Preliminary Report

Automatic Seismic Interpretation Techniques

Tayfun Karaderi

Department of Earth Sciences, Imperial College London, United Kingdom

Supervisors: Ehsan Naeini (Ikon), Olivier Dubrule (Imperial), Lukas Mosser (Imperial)

1. Introduction

Automatic interpretation has long been an active topic of research and innovation for seismic exploration and reservoir characterisation workflows. Over the years various algorithms have been introduced and implemented for interpreting salt bodies [1], first breaks [2], horizons [3], channel bodies and faults [4]. However, most of the practical implementations, especially in commercial software, often rely on choosing a set of attributes in advance and then training the classification algorithm to discriminate between different classes which still involve human tuning and validations which ultimately mean such methods are time-consuming.

In this project, we propose to use supervised deep learning methods for automatic interpretation which has the advantage of combining attribute extraction and classification in one network. Our main objective is to analyse performance in terms of prediction accuracy and computation time of various deep learning techniques (e.g. sliding window classifier approach and fully convolutional networks) as well as various neural network architectures (e.g. auto-encoders and U-Net) for seismic image segmentation. Then, the best technique for seismic image segmentation would be incorporated into the RokDoc software so that a generalist geoscientist can make use of the machine learning method to train a model from a small subset of labelled seismic data and predict the labels for the entire volume of their seismic dataset with ease, not worrying about writing custom software for each use case.

In this project, we make use of the seismic dataset containing 12650 inline slices of size 401x251 pixels obtained from the Forties oil field in the North Sea (UK production block 21/10) by BP. RokDoc software will be used throughout the project which has an API consisting of an XML file for creating the user interface and a Python file for programming the algorithmic part of the software.

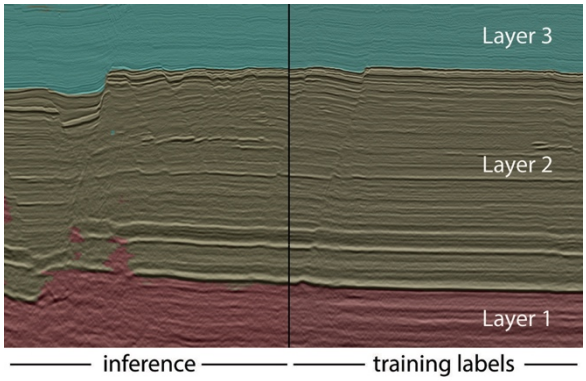

Figure 1 : Classification on a 3D seismic dataset by Badrinarayan et al [11]. Right hand side shows the inline slice used for training, and the left side shows the predictions after training.

2. A Brief Literature Review

In the last few decades, it was shown that simple neural network architectures such as MLPs (multi-layer perceptrons) can be used for automatic seismic interpretation techniques such as to segment seismic facies in 3D volumes [7, 8, 14]. However, these methods heavily relied on choosing multiple attributes in advance and, only very recently, the next generation of machine learning techniques such as CNNs (convolutional neural networks) have been successfully applied for automatic seismic interpretation (e.g. by Waldeland et al. [1, 10] and by Badrinarayan et al [11] (Figure 1)). This transformation of automatic interpretation techniques has been empowered by algorithmic advances in the field of machine learning (e.g. developments of CNNs by Krinzhevsky in 2012 [9], GANs by Goodfellow in 2014 [12], U-Net by Ronneberger 2015 [13]) as well as the developments in open source libraries and geoscience specific libraries, GPU enabled high-performance computing and cloud computing and the emergence of data analytics platforms.

Waldeland and Solberg [1] were the first to use supervised deep learning methods in seismic datasets. In their work, they used a sliding window approach where 65x65x65 cube patches were extracted from their seismic data and a 7 hidden-layer CNN was used to classify the center pixels of the overlapping patches over and over again to classify each pixel in their dataset as salt or not salt. In this paper, it was mentioned that one of the challenges of working with seismic data is the very large size of the data requiring a high amount of memory in the GPU which limits the size of their input in order to justify their choice of using small patches and a sliding window approach. They mentioned that training the model on a single labelled inline slice and using data augmentation techniques of random stretching (±20%), random flipping of x and y axis, and random rotations (±180° in x, y and ±15° in z) was sufficient to classify the rest of the dataset. The work of Waldeland then inspired the summer project of Charles Rutherford IIdstad in ConocoPhillips in 2017 who implemented the MalenoV [5] (machine learning of Voxels) software for multi facies classification on 3D seismic datasets which uses a very similar sliding window image segmentation method as the Waldeland's implementation. In their GitHub repository, it was mentioned that their classification time was the major problem of their sliding window approach. Then, their implementation on Matlab using Keras was translated into Python using Pytorch by Lukas Mosser. This report is available at [6] on Github.

