



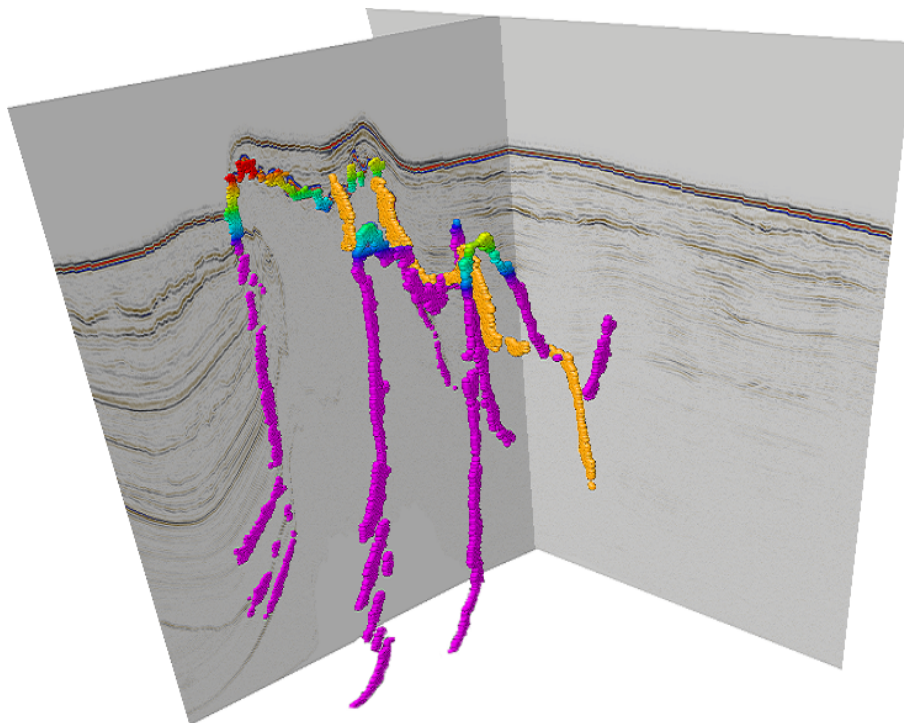
NTNU – Trondheim
Norwegian University of
Science and Technology

Schlumberger

TDT4290 - CUSTOMER DRIVEN PROJECT

NOVEMBER 11, 2016

Auto Track Sub-surface Salt Structures



Group 7

Tobias Ask
Thomas Drabløs Frøysa
Martin Mølne
Magne Skjæran
Y Nhu Camilla Tran
Ida Wold

Customer Contacts

Qiang Fu
Olesya Zimina

Advisor

Francesco Valerio Gianni

Trondheim, November 11, 2016

Tobias Ask



Signature

Thomas Drabløs Frøysa



Signature

Martin Mølne



Signature

Magne Skjæran




Signature

Y Nhu Camilla Tran



Signature

Ida Wold



Signature

Abstract

In recent years, major oil and gas discoveries have been made in deep water locations containing salt bodies in the Gulf of Mexico, offshore Brazil, and near West Africa. Salt bodies are large sub-surface concentrations of salt, and are formed by natural geological activities. In the presence of salt bodies, it is important to accurately map their shape and delineate complex salt structures in 3D to ultimately optimize drilling plans.

Salt bodies are normally discovered using seismic imaging; a technique where seismic waves are bounced off of sub-surface structures, the timing of which can be used to determine where the properties of these structures change. Delineating the shape of salt bodies can be highly difficult due to how seismic imaging interacts with the complex overhang geometries many salt bodies contain.

The team was tasked to create a solution to assist in the automated delineation of salt in the petroleum Exploration and Production software platform Petrel, as the current process is time intensive and prone to user error. The current process consists of limited automatic tracking of the top and bottom of salt bodies, combined with manual delineation.

The team created a program named NetSalt. It consisted of a convolutional neural network with supporting functionality, wrapped in a Petrel plug-in. This network could be trained on already delineated seismic data. When trained on parts of a given seismic cube, it was able to accurately delineate much of the rest of the cube, though with some false positives and negatives.

While the training time and run time of the network were both significant, the program had the major advantage of being able to delineate with minimal human intervention, thus freeing up human resources for more productive pursuits. Overall the time used was roughly comparable to the time it would take for an educated interpreter to achieve similar results.

