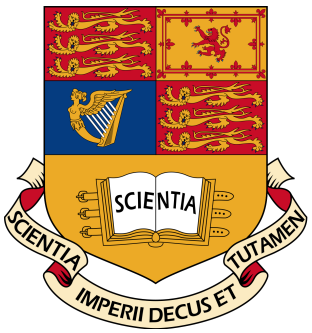

MSc Individual Research Project Plan

Imperial College London

Department of Earth Science and Engineering

MSc Applied Computational Science and Engineering

Neural Networks Applied to Signal Separation on Multi-Sensor Array Data

Author:
Mattia Guerri

Internal Supervisors:
Prof. Olivier Dubrule,
Dr. Lukas Mosser

External Supervisor:
Dr. Song Hou

June 28, 2019

Chapter 1

Introduction

1.1 General Outline

The present work serves as a documentation of the plans relative to the final Independent Research Project. The project is conducted in form of an internship with the company CGG, one of the world leading company in geophysical services. CGG is making available office space, datasets, computational resources and technical supervision. The project is supervised by Imperial College Pr. Olivier Dubrule and researcher Dr. Lukas Mosser (hereafter defined as *internal supervisors*), and by CGG employee Dr. Song Hou (hereafter defined as *external supervisor*).
In this first chapter I give a brief explanation of the project background, its rationale, its goal, and a list of the project objectives. In the next chapter, I discuss the objectives and the tasks that will be necessary to accomplish in order to achieve them. In addition, a brief literature review concerning neural networks and their application in image analysis is given. In Chapter 3, I detail the timing of execution of the project, its milestones and corresponding deliverables.

1.2 Background

Seismic surveys are a tool of prime importance in hydrocarbon exploration [14]. When conducted offshore, these surveys typically consist in a ship towing a number (usually 2) of airguns (the sources) and cables. Attached to each cable there is an array of hydrophones (the receivers). A detailed description of the operations required for off-shore seismic data acquisition is available in [4]. In brief, the airguns are fired to produce compression waves that propagate in the water, reach the seabed and propagate through it in the subsurface. Downwards propagating waves are reflected upwards when encountering discontinuities in the medium acoustic impedance (function of the material density and compressional wave velocity). Reflected waves are recorded by the receivers. The resulting dataset is properly processed in order to obtain a representation of the acoustic impedance discontinuities present in the subsurface. These are interpreted from a geological point view, on multiple levels, estimating for example, the geological structures and, in favourable cases, the lithologies or the physical attributes of the rocks (density, porosity, etc.). The final goal is to identify geological systems (the so-called plays and prospects) that are prone to generate and store hydrocarbon resources, as oil and natural gas. Later in the life of the field, seismic is utilised to improve the reservoir models.

1.3 Rationale and Goal

During a dataset acquisition, the 2 airguns are usually fired sequentially, with a certain waiting time between the fire of one source and the other one (approximately from 8 to 12s). The waiting is necessary for the vanishing of all the energy generated by the previous fire. Alternatively, in a simultaneous (or blended) acquisition, the waiting time between the fires is reduced. This results in a faster, hence cheaper acquisition. Another approach followed when obtaining blended acquisitions consists in firing multiple sources within the same survey time, with the effect of having better data quality. Combination of the two approaches, reduced waiting time and multiple sources, is also possible, combining the benefits, faster acquisition and better data quality, respectively. The downside of these techniques is that the recorded signals are generated by the interference of waves

coming from multiple sources. The blended dataset, characterized by these interferences, requires additional pre-processing procedures with respect to the not blended ones. Such procedures, aiming to isolate the signals generating by the single sources, are identified as signal de-blending. Algorithms performing signal de-blending require enormous computational time and resources.
The goal of the project is to explore the possibility of employing neural networks algorithms to perform the signal separation process. The idea is to train the networks on the results obtained with the standard de-blending algorithms already employed by CGG. Training neural networks is also a computationally demanding task, however, once the training is completed, the inference stage, which consists in using the trained network to obtain new results, is a relatively fast procedure. The possibility of obtaining a signal separation algorithm that is faster than those already available is of great interest to CGG. It would allow indeed to save time and computational resources, which translates in more resources available for other tasks, and a faster delivery of the products to the clients.