3. Work Done in the First Two Weeks

This PyTorch implementation of MalenoV [6] by Lukas Mosser was taken as the starting point for our project. This GitHub repository contained python files, Jupyter notebooks and data files for the Dutch F3 seismic dataset. The most important parts contained in this repository were the train.py and test.py files which are used to train a model using one slice of labelled data and to make label predictions on a different slice of the seismic cube. These predictions are saved into two different files of labels.npy and indices.npy which contains the labels and their indices on the seismic cube respectively. These two python files have dependencies in four different python files of datasets.py, model.py, options.py and utils.py which all have several classes and functions in them, but their main usages are to slice the 65x65x65 seismic cubes around the indices for training and predicting, define the model to be used for training, set the command line options, and to save the checkpoints (per epoch) of the trained model respectively. There are additionally three Jupyter notebooks, one of them is to convert SEGY type data files into NPY type files, another is to split the training inline slice into training and validation sets and save them into train_split.npy and val_split.npy files, and the other notebook is to visualize the predictions saved by test.py file.

Although this was a relatively recently written software some of the libraries imported were old and no longer existed and minor changes had to be done to make it work. For example, the shuffleSplit class imported from sklearn.cross validation in several python files no longer existed and moved to sklearn.model_selection. The internals of this class were also changed, and minor changes had to be done inside the python files. A few minor changes were also made to make this software easier to work with (e.g. a function was added into train.py to automatically plot training and validation accuracy/loss as a function of number of epochs, the original function used to split training and validation sets was not understood and rewritten, the visualization tool for predictions was improved).

Once the software has been made to work it was applied on the F3 dataset (which has 9 facies classes). On the original GitHub repository, the labels were given for only one inline slice (index 239) of the seismic cube. Therefore, this slice was split into training and validation sets with roughly equal amounts of labels for each class in the training and validation sets. A test set was not used since no labels for the other inline slices were provided. Then, the given default 8-layer model (figure 2) was trained for 10 epochs with a batch size of 64 on the training set (figure 3) and the accuracy and losses on the validation set were obtained (figure 5). Then, test.py was used to make label predictions on the inline slice with index 147 and this slice was visualised (figure 3).

	Size of input image n	Number of input channels	f	p	s	Size of output image (n+2p-f)/s+1	Number of output channels or filters	Number of output neurons	Size of Filter + 1	Number of Parameters
Conv1	65	1	5	2	4	17	50	245650	126	6300
Conv2	17	50	3	1	2	9	50	36450	1351	67550
Conv3	9	50	3	1	2	5	50	6250	1351	67550
Conv4	5	50	3	1	2	3	50	1350	1351	67550
Conv5	3	50	3	1	2	2	50	400	1351	67550
FC1	400							50	401	20050
FC2	50							10	51	510
Softmax	10							9	11	99
							Total Neurons	290169	Total Parameters	297159

Figure 2: The CNN model used for training on the Dutch F3 dataset. Batch normalization was used between each convolutional layer as well as a dropout rate of 0.2. The model used for the Forties dataset is identical to the above but the final SoftMax layer has 5 instead of 9 output neurons.

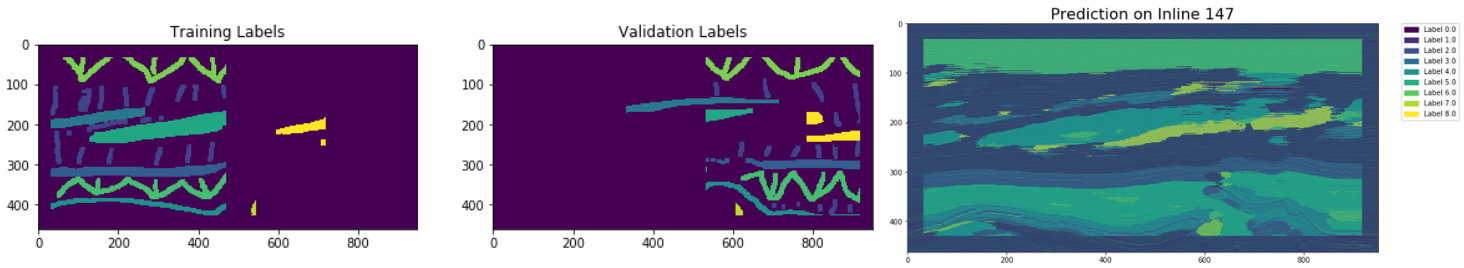

Figure 3: The 239th inline slice of F3 dataset used for training was split into training and validation sets as in the figure. On the right-hand side, we see a prediction on the 147th inline slice.

