforms the traditional filtering methods in denoising. From the noisy input, the features of the image can be clearly identified. Similar results are obtained in Figs. 12 and 13. However, for Sample 1200 (Fig. 12) although our autoencoder performs well and achieves better accuracy than the other filtering methods, the demarcation of the geological features seems to be more prominent in the results obtained using total variational filtering.

Along this line, an additional test was conducted where more images were used for training. Unlike the random Gaussian noise, these images were corrupted using uniform random noise. A total of 44800 images where used for training and the model is denoted by `ca_50_1_5` in Table 1. Out of these images, 20% were

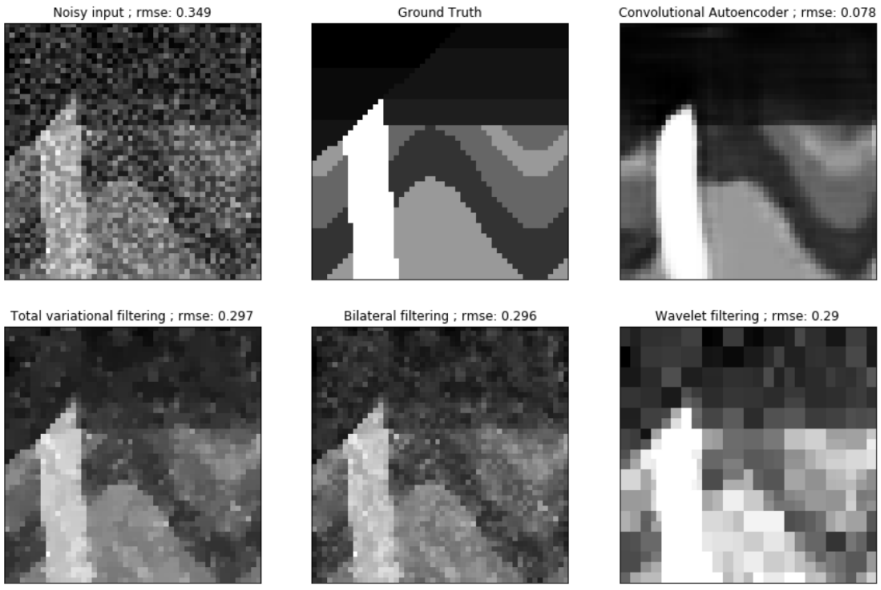

Figure 8: Noisy test image, the ground truth version, and the denoised results obtained using convolutional autoencoder, total variational filtering, bilateral filtering and wavelet filtering for test sample 320. The resolution of these images is 50×50 and the autoencoder model used here is `ca_50_1_3`.