1.4 Objectives

The project objectives are the following:

1. **Literature Review**
2. **Software**
3. **Results**
4. **Report**

Each objective is broken down in specific tasks. Description of the objectives and how I intend to execute the tasks constituting them follows in the next chapter.

Chapter 2

Objectives

2.1 Literature Review

There are two topics to be reviewed, i) seismic data acquisition, in particular marine acquisitions; ii) application of neural networks in image analysis. A short description of the first topic has been given in section 1.2. Here I briefly review neural networks and their applications in image processing. A more detailed review of both fields will be featured in the final report.

2.1.1 Neural Networks and Their Application in Image Analysis

The last decade has seen an enormous amount of resources invested in the research and application of algorithms classifiable as *neural networks* (other terms often used to indicate such methods are *deep learning*, *machine learning* and *artificial intelligence*, but note that these expressions have different meaning and should not be used as synonyms). Neural networks have been known for decades [2]. Only recently though two major factors have made their application extremely effective. These are, i) an unprecedented explosion in the amount of digitised data, ii) an equally unprecedented amount of vast (and cheap) computational power. Neural networks are effectively applied in numerous fields. Their superiority with respect to other methods is sometimes so significant that a term like *disruption* is often used by the media to describe the introduction in a certain field of artificial intelligence in general, and neural networks in particular. The strong scientific and industrial interests surrounding this kind of algorithms translate in a vast literature and on-line material dealing with it. Here I limit my effort in defining and briefly address the main publications presenting techniques that are applicable to the problem at hand.
A Neural network is defined by a number of layers. The first one, called the input layer, is represented by the input data. In the simplest form of a neural network, the input data is a vector. Each layer performs certain numerical operations on the elements of the input vector and generates an output vector that is fed into the next layer. In the so-called *supervised learning* approach, the network is trained to generate a certain output in response to the input of a certain data. To perform the training, the network necessitates examples of what kind of output is expected when considering a certain input. These already defined outputs are called *targets*, since they represent what the network has to be able to reproduce. Input/targets couples are called *training examples* and they are used in the training stage (see figure 2.1 for examples of input, output and target relative to the project). Numerically, the training consists in minimizing a function quantifying the difference between the network output and the target. After the training stage is completed, the performance of the network is established testing its ability to reproduce a new set of targets, which were not used in the training process. When dealing with images, it is convenient to input in the network a 2 dimensional array instead of a vector. In this case, the fundamental numerical operation performed by the network is convolution. It consists in applying kernels (also defined as *filters*) of various sizes to the image. The output of a number of kernels is joined in a single volume and used as the input for the next set of convolutions. This creates a structure organized in layers, each one defined by convolutions involving a number of filters. Networks of this kind are known as *Convolutional Neural Networks* (CNNs). The structure characterising a network is often defined as its *architecture*. In the following, I will use the terms *network*, *architecture*, *algorithm* or simply *model* as synonyms.
In the field of computer vision, CNNs are used for various tasks, for example image classification, object detection and image semantic segmentation. The specific issue targeted by this project shares similarities with the latter. Image semantic segmentation consists in, given an input image,

classifying every pixel of the image in a certain number of classes. If the input image pictures a road for example, these classes might be, cars, road, sidewalk, building and so on. Image segmentation techniques are applied in many industries, some of them bearing significant economic importance. For instance, image segmentation is a fundamental operation in self driving cars technology and robotics. One of the first successful applications of CNNs in image processing was introduced in [7], nowadays considered a milestone in the development of machine learning. Their algorithm was capable of identifying images of hand-written digits with staggering accuracy (approximately 99.2% on the test set). More recently, another fundamental step in the development of CNNs was represented by the algorithm known as AlexNet [6]. In performing image classification, AlexNet achieved a performance largely superior with respect to those displayed by other, more traditional, methods. Such a decisive success spurred a vigorous interest in CNNs and since then many algorithms featuring convolutional layers have been proposed. Some notable examples are the models known as VGG16 [15], GoogLeNet [16] (introducing the renowned inception module) and ResNet [3]. A first significant development of an architecture specifically targeting image semantic segmentation was introduced in [10]. In this work the authors modify well-known CNNs (Like VGG16 and GoogLeNet), that were essentially developed to perform image classification, in order to adapt them to the image segmentation task. Their architectures, featuring only convolutional layers, are defined as *Fully Convolutional Networks* (FCNs). The same year, an extension to these models was proposed in [11]. They presented networks constituted by a convolutional and a transposed convolutional side. Transposed convolution can be seen as the inverse operation of convolution. While the first one usually reduces the size of the input image, the second one increases it. Another extension to [10] was put forward in [13]. Their architecture, known as *U-Net*, also features a convolutional and a transposed convolutional path. In addition, new operations are introduced aiming to transmit information from shallow to deep layers, bypassing layers in between. Other architectures, all based on convolutional layers, have been implemented to tackle the image segmentation task, some examples are: ParseNet [9], FPN (Feature Pyramid Network) [8], DeepLabv3+ [1], and many others (see [12] for a comprehensive review of the topic).