Acknowledgements

The team wishes to thank its advisor, Francesco Valerio Gianni, for his guidance and supervision of the project. The team would also like to thank Schlumberger for the project assignment, with special thanks to the team's contacts from the company - Qiang Fu and Olesya Zimina. They were very helpful throughout the project, and provided a lot of useful feedback, while also giving the team freedom to decide how to tackle the problem. Another helpful employee from the company is Victor Aarre, who was kind enough to meet with the team for technical discussions.

Furthermore, the team is grateful for the introduction to geological concepts it received from people at the Department of Petroleum Engineering and Applied Geophysics at NTNU. Particular thanks goes to Ståle Emil Johansen for giving the team an overview of seismic imaging and interpretation, and to Dicky Harishidayat for showing the team how seismic interpretation is done in software, as well as supplying it with seismic data from the NTNU- NPD- SCHLUMBERGER PETREL READY Database.

The team would also like to thank the SEG Advanced Modeling Program (SEAM) for their public seismic data set, which was invaluable both during the development of the product and the writing of the report.

Finally, the team is grateful to the course staff of TDT4290 for everything the team got the opportunity to learn throughout this course along with its lectures.

Outline

This section contains the goal of the report and a brief description of all the sections in this report, as well as the glossary.

Goal of the report

The goal of this report is to give the reader an insight into what the team's task was, and how the team solved the problem.

Section overview

The following sections gives a brief summary of each section in the report.

Section 1: Introduction

Contains an introduction to the problem and a description of the customer, the stakeholders, and people contacted during the project.

Section 2: Planning

Describes how the project was planned including different roles, templates and standard, quality assurance, and risk analysis.

Section 3: Preliminary Study

Explains how domain knowledge was acquired by the team, relevant concepts within the domain, the planned solution, and salt delineation methods in Petrel.

Section 4: Technologies

Describes different technologies the team used during development and for the final product.

Section 5: Development Methodologies

Covers which development methodologies the team used during the project. The main content is Waterfall, Scrum-elements, and Kanban.

Section 6: Requirements

Describes which requirements, user stories, and acceptance criteria the product had.

Section 7: System Design

Contains a description of the system's architecture and quality attributes.

Section 8: Programming

Describes code standards and code review that the developers utilized while developing the product.

Section 9: Testing

Goes through how the product was tested through a variety of testing methods.

Section 10: Implementation Phase

Contains a description about what the team accomplished during the implementation phase which spanned week 38 to 44.

Section 11: Result

Describes what the team has accomplished, and what could be done in the future to improve it.

Section 12: Evaluation and Reflection

Contains the team's evaluation of and reflection on the project work, group dynamics, the customer, the advisor, and NTNU.

Glossary

Describes petroleum and computer science terms that have been used in this report.

Contents

Abstract	i
Acknowledgements	ii
Outline	iii
1 Introduction	1
1.1 Customer	1
1.2 Problem Description	1
1.3 Stakeholders	6
1.4 People Contacted During the Project	8
2 Planning	10
2.1 Project Plan	10
2.2 Project Organization	11
2.3 Communication	12
2.4 Roles	14
2.5 Templates and Standards	15
2.6 Version Control Procedures	17
2.7 Quality Assurance	20
2.8 Risk Analysis	20
3 Preliminary Study	26
3.1 Salt Bodies and Delineation	26
3.2 Business Requirements	27
3.2.1 Accuracy	27
3.2.2 Usability	27
3.2.3 Performance	27
3.3 Salt Delineation Methods in Petrel	29
3.4 Neural Networks	30
3.5 Planned Solution	38
3.6 Alternative Solutions	39
3.6.1 Bayesian attribute-based salt-detection	41
3.6.2 Edge Detection	41
3.6.3 Level Set Method	42
4 Technologies	43
4.1 Data Sets	43
4.2 Petrel	43
4.3 Ocean Software Development Framework	44
4.4 Python	44
4.5 Operating System	46
5 Development Methodologies	48