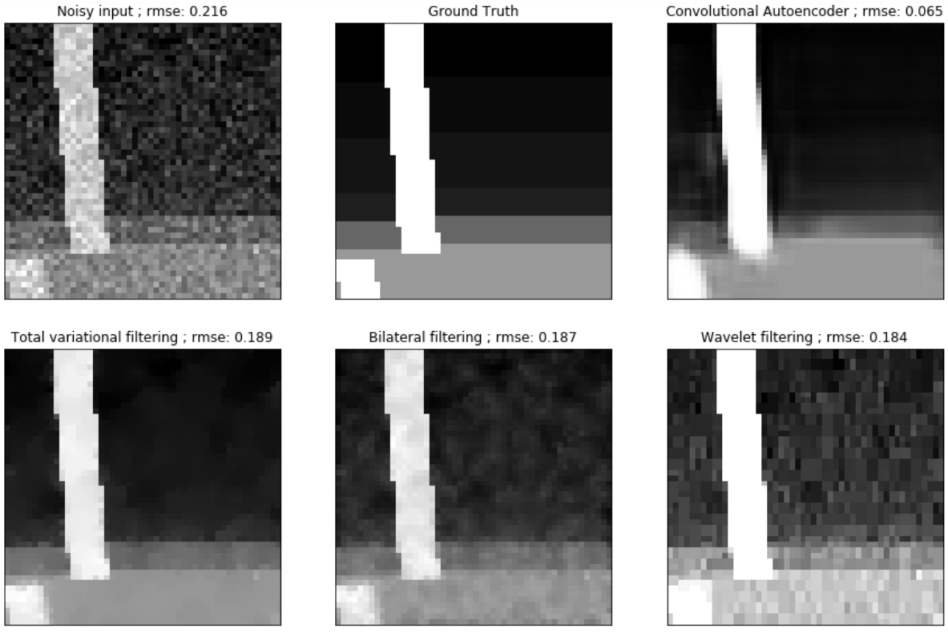

Figure 9: Noisy test image, the ground truth version, and the denoised results obtained using convolutional autoencoder, total variational filtering, bilateral filtering and wavelet filtering for test sample 1200. The resolution of these images is 50×50 and the autoencoder model used here is `ca_50_1_3`.

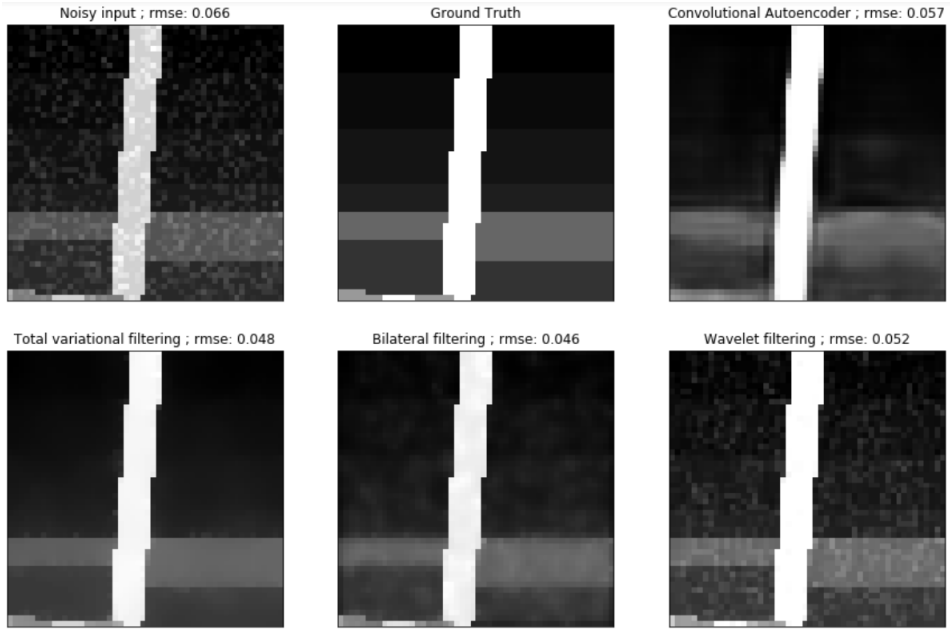

Figure 10: Noisy test image, the ground truth version, and the denoised results obtained using convolutional autoencoder, total variational filtering, bilateral filtering and wavelet filtering for test sample 968. The resolution of these images is 50×50 and the autoencoder model used here is `ca_50_1_3`.