2.2 Software

2.2.1 Models Selection and Adaptation

In developing the code, the first step is to choose which architectures are going to be used in the project. As stated above, the algorithms that better suit the issue of interest are those employed in image semantic segmentation. After particular architectures have been chosen, they can be implemented from scratch, following the corresponding publication. Alternatively, the code can be retrieved from on-line sources, since these algorithms are usually open-source. Once the code is obtained, it is necessary to adapt it to my particular goal. Image segmentation methods usually classify the input pixels in classes. In my case instead there are no classes involved, the goal is simply to minimize the difference between the network output and the target. In Figure 2.1, top-left panel, I show an example of the input data used in the project. It consists in a one channel image essentially obtained plotting the recordings of all the receivers positioned on one of the cables towed by the ship during acquisition. The signal generated by one source develops from the left margin of the panel. In its centre, the dome like structure is the signal produced by the second source. This image is therefore presenting the blended signals. The top right panel shows the target that the neural network is supposed to reproduce. It is clearly visible how the signal corresponding to the second source has been removed from it. The image was produced employing CGG's standard de-blending algorithms. These Input/target couples will constitute the training examples. The bottom left panel shows the output that one of the neural networks that I have recently developed (CGG-12, see following section) is able to generate after the training stage is executed. The goal is for this output to be as close as possible to the target. The general structure of the target is indeed reproduced. However, observing the bottom right panel, which is the difference between the output of the network and the target, we can see how the network leaks some of the signal. The main challenge of the project will be to develop an algorithm, or an ensemble of them, capable of reducing the leakage as much as possible.

2.2.2 Models Development

This task consists in moving beyond already established algorithms, and building up new architectures specifically designed to tackle the problem at hand. CGG engineers have already developed