5.1	Waterfall model	48
5.2	Scrum-elements	50
5.3	Kanban	52
6	Requirements	55
6.1	Requirement - Delineate Salt Body	55
6.2	User story - as Petrel user	55
6.3	Acceptance Criteria	55
7	System Design	57
7.1	Quality Attributes	57
7.1.1	Security	57
7.1.2	Performance	58
7.1.3	Usability	59
7.1.4	Maintainability	59
7.2	Architecturally Significant Requirements	60
7.2.1	Technical Constraints	60
7.2.2	Business Constraints	61
7.2.3	Functional Requirements	61
7.3	Architecture	61
7.4	Data Flow in the System	63
7.4.1	Neural Network Training	64
7.4.2	Salt Delineation	64
8	Programming	65
8.1	Code Standards	65
8.2	Code Review	66
9	Testing	67
9.1	Testing Process Framework	67
9.2	Test Methods	68
9.3	Test plan	71
9.4	Test Case Specification	73
9.5	Test Completion Reports	73
9.5.1	Unit Tests Completion Report	73
9.5.2	Component Tests Completion Report	74
9.5.3	System Tests Completion Report	74
10	Implementation Phase	76
10.1	Milestones	76
10.2	Week 38	76
10.3	Week 39	77
10.4	Week 40	80
10.5	Week 41	84
10.6	Week 42	88
10.7	Week 43	91

10.8 Week 44	94
11 Result	96
11.1 Neural network	96
11.1.1 Quality	96
11.1.2 Performance	98
11.2 Plug-in	98
11.2.1 Mockup	98
11.2.2 Reality	99
11.3 Potential improvements	99
11.3.1 Network	101
11.3.2 Plug-in	102
12 Evaluation and Reflection	103
12.1 Project work	103
12.1.1 Research/planning	103
12.1.2 Implementation	104
12.1.3 Report	105
12.2 Group dynamics	106
12.3 Customer	107
12.4 NTNU	108
12.5 Advisor	109
References	113
Glossary	114
Appendices	116
A Meeting Minutes Templates	116
B Installation Guide	119
B.1 Prerequisites	119
B.2 Python program	119
B.2.1 Step 1 - Install Docker	119
B.2.2 Step 2 - Build Docker image	119
B.3 Petrel plug-in	120
C User manual	121
C.1 Python program	121
C.2 Petrel plug-in	123
C.2.1 Run the Neural Network for the Example-Project	124
C.2.2 Run the Network for Other Projects	127
D Test Case Specifications	129

List of Figures

1.1	A salt body highlighted with colors for the different directions	2
1.2	A complex salt-body with its delineation in blue.	3
1.3	Complex 3D-structure seen from the top	4
1.4	A cross-section with its salt delineation in green.	5
2.1	Project plan	10
2.2	Google Drive folders	17
3.1	A surface created from a salt delineation	28
3.2	Manual interpretation in Petrel	29
3.3	Seeded 3D auto tracking shown in a 3D window.	31
3.4	Seeded 3D auto tracking in 3D with an intersection included	32
3.5	Multi-Z points in an interpretation window	33
3.6	Multiple crosslines interpreted using multi-Z shown in a 3D window	34
3.7	Mesh created from the Multi-Z interpretation in Figure 3.6	35
3.8	General neural network structure	36
3.9	Overfitting of training data	37
3.10	Training the neural network	39
3.11	Running the classifier over a cross section	40
5.1	Phases of the waterfall model	49
5.2	Kanban board	54
7.1	Performance scenario	59
7.2	Maintainability scenario	60
7.3	CNN architecture	62
7.4	System development view	63
7.5	Data flow during training	64
7.6	Data flow during delineation	65
8.1	Code sample	66
8.2	Code sample	67
9.1	Cross-section with complex salt shapes	70
10.1	Early input and output	78
10.2	Early output	79
10.3	Left: Original output. Right: Post-processed output	81
10.4	NN output from single-channel input	82
10.5	NN output from downscaled single-channel input	83
10.6	NN output	85
10.7	Mockup of Petrel plug-in	87
10.8	A context-menu in Petrel	89
10.9	Dialogue box in Petrel for adding training data	90
10.10	A set of points showing the salt delineation without a cross-section	92
10.11	A cross-section with the delineation result	93
11.1	The effect of training on the neural network	97
11.2	Mockup of Petrel plug-in	100
B.1	Docker Quickstart Terminal	120
B.2	Successful Docker image build	121