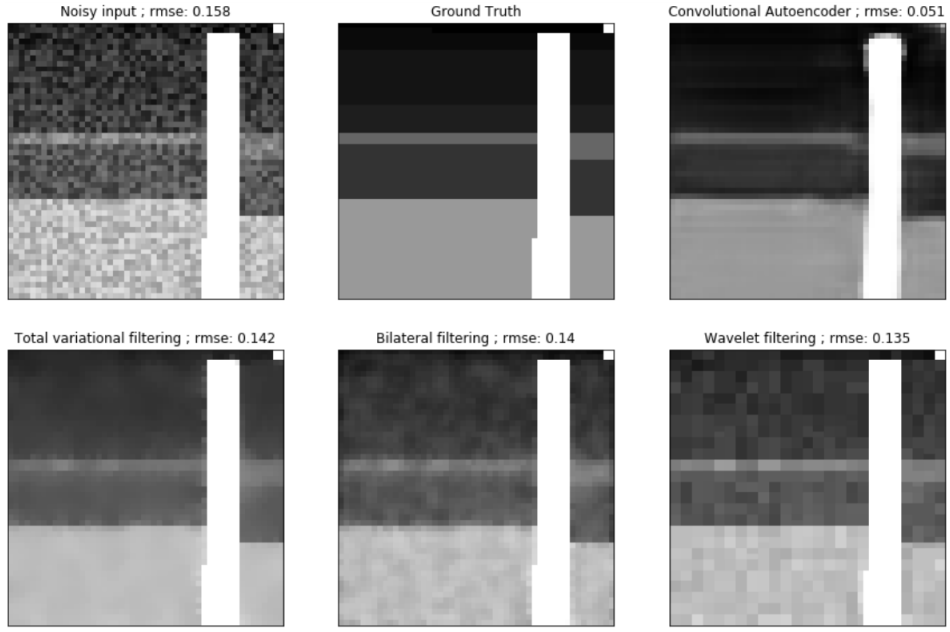

Figure 11: Noisy test image, the ground truth version, and the denoised results obtained using convolutional autoencoder, total variational filtering, bilateral filtering and wavelet filtering for test sample 2267. The resolution of these images is 50×50 and the autoencoder model used here is `ca_50_1_5`.

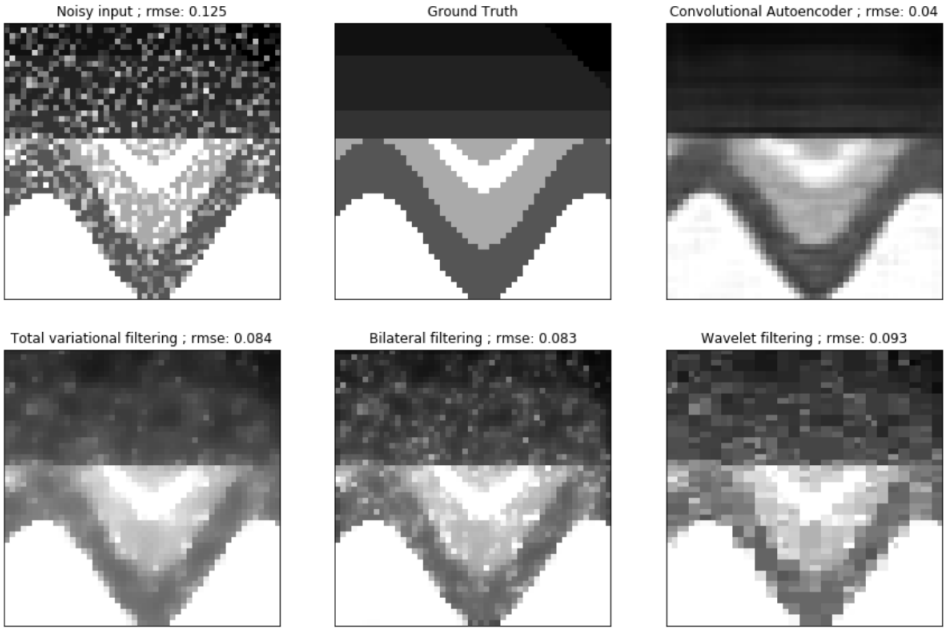

Figure 12: Noisy test image, the ground truth version, and the denoised results obtained using convolutional autoencoder, total variational filtering, bilateral filtering and wavelet filtering for test sample 820. The resolution of these images is 50×50 and the autoencoder model used here is `ca_50_1_5`.