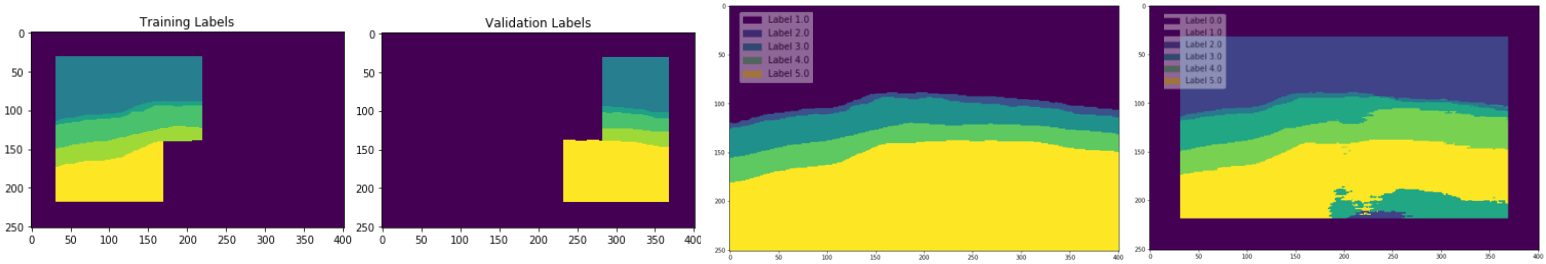

Figure 4: The 12405th inline slice of Forties dataset used for training was split into training and validation sets as in the figure. On the right-hand side, we see the actual labels of the 12410th inline slice followed by our prediction on the 12410th inline slice. Note that we have 5 labels and label 0 means unlabelled data on the edges of the inline slice where a window block could not be fit.

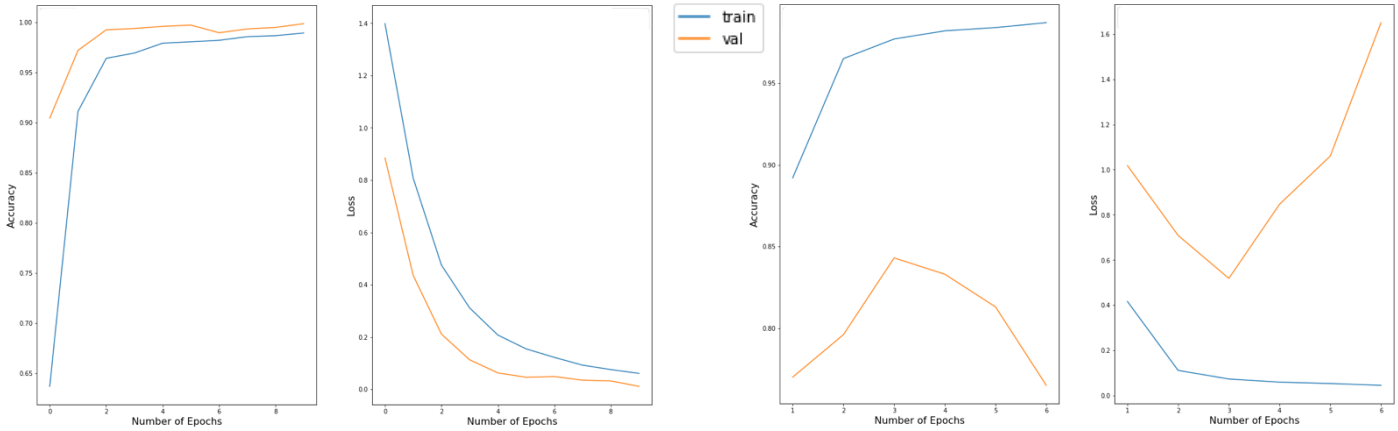

Figure 5: Accuracy and loss as a function of the number of epochs for the F3 dataset on the left and the Forties dataset on the right.