C.1	A seismic cube with no data	124
C.2	The seismic cube in the input-pane of Petrel	125
C.3	The Petrel Help Center menu	128
D.1	Test cross-section from SEAM delineated	131
D.2	Simple cross-section from SEAM Interpretation Challenge Time . .	132
D.3	Complex cross-section from SEAM Interpretation Challenge Time .	133

List of Tables

1.1	Stakeholders in the project	6
1.2	Contact information for the team	6
1.3	Contact information for the customer	7
1.4	Contact information for the advisor	7
1.5	People contacted during project	8
2.1	Overview of different roles	12
2.2	Risk 1 Misunderstanding the project description	22
2.3	Risk 2 Missing training data	22
2.4	Risk 3 Technologies used takes longer to learn	23
2.5	Risk 4 Conflicting views of how to solve the problem	23
2.6	Risk 5 Not able to find suitable neural network	24
2.7	Risk 6 Technical Difficulties	24
2.8	Risk 7 Approach to the problem turns out not to work	25
2.9	Risk 8 Team member is unavailable for a short period	25
2.10	Risk 9 Team member unavailable for a long period of time	26
7.1	Assets identified during Protection Poker	57
9.1	Testing Activities and related estimates	73
9.2	Test case Template	73
D.1	Test case U01	129
D.2	Test case U02	129
D.3	Test case U03	129
D.4	Test case U04	130
D.5	Test case U05	130
D.6	Test case U06	130
D.7	Test case U07	131
D.8	Component test 01	131
D.9	Component test 02	132
D.10	System test 01	132
D.11	System test 02	133

1 Introduction

The project covered by this report was done as part of the course *TDT4290 - Customer Driven Project*. Section 1.1 describes the project's customer. Section 1.2 describes the problem. Section 1.3 lists all the stakeholders for this project. Section 1.4 lists the people the team contacted for help regarding petroleum related topics and machine learning. Technical terms used in this report can be found in the glossary.

1.1 Customer

The customer for the project was Schlumberger. Schlumberger is the world's largest provider of services for petroleum exploration and production [40]. The team primarily interacted with the customer's two representatives Qiang Fu and Olesya Zimina.

1.2 Problem Description

Sub surface salt bodies can indicate the presence of petroleum products. See Figure 1.1 for an example of a salt body. To ease the exploration of surrounding areas, it is of interest to identify salt bodies. This is done by delineating their edges in seismic data. The seismic data used in this project originate from three different data sets, see Section 4.1. Of these, the team most frequently used the data set SEAM, which is made of synthetic data. It contains less noise than the results from real seismic surveys.

Delineating salt bodies can be difficult because of their complex shapes, see Figure 1.2 and Figure 1.3, and the fact that the edge may pass the same vertical line multiple times. In conventional seismic interpretation the layers surveyed do not cross the same X, Y coordinates multiple times, which makes interpreting them simpler. The methods used for this are used for the top and bottom of salt bodies, as they normally behave in this fashion. The flanks of the salt on the other hand will often pass under the same surface position multiple times, and is often surrounded by noise, which further complicates the delineation process.

The challenge this project aimed to address was to be able to automatically delineate salt structures in seismic data, including their flanks. Today the delineation of the flanks is done manually. This is a time consuming process where an interpreter goes through numerous seismic cross-sections and attempts to identify the edge of the salt body based on previous experience. See Figure 1.4 for an illustration of manual interpretation of a single cross-section.

Due to the complex nature of salt bodies, the solution had to be flexible enough to accurately describe the shape of these structures. In addition, it was important