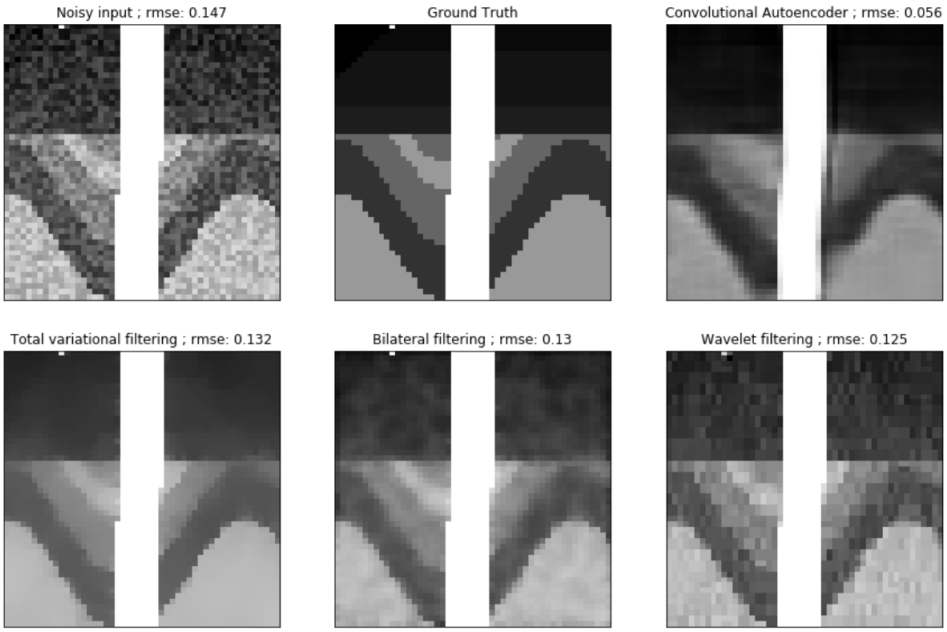

Figure 13: Noisy test image, the ground truth version, and the denoised results obtained using convolutional autoencoder, total variational filtering, bilateral filtering and wavelet filtering for test sample 2000. The resolution of these images is 50×50 and the autoencoder model used here is `ca_50_1_5`.

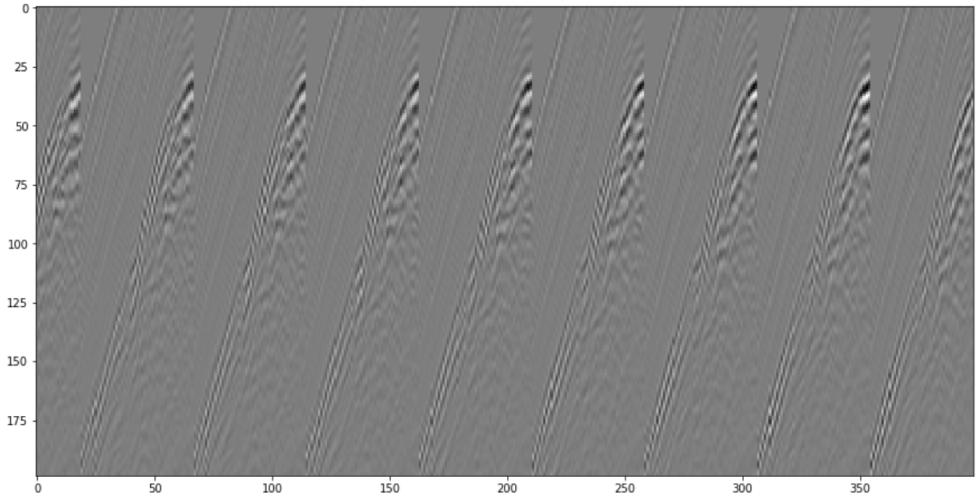

Figure 14: Small section of the large real seismic data obtained from [3], used for training and test purpose in this study.

noisefree, 40% were corrupted with random Gaussian noise and the remaining were corrupted with uniform random noise. Further, the trained model is tested on 11200 images and out of these 3 images are randomly chosen.The rmse and mae values for these 3 images obtained using the model as well as by the baseline filtering techniques are reported in Table 1.The 3 noisy inputs as well as their denoised versions obtained using the trained autoencoder model as well as those using the traditional filtering methods are shown in Figs. 8, 9 and 10. Visually, it seems that the autoencoder model when trained with more noisy images seems to perform better in terms of denoising. The demarcated geological boundaries seem to be more prominent. For all the 3 cases, the results of autoencoder are better than the filtering methods. Interestingly for sample 2000, the results of total variational filtering seem to be better than that of the autoencoder model.

Thus, from the various synthetic cases, it has been observed that the autoencoder model in general, when trained properly, can remove random noise. It has proved to be superior over the traditional filtering methods. However, it has been observed that for several cases, although the autoencoder model reduces the error on a global scale, it fails to recover the features of the image properly. Visually, at several instances, the results of total variational filtering seem to be superior over the autoencoder model. Clearly, the current mse loss funtion is not capable of handling this issue, and a different loss function is needed which is capable of denoising at a more localized level.