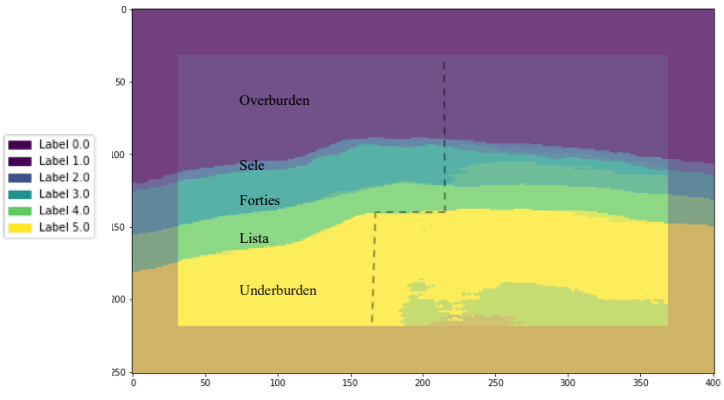

Figure 6: Predicted labels painted onto the actual labels of the 12410th inline slice of the Forties dataset. The four horizons can be seen in this plot. The single coloured regions between horizons show agreement and regions of mixed colours show disagreement between predicted and actual labels. The left side of the dashed line indicates the region of 12405th inline used for training the model.

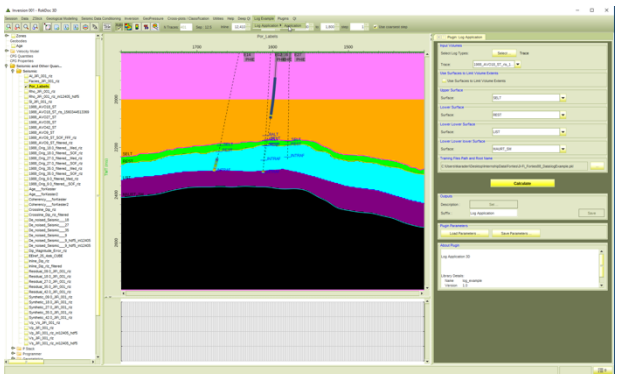

Figure 7: The labelling tool used for the Forties dataset.

In the accuracy plots (figure 5) for the model trained on the F3 dataset, it was observed that the model neither overfitted nor underfitted the data and 98.9% accuracy was obtained on the validation set in 10 epochs of training.

After testing on the publicly available F3 dataset the model in figure 2 was adapted for the Forties dataset by setting the number of output neurons of the final fully convolutional layer to 5 (since the Forties dataset has 5 different classes). Then, the labelling code in the RokDoc which used to take in an upper and lower horizon surface to label the in-between was adapted to take in 4 horizon surfaces (as in figure 7) to obtain the 5 labels for the entire volume of the Forties dataset. Then, the labels and the seismic cube amplitudes data were imported out of the RokDoc for training purposes using the GPU provided on the Google Colab cloud. The 12405th inline slice was used for training purposes and was divided into training and validation sets (as in figure 4). Upon training the model for five epochs it was evident from our training and validation accuracy graphs (figure 5) that the model overfitted the data. Then, the model trained for four epochs which had a validation accuracy of 81.4% was used to predict the labels of the 12410th inline slice (figure 4). This prediction had an accuracy score of 88.1%. However, the overfitting was also very evident from the fact that the seismic data used for training clearly seemed to be much more accurately predicted. Figure 6 highlights this overfitting by showing that the actual labels and predicted labels on 12410th inline slice align closely in the training region of 12405th inline slice but not as much in the other regions.

4. Proposed Approach

The greatest disadvantage of the sliding window approach applied to seismic datasets was highlighted as the segmentation time for the MalenoV software in their original GitHub repository [5]. The reason for this being that a seismic cube patch of size 65x65x65 was being used as an input to a model that has roughly 300000 neurons and model parameters to label a single pixel. This process has to be repeated

Automatic Seismic Interpretation Techniques Project Schedule

Gantt Chart Template © 2006-2018 by Vertex42.com.