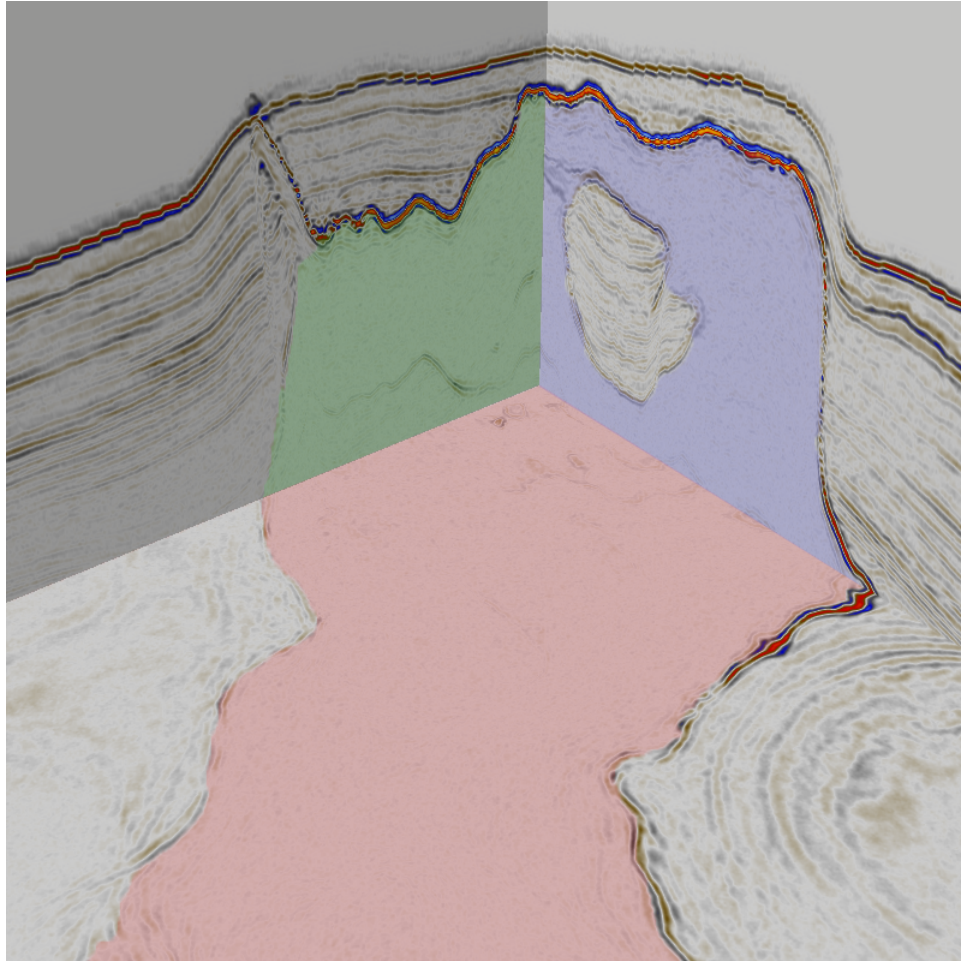


Figure 1.1: The colored parts of this figure is the salt body. The salt in the inline-direction of the seismic image is highlighted in green. The salt in the crossline-direction is highlighted in blue. The salt in the timeslice-direction is highlighted in red.