4.2 Real data

The concept of convolutional autoencoder has been shown to work well for removing random noise from synthetic geophysical data. However, the original goal of this project was to investigate its potential for the removal of random noise in real seismic data. Along this line, first it was investigated whether the autoencoder model `ca_50_1_5`, trained on synthetic data, could be used for removing random noise from real seismic data. For this purpose seismic section from [3] was taken. For visualiazation purpose, a small section of this large image is shown in Fig. 14. Although the results are not shown here, the model failed completely to work on this real seismic section.

Thus, it was decided to train a convolutional autoencoder model using real seismic data as training set. The details related to the trained model `ca_seismic_1` are described in Table 1. From the large seismic section, 88101 seismic images of size 50×50 each were extracted. Further two noisy versions of each of these images were generated, thus making the total dataset size as 264303. Out of this, 80% of the images were used for training. Once the model had been trained, it was tested on a set of 52861 images. From these images, we randomly chose 2 images to see whether the model could be trained for seismic data or not.

Figs. 15 and 16 show the results of convolutional autoencoder on two sample seismic slices. For both the cases, the resultant output was so bad that it was not needed to study the error values or compare with the

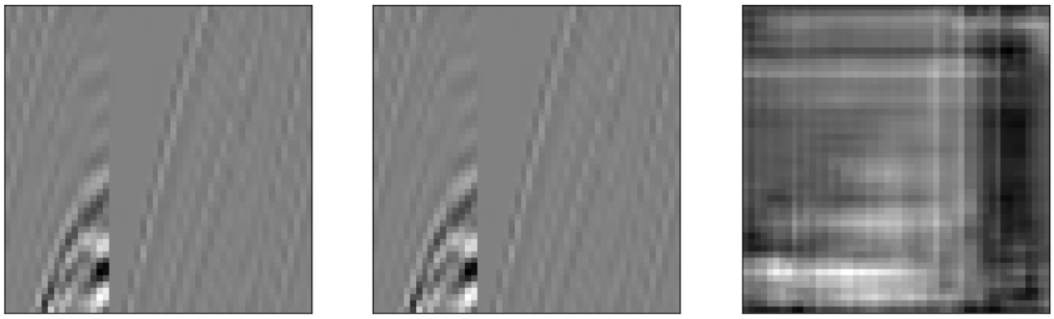

Figure 15: Synthetic geophysical images (with added normal noise) comprising multiple geological features on a resolution of 100×100 pixels. These images have been adapted by adding normal noise to the images available at [5].

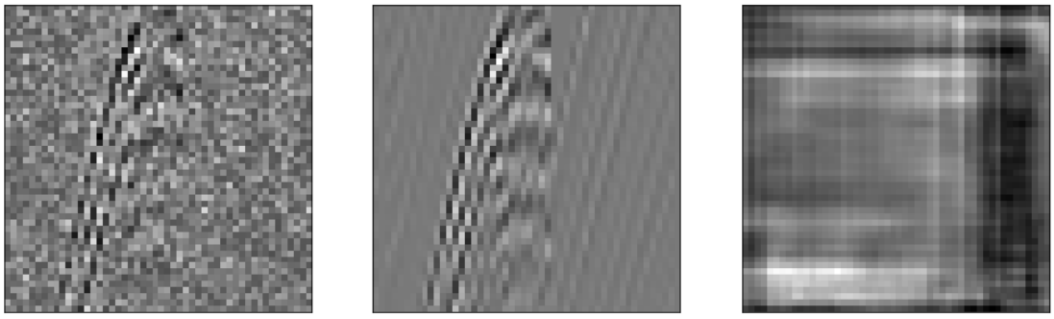

Figure 16: Synthetic geophysical images (with added normal noise) comprising multiple geological features on a resolution of 100×100 pixels. These images have been adapted by adding normal noise to the images available at [5].

filtering methods. Clearly, this is something would not be acceptable, hence the study on real seismic data was stopped here. It was identified that the resolution of the features in the seismic data is quite high, and the chosen size of the input images is too low to capture it.