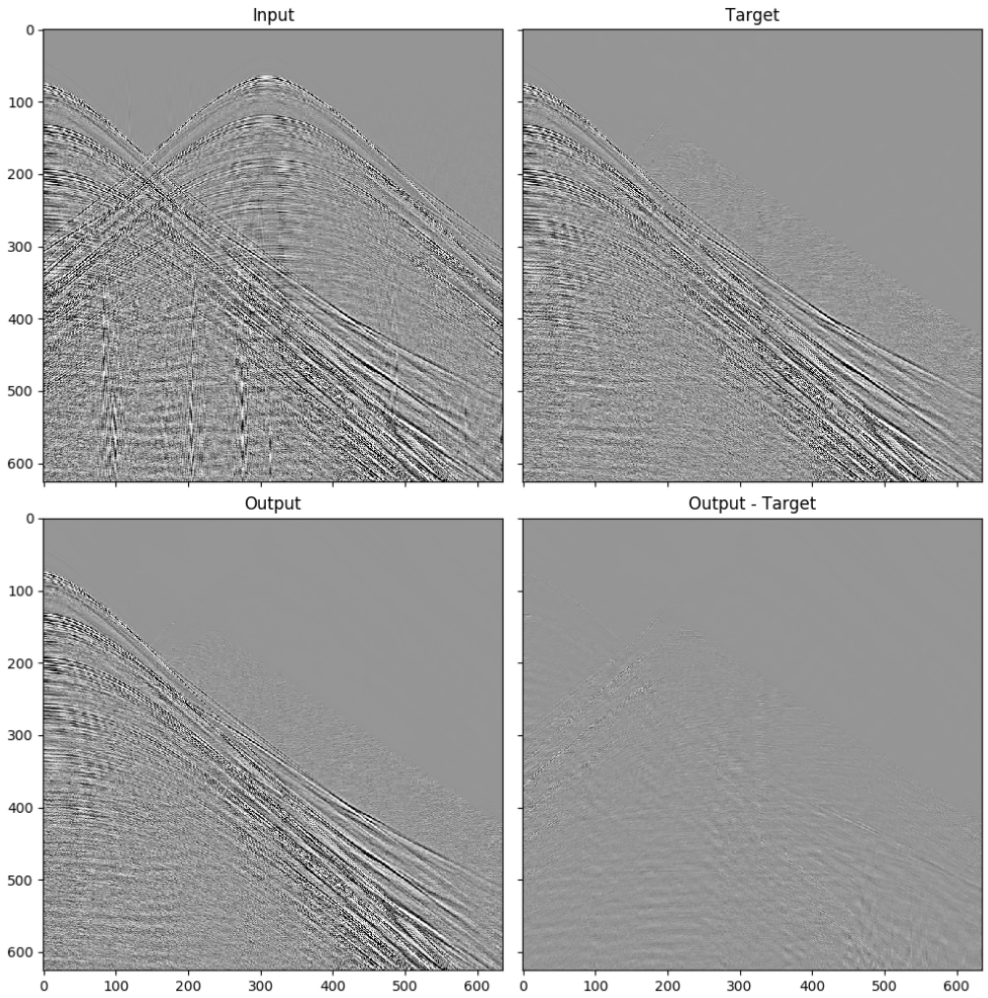

Figure 2.1: **Top left:** Example of the input data that is fed into the neural network. **Top right:** De-blended signal obtained using standard procedures. **Bottom left:** De-blended signal obtained with the neural network CGG-12. **Bottom right:** Difference between the two de-blended signals.

a convolutional neural network, formed by 8 layers. The algorithm (indicated as CGG-8) is similar to the approach presented by [10]. The layers maintain the same height and width of the input data and the output of the last convolution is directly compared with the target in order to compute the loss function in a pixel-wise fashion. The loss function quantifies the difference between the neural network output and the target. At this stage it is a simple L_2 norm, but the effects of other functions will be explored. I have first tested the performance of this algorithm. Then, I have made some modification on it, adding input data normalization and standardization, and implementing batch normalization [5]. I have then developed similar architectures, in particular adding more layers, testing structures characterised by 12 (CGG-12) and 15 (CGG-15) layers. The first experiments revealed that normalizing/standardising the input, adding more layers and using batch normalization results in algorithms showing better performance with respect to CGG8. Given the large memory required by the deeper (featuring more layers) models, I have developed the algorithms in a way that they can run in parallel on multiple GPUs (Graphic Processing Units). I intend to develop even deeper architectures, aiming to obtain models showing better accuracy. This will require the implementation of the so-called residual-block [3], which helps in coping with the issue of gradients vanishing/exploding and facilitates the convergence of the algorithm. Furthermore, it will be interesting to build stack of different architectures. For example, an FCN structure, with convolutions characterized by relatively small kernel size, can be joined with a convolutional/transposed convolutional architecture, featuring convolutions having larger kernels. The goal would be to obtain an architecture capable of extracting information at multiple length scale from the input images. Another viable option pointing in the same direction, is to fit an FCN with inception modules [16]. An inception module consists in stacking up together in a single volume the output obtained by a different operations, for example, convolutions with different

kernel sizes. The different size of the kernels adopted in the convolutions should again increase the ability of the network to learn patterns characterized by different length scale. Many other experiments can be attempted, with the only limitations being represented by the time constraint and hardware memory limits.