Ikon Science

Project Start Date: 6/11/2019 (Tuesday) | Display Week: 1 | Project Lead: Ehsan Naeini

WBS	TASK	LEAD	START	END	DAYS	% DONE	WORK DAYS
1	**Preparatory Work**		Tue 6/11/19	Fri 6/14/19		100%	4
1.1	Familiarization with the RokDok software environment		Tue 6/11/19	Wed 6/12/19	2	100%	2
1.2	Familiarization with the project/Literature review		Thu 6/13/19	Fri 6/14/19	2	100%	2
2	**Sliding Window Approach**		Mon 6/17/19	Sun 7/07/19		60%	15
2.1	Study, run and analyze Lukas Mossers implenetation [6].		Mon 6/17/19	Wed 6/19/19	3	100%	3
2.2	Adapt the code for the Forties seismic dataset.		Thu 6/20/19	Fri 6/21/19	2	100%	2
2.3	Do hyper-parameter tuning, data augmentation to improve accuracy.		Mon 6/24/19	Sun 7/07/19	14	50%	10
2.4	Implement mirroring padding to label all of the seismic inline slice.		Mon 6/24/19	Sun 7/07/19	14	0%	10
3	**FCN approach**		Mon 7/08/19	Fri 8/02/19			20
3.1	Find a auto-encoder or U-Net implenentation to segment 2D images.		Mon 7/08/19	Mon 7/08/19	1	0%	1
3.2	Study, run and analyze this implenetation.		Tue 7/09/19	Fri 7/12/19	4	0%	4
3.3	Adapt this implimentation to work with the Forties seismic dataset.		Mon 7/15/19	Fri 7/19/19	5	0%	5
3.4	Do hyper-parameter tuning, data augmentation to improve accuracy.		Mon 7/22/19	Fri 7/26/19	5	0%	5
3.5	Analyze the performance in terms of prediction accuracy and time. Compare with sliding window approach.		Mon 7/29/19	Fri 8/02/19	5	0%	5
4	**Investigations**		Mon 8/05/19	Fri 8/09/19			5
4.1	Apply FCN to seismic volume in different slicing directions. Compare the predictions for the seismic volume.		Mon 8/05/19	Wed 8/07/19	3	0%	3
4.2	Investigate model averaging techniques to improve the overall accuracy.		Thu 8/08/19	Fri 8/09/19	2	0%	2
5	**Incorporate into RokDoc**		Mon 8/12/19	Fri 8/16/19			5
5.1	Incorporate the best method into RokDoc (including a user interface)		Mon 8/12/19	Fri 8/16/19	5	0%	5
6	**Final Report and slides**		Mon 8/19/19	Tue 9/03/19			12
6.1	Write the final report		Mon 8/19/19	Sat 8/31/19	12	0%	10
6.2	Prepare power point slides for presentation		Sun 9/01/19	Tue 9/03/19	3	0%	2

Figure 8: An outline of our project plan for the following ten weeks.

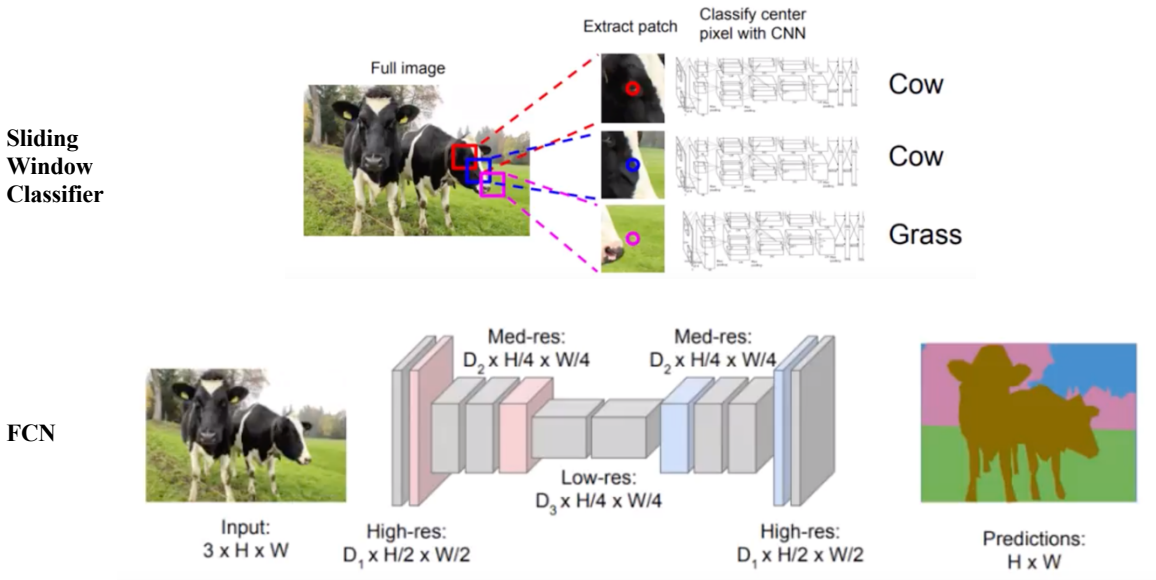

Figure 9: A comparison of the FCN and the SWC approaches. SWCs extract patches for each pixel and labels each patch using a CNN. FCNs on the other hand map the input image directly to a segmented image of the same size in one go.