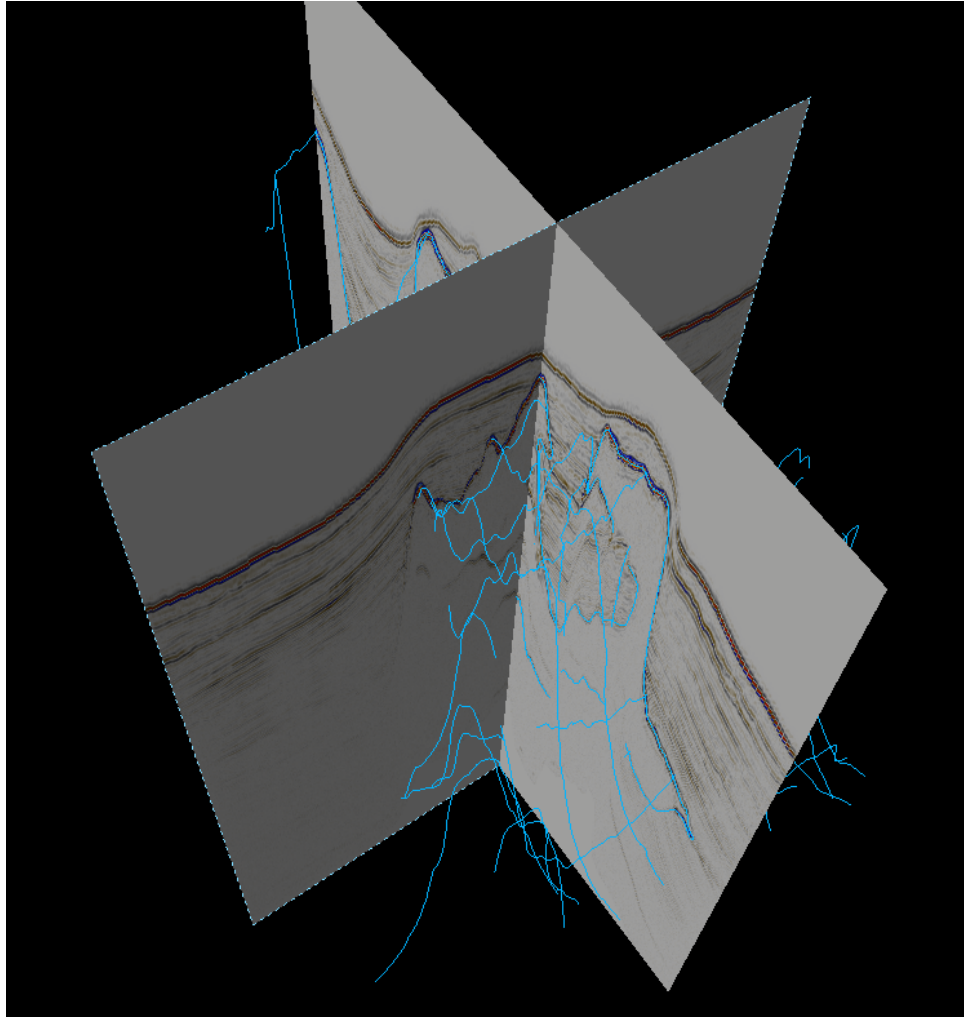


Figure 1.2: A complex salt-body with its delineation in blue.

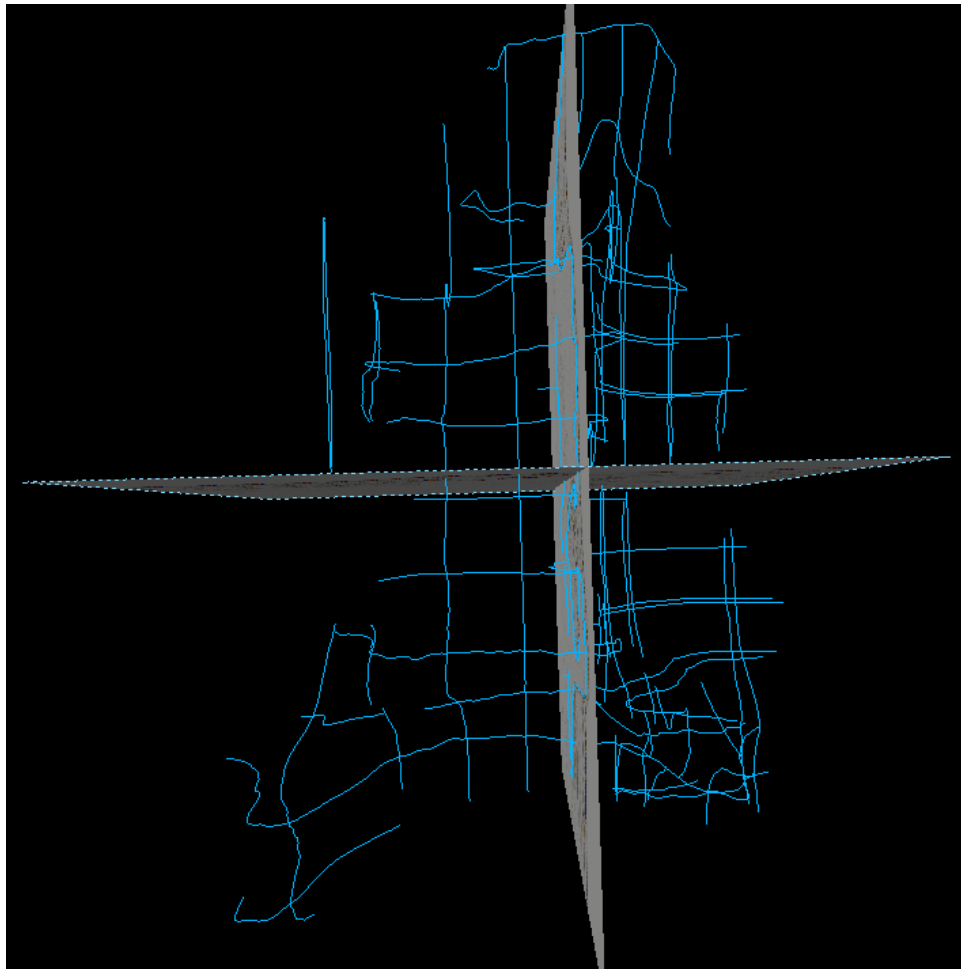


Figure 1.3: The same complex salt-body as in Figure 1.2, with its delineation in blue. Seen from the top to show the complex 3D-structure.