Thus, from these numerical experiments, it can be deduced that convolutional autoencoders definitely have potential in denoising of the images. However, in particular for the seismic images used in thsi study, the chosen model complexity is not sufficient, and future research should include experimenting with larger images and more complex networks.

5 Conclusion

5.1 Free-form Visualization

Since the goal of this project was to investigate the potential of convolutional neural networks for denoising of seismic data, the most apt visualizations would be Figs. 15 and 16. Both these results state that the convolutional autoencoders trained in this work are not yet suitable enough to be used for removing random noise from seismic data, and further research work is needed.

5.2 Reflection

The goal of this capstone project was to gain-hands on experience on a relevant machine learning problem. The entire process from choosing the problem to discussion of end results can be summarized in the following steps.

1. The problem of removing random noise from geophysical seismic images was to be carried out using machine learning.
2. Freely-available datasets are limited. To circumvent this issue synthetic images were used. Further, real datasets from the geophysical community were obtained.
3. Benchmarks were chosen, in particular wavelet filtering, total variation filtering and bilateral filtering were used.
4. Several convolutional autoencoder models were trained with different sets of configurations.
5. Final tests were done of real seismic images and conclusions deduced.

Each of these steps has been a challenging one for me, and together these have been an immense source of learning. One of the biggest challenges with handling seismic data has been the choice of training data. Seismic data describes the geological representation inside the earth. Since the geology varies spatially a lot, neural network models trained on one region cannot really be used to process data from another region. Since the goal of this project was to remove random noise from seismic data, it was hoped that denoising autoencoder models trained on a massive set of synthetic geophysical images would be able to handle the seismic images. It was tried, and drastic failure was achieved. As stated earlier, the reason is the limited bandwidth of the trained models, and it is hoped that with larger datasets, improved results can be obtained. Further, the model that was trained with real seismic images also failed. Here the test data belongs to the same distribution from which the training set was chosen, yet it failed. The reason is that the real seismic images contain very complex fine features which our coarse models could not capture. With more powerful computers and higher resolution networks, this issue can be overcome.

The most interesting aspect of this project has been the improvement that could be gained on coarse images. Compared to the three filtering based denoising methods, significantly better results could be obtained using the convolutional autoencoders. This motivates me that even though the current models could not denoise the seismic images used in this study, there is certainly hope to do so, and it would be a future direction of research to pursue.

5.3 Improvement

To develop high-resolution denoising models based on convolutional autoencoders that can be reliably deployed, a few directions of improvement can be outlined and these are discussed in the points below.

1. The real seismic images are limited, and reliability is only based on synthetic images. Thus, the set of synthetic geophysical images that are used to train the model need to be throughly studied. It needs to be ensured that these images contain sufficient diversity to capture the complicated hidden representations of seismic data. Moreover, physics-based synthetic seismic images should be included. Forward-modeling of seismics is a complicated and computationally expensive process. However, with the powerful computers available in the geophysical industry, it should not be difficult to generate a wide-variety of physics-based synthetic seismic images, and this should be able to lead to significant improvements.

2. The choice of CNN network has not been explored much in this study. In particular simple, not so large networks, have been chosen. Clearly, during the decoding, the simplified upscaling possibly fails to recover important features of the images. Thus, the network needs to be thoroughly studied, and variations need to be experimented with.

3. For the autoencoder model trained using real seismic images, an image resolution of 50×50 is possibly not sufficient to capture the representations that exist in the seismic data. Seismic features span over a large set of points, and using higher-resolution/larger seismic images during training should be able to help.

A Reproducibility source

A.1 Project location

This project has been completed using Python, Tensorflow and Keras. The source code is available at [7]. Since the data size is large, it needs to be downloaded as a single zipped file from [6]. In the provided zip file, only files for `ca_50_1_5` are provided for experimenting. This has been done due to size limits. The downloaded zipped file needs to be extracted in the root folder such that `data` folder is adjacent to the `code` folder. In case you would like to experiment with the other models, the training and test data can be generated using the provided Python scripts and the raw data available in `raw` folder. If you have any questions related to the project, please contact the author at `GuptaDeepak2806@gmail.com`.