2.2.3 Models Testing

As established architectures are adapted and new ones developed, the following step is to test their performance. The usual practice is to train the algorithm on a first set of data, the training set, and to validate it on a second one, the validation set. The accuracy of the algorithm is finally assessed on a third set of data, the test set. An important operation consists in exploring the effects of changing the so-called *hyper-parameters*, a process known as *hyper-parameters tuning*. In machine learning, the term *hyper-parameters* indicates all those properties of the network that remain constant throughout the training process, as opposed to the parameters that are instead modified by it. Some examples are, number of layers, kernel sizes defining the convolution operations, and number of kernels constituting each convolution. Models testing, and the other tasks presented in the previous two sections, are being conducted in parallel. This is necessary considering the time required to train a model, which can range from few hours to multiple days. For example, while hyper-parameters tuning is carried out on one model, a new, previously untested architecture can be trained. Running multiple algorithms at the same time is possible thanks to the computational resources made available by CGG. At the moment I have access to various server nodes, each of them fitted with a number of GPUs. Using GPUs to train neural networks is not strictly necessary. The algorithm can be perfectly trained on a CPU (Central Processing Unit). A GPU though as many more computational core than a CPU, implying a significant reduction in the time necessary for training the model.

2.2.4 Software Developing and Reviewing

In terms of the code, the final product of the project is a software that wraps all the tested networks and includes modules for plotting the results. The software is a stand-alone package. The user will be able to interact with it through an input file consisting in a list of variables that define the experiment, for example, which network will be trained, paths to the folder containing the input data and to the one containing the network output. I plan to produce a document briefly explaining the main features of the code and listing the requirements necessary in order to run it successively. In addition, I intend to add a small example dataset and a Jupyter notebook performing a demonstration of how the software should be used and its main capabilities.

2.3 Results

This objective consists in gathering all the results obtained with the various models and analyse them. The focus will be in investigating the following issues: i) understanding what is the optimal amount of data on which to train the algorithm on, this is important to assess as a larger dataset implies a longer time necessary for the training process; ii) exploring the limitation of the models, such as recovery of signals characterized by different amplitudes and frequencies; iii) analyse how a network trained on a dataset acquired on a certain geological setting performs on other datasets, linked to completely different geological settings. Once the results have been produced, I will plot figures summarizing the main points evidenced by the experiments. The plan is to include the figures, or a selection of them, in chapter 3 and 4 (Results and Discussion, respectively) of the final report.

2.4 Report

The last objective is represented by the final report. It will be structured in abstract, introduction, methodology, results, discussion, conclusion and bibliography. It will be important to have a first draft of the report a few weeks before the project final deadline, in order to allow enough time for correction and reviewing. More details regarding the time schedule of the project are presented in the next chapter.

Chapter 3

Project Time Schedule

The literature review, for what concern both marine seismic surveys and neural networks applications in image analysis, has been largely accomplished and supposed to be completed by the end of June. Additional work will be done in the future in relation to specific issues that might arise as I proceed in the development of the project. I plan to finish the task of models selection and adaptation by the end of the second week of July (14/07). The development of new architectures is going to be completed by the 21^{st} of July. In order to have enough time to analyse the results and, if needed, performs further experiments, I plan to have all the testing accomplished by the end of the fourth week of July (28/07). There will be an intermediate deadline in mid-July (14/07), when a preliminary version of the software will be submitted to the internal supervisors for revision. A more elaborate version, also implementing the supervisors corrections, is supposed to be sent back for further revision at the end of July (28/07). During The last week of July and first week of August (from 29/07 to 04/08), I will gather all the obtained results, analyse them and produce figures exemplifying the major points highlighted by the analysis. Following this, it will probably occur that some more experiments are needed. Scheduling this task at the beginning of August will allow enough time for that. August 4^{th} represents the deadline for sending to the internal supervisors the figures relative to results display and analysis. As said above, the idea is to feature such figures in Chapter 3 (Results) and 4 (Discussion) of the final report. During the first two weeks of August (from 29/07 to 11/8) I will write a first draft of the final report. It will be particularly important to have chapter 3 and 4 ready and submitted to the internal supervisors by August 11^{th}. This is because the revision of these chapters by the supervisors might reveal the need for further experiments. The remaining time before the final deadline (from 12/08 to 30/08) will be employed for conducting further experiments (if needed) and for implementing the corrections and feed-backs received by the supervisors, both on the software and the report.
Below is a Gantt chart exemplifying the time schedule of the project. The project milestones (MS in the chart) represent key dates where I plan to complete tasks and submit the outcome to the internal supervisors. The milestones are the following:

- **MS1, 30/06:** Literature review completed. Introduction (Chapter 1 of the final report) section regarding the literature review is written.
- **MS2, 14/07:** First preliminary version of the software is sent to the internal supervisors for revision.
- **MS3, 28/07:** The software is completed and sent to internal supervisors for further revision.
- **MS4, 04/08:** The results have been collected and analysed. Figures displaying major results and exemplifying their analysis are sent to the internal supervisors.
- **MS5, 11/08:** First draft of the final report is completed and sent to supervisors.
- **MS6, 30/08:** Corrections regarding both the software and the report are implemented, the final versions of these components are uploaded into the project GitHub folder.

For what concern the external supervisor, the interaction will be continuous. Two kinds of meetings have been already scheduled. The first one (weekly) is specific to the de-blending process. Other engineers involved in the development of signal separation algorithms, both the standard ones and the new ones based on machine learning, will be present. The second one (bi-weekly) is more general and it is an occasion where all the interns (and their supervisors) involved in projects dealing with machine learning can present and discuss their results.

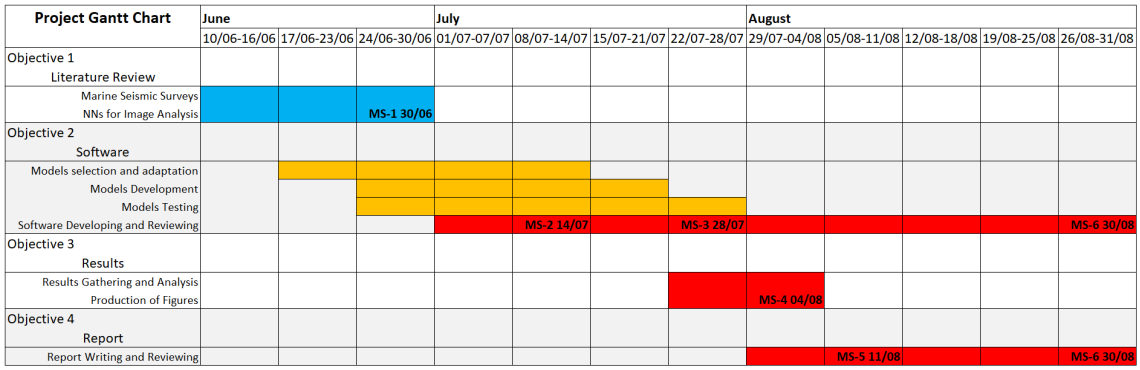

Figure 3.1: **Project Gantt Chart:** Light blue is for accomplished objectives, orange for those in development and red indicates future ones. MS stands for milestone (see text above for details).

Bibliography

[1] L. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. *CoRR*, abs/1802.02611, 2018.

[2] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. The MIT Press, 2016.

[3] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. pages 770–778, 06 2016.

[4] IAGC. An overview of marine seismic operations. pages 1–50, 01 2011.

[5] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. pages 448–456, 2015.

[6] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012.

[7] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. In *Proceedings of the IEEE*, pages 2278–2324, 1998.

[8] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie. Feature pyramid networks for object detection. In *CVPR*, 2017.

[9] W. Liu, A. Rabinovich, and A. C. Berg. Parsenet: Looking wider to see better. *CoRR*, 2015.

[10] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

[11] H. Noh, S. Hong, and B. Han. Learning deconvolution network for semantic segmentation. *CoRR*, abs/1505.04366, 2015.

[12] W. Rawat and Z. Wang. Deep convolutional neural networks for image classification: A comprehensive review. *Neural Comput.*, 29(9):2352–2449, Sept. 2017.

[13] O. Ronneberger, P.Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, volume 9351 of *LNCS*, pages 234–241. Springer, 2015. (available on arXiv:1505.04597 [cs.CV]).

[14] R. Sheriff and L. Geldart. *Exploration Seismology*. Cambridge University Press, 1995.

[15] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014.

[16] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *Computer Vision and Pattern Recognition (CVPR)*, 2015.