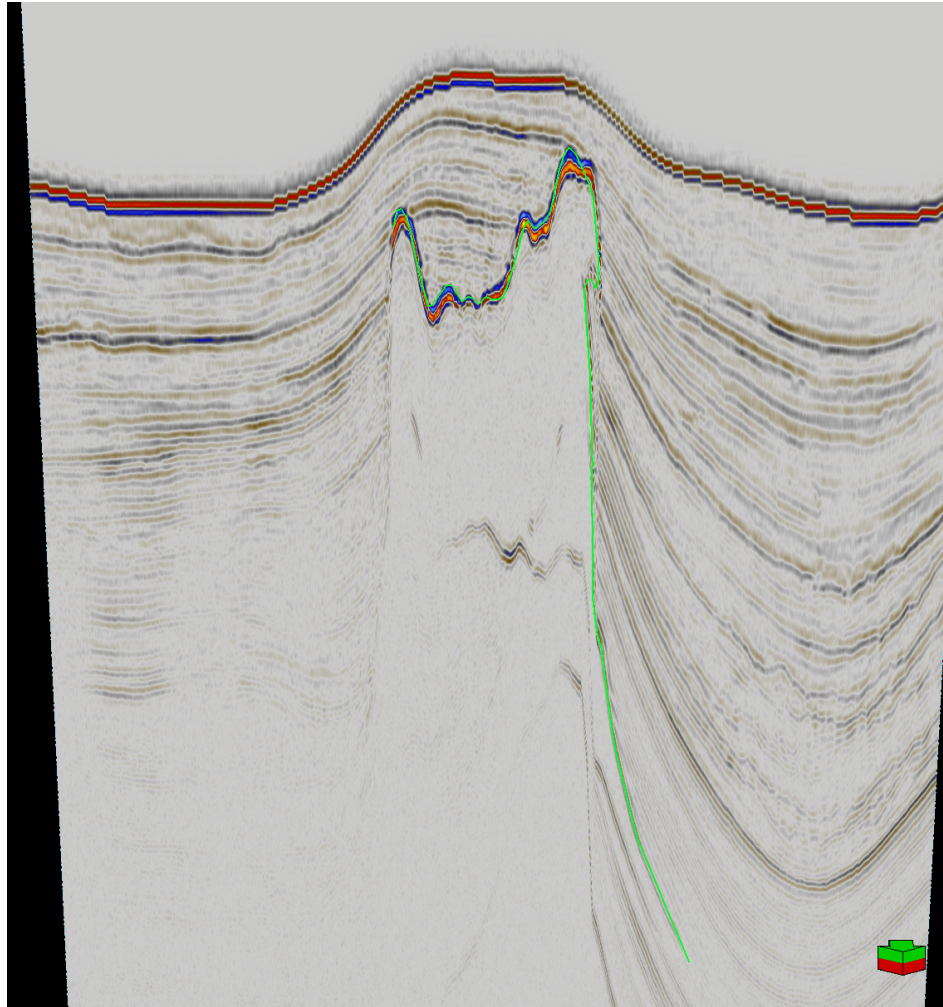


Figure 1.4: A cross-section with its salt delineation in green.

that the new approach was efficient so it could reduce the cost of human efforts in the delineation process.

1.3 Stakeholders

A stakeholder is any person or entity that has interest in a project. This is not limited to stakeholders directly involved in the project. There were several stakeholders for this project, both directly and indirectly involved. They are listed in Table 1.1, and described in more detail below.

Stakeholder	Reason for interest
The team	Graded by resulting product
Customer	Wants a solution to the given problem
Advisor	Interested in good internal and external communication
IDI staff	The project represents NTNU
Examinators	Grades the final product

Table 1.1: Stakeholders in the project

The Team

The team for the project consisted of six students, studying computer science and informatics. Their names, emails, and study programs can be found in Table 1.2. The team's main involvement in the project was developing the solution based on the customer's requirements, writing the project report, and holding a presentation. The grade in the subject being dependent on the resulting report and presentation further reinforces their stake in the project.

Name	Email address	Study program
Tobias Ask	tobiaas@stud.ntnu.no	Computer Science
Thomas Drabløs Frøysa	thomasdf@stud.ntnu.no	Computer Science
Martin Mølne	martmol@stud.ntnu.no	Computer Science
Magne Skjæran	magneskj@stud.ntnu.no	Computer Science
Y Nhu Camilla Tran	yntran@stud.ntnu.no	Informatics
Ida Wold	idawol@stud.ntnu.no	Computer Science

Table 1.2: Contact information for the team

The Customer Representatives

Other than the team, the most important stakeholders in the project were the two representatives from the customer Schlumberger. Their contact information can

be found in Table 1.3. They had an interest in the end result for possible future use in their products. The representatives were the main point of contact between the team and the company. Qiang Fu primarily handled topics related to software development, while Olesya Zimina handled administrative matters and anything relating to the seismic interpretation workflow.

Name	Email address
Qiang Fu	QFu3@slb.com
Olesya Zimina	Ozimina@slb.com

Table 1.3: Contact information for the customer

The Advisor

Francesco Valerio Gianni was the advisor for the team. His contact information can be found in Table 1.4. Gianni's role was to keep an eye on the project's progress, and making sure communicating within the team, or between the team and the customer, did not break down.

Name	Email address
Francesco Valerio Gianni	francesco.gianni@idi.ntnu.no

Table 1.4: Contact information for the advisor

IDI Staff



The course was organized, and the project chosen, by staff at IDI, the NTNU Department of Computer and Information Science. They were interested in how the team represents the university, making them stakeholders.

Examiners

The examiners for the project are responsible for evaluating and grading the project, and are therefore interested in the work done in the project.

Name	Date of first contact	Place or way of contact	Email address
Knut Reitan Backe	26.08.16	Email	knut.reitan.backe@ntnu.no
Igor Barros Barbosa	29.08.16	Email	igorbb@idi.ntnu.no
Aleksander Rognhaugen	30.08.16	Email	aleksaro@idi.ntnu.no
Theoharis Theoharis	30.08.16	Email	theotheo@idi.ntnu.no
Frank Lindseth	30.08.16	Email	frankl@idi.ntnu.no
Ståle Emil Johansen	31.08.16	Petroleum Institute	stale.johansen@ntnu.no
Dicky Harishidayat	06.09.16	Petroleum Institute	dicky.hidayat@ntnu.no
Mathias Bellout	06.09.16	Petroleum Institute	mathias.bellout@ntnu.no
Victor Aarre	08.09.16	NTNU Gløshaugen	vaarre@slb.com
Erlend Våtevik	19.09.16	Email	erlend.vatevik@ntnu.no
Nader Salman	12.10.16	Skype for Business	nsalman@slb.com
Liyuan Xing	12.10.16	Email	liyuan.xing@idi.ntnu.no

Table 1.5: People contacted during project

1.4 People Contacted During the Project

The team was in contact with several people during the project. Their contact information can be found in Table 1.5. Broadly the reasons for contacting these people fell into two categories: acquiring a better understanding of machine learning, and a better understanding of petroleum related technology and terminology.

Petroleum

At the start of the project the team lacked any but the most cursory experience with the petroleum industry and geoscience, both of which were central to the project. The team were therefore in contact with a number of people to better understand different aspects of this domain.

Early on the team contacted Ståle Emil Johansen, a professor at NTNU’s Department of Petroleum Engineering and Applied Geophysics (hereafter, the ”Petroleum Institute”), to get a basic understanding of seismic imaging in relation to the petroleum industry, and acquire further contacts so as to learn how to use Petrel and learn more about the domain. Johansen put the team into contact with Dicky Harishidayat. Harishidayat is a PhD student at the Petroleum Institute, and gave the team an introduction to Petrel and the delineation of salt bodies, and put the team into contact with Mathias Bellout. Bellout is