A.2 Project structure

- **denoise-autoencoder**: Root folder
  - **Code**: Contains the Python and iPython scripts, as well as the trained convolutional autoencoder models used in this study.
    - **trained_models**: Contains folders, each corresponding to a trained autoencoder network. Within these folders are contained the `*.h5` and `*.json` files to load the respective models.
  - **Data**: Contains the synthetic and real datasets used in this study.
    - **raw**: Contains the raw data obtained from various sources which is used to create our study data using certain processing.
      - **2D_Seismic_Alaska**: Contains the real seismic data obtained from [3], and `slices` folder containing image slices extracted from it.
      - `cseg_fault_dyke_fold_model`: Contains raw dataset obtained from [4].
    - **processed**: Processed data used in this study.
      - `2D_Seismic_Alaska`: Contains the input and output datasets, and the parts of these data used for training and testing the autoencoder model `ca_seismic_1`.

- `cseg_fault_dyke_fold_model_50`: Contains input and output datasets, and their partitions into training and test sets used for the autoencoder model `ca_50_1_1`.
- `cseg_fault_dyke_fold_model_50_2`: Contains input and output datasets, and their partitions into training and test sets used for the autoencoder model `ca_50_1_3`.
- `cseg_fault_dyke_fold_model_50_3`: Contains input and output datasets, and their partitions into training and test sets used for the autoencoder model `ca_50_1_5`.
- `cseg_fault_dyke_fold_model_100_1`: Contains input and output datasets, and their partitions into training and test sets used for the autoencoder model `ca_100_1_1`.

A.3 Python scripts

- `create_data_case1.py`: Takes a series of input images and generates their noisy versions by calling `noise_adder.py`.
- `noise_adder.py`: Adds uniform random or random normal noise of a certain amount to an input image.
- `simple_autoencoder.ipynb`: Contains script to train a convolutional autoncode network.
- `test_simple_autoencoder.ipynb`: Tests the performance of a trained network on test images.
- `test_filtering`: Compares the trained autoencoder network with the traditional filtering methods on the test data.
- `create_seismic_slices.ipynb`: Creates the seismic slices the large real dataset taken from [3].
- `add_noise_to_seismic`: Adds noise to a seismic slice.
- `create_test_seismic`: Ununsed yet due to not so good results obtained for the seismic case.

References

[1] Y. Bengio. Learning deep architectures for AI. *Foundations and Trends in Machine Learning*, 2(1):1–127, 2009.

[2] D. Bhowmick, D. K. Gupta, S. Maiti, and U. Shankar. Stacked autoencoders based machine learning for noise reduction and signal reconstruction in geophysical data. *Neural Computing and Applications (under review)*, 2019.

[3] SEG Open data. http://software.seg.org/datasets/2d/index.html.

[4] G. Ganssle. https://github.com/gganssle/cseg-imXlate.

[5] G. Ganssle. Denoising seismic records with image translational networks. *CSEG RECORDER*, 43(1):32–34, 2018.

[6] D. K. Gupta. https://drive.google.com/file/d/1zxc6j-u9hkvrmejtxs76hdlvnyrffwr3/view?usp=sharing.

[7] D. K. Gupta. https://github.com/dkgupta90/denoise-convautoencoder/.

[8] A. P. Valentine, L. M. Kalnins, and J. Trampert. Discovery and analysis of topographic features using learning algorithms: A seamount case study. *Geophys. Res. Lett.*, 40(12):3048–3054, 2013.

[9] Y. Wu, Y. Lin, and Z. Zhou. InversionNet: Accurate and efficient seismic waveform inversion with convolutional neural networks. In *SEG Technical Program Expanded Abstracts*, pages 2096–2100, 2018.

[10] Y. Xiong and R. Zuo. Recognition of geochemical anomalies using a deep autoencoder network. *Comput. Geosci.*, 86:75–82, 2016.

[11] T. Zhao and P. Mukopadhyay. A fault detection workflow using deep learning and image processing. In *SEG Technical Program Expanded Abstracts*, pages 1966–1970, 2018.