for each pixel (100651 times) to segment a single inline slice of the Forties dataset. This is highly computationally expensive especially given the very large size of our seismic dataset which contains 12650 inline slices (i.e. 1273235150 cube patches are needed to be inputted into the trained CNN to label the entire seismic volume). Also, without any padding, this method fails to label pixels that lie close to the edges of the seismic cube (when the distance between the pixel and cube edge is less than that of the half cube patch size). For these reasons, we propose to use FCNs (fully convolutional networks) approach to image segmentation which has the advantage of taking a seismic inline slice (2D image instead of a 3D cube) as input and labelling all of the pixels in that slice in one go (compared to 100651 cube inputs required by the sliding window classifier to segment this slice). However, one disadvantage of using FCNs is that they tend to have more neurons than the CNN classifiers used in sliding window approaches. For this reason, we plan to use a convolutional autoencoder (similar to figure 9) as our FCN architecture. This is computationally more efficient due to down-sampling that minimizes the number of neurons. We aim to train our model with a minimal amount of labelled inline slices since manually labelling these tend to be a time-consuming and expensive process. For this reason, excessive amounts of data augmentation will be used and if time permits more state-of-the-art techniques such as U-Nets may be implemented. The goal is to compare the performance in terms of accuracy and classification time of different image segmentation techniques and neural network architectures as applied to the Forties dataset. Our detailed project plan for the next ten weeks is summarized in figure 8.

References

[1] Waldeland, A.U., Solberg, A.H.S.S. [2017]. Salt Classification Using Deep Learning. Conference: 79th EAGE Conference. 10.3997/2214-4609.201700918.

[2] Sabbione, J., Velis, D. [2010]. Automatic first-breaks picking: New strategies and algorithms. *Geophysics*. **75**. 10.1190/1.3463703.

[3] Yu, Y. [2011]. Automatic Horizon Picking in 3D Seismic Data Using Optical Filters and Minimum Spanning Tree. SEG Technical Program Expanded Abstracts. **30**.

[4] Admasu, F., Tönnies, K. [2005]. An Approach towards Automated Fault Interpretations in Seismic Data. Conference: Simulation und Visualisierung. 207-220.

[5] Github repository for MalenoV by Peter bolgebrygg: https://github.com/bolgebrygg/MalenoV, last accessed on 25/06/19.

[6] Github repository for MalenoV by Lukas Mosser: https://github.com/LukasMosser/asi-pytorch, last accessed on 25/06/19.

[7] Zhao, T., Jayaram, V., Roy, A. and Marfurt, K.J. [2015]. A comparison of classification techniques for seismic facies recognition. *Interpretation*, **3**(4), SAE29-SAE58.

[8] Qi, J., Lin, T., Zhao, T., Li, F. and Marfurt, K. [2016]. Semisupervised multiattribute seismic facies analysis. *Interpretation*, **4**(1), SB91 SB106.

[9] Krizhevsky, A., Sutskever, I. and Hinton, G.E. [2012]. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, 1097-1105.

[10] Waldeland, A.U., Jensen, A.C., Gelius, L.J. and Solberg, A.H.S. [2018]. Convolutional neural networks for automated seismic interpretation. *The Leading Edge*, **37**(7), 529-537.

[11] Badrinarayanan, V., Kendall, A. and Cipolla, R. (2015). Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *arXiv preprint,* arXiv:1511.00561.

[12] Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S. and Bengio, Y. [2014]. Generative adversarial nets. *Advances in neural information processing systems*, 2672-2680.

[13] Ronneberger, O., Fischer P., Brox, T., U-Net: Convolutional Networks for Biomedical Image Segmentation, *arXiv preprint* arXiv:1505.04597.

[14] Meldahl, P., Heggland, R., Bril, B. and de Groot, P. [2001]. Identifying faults and gas chimneys using multiattributes and neural networks. *The Leading Edge*, **20**(5), 474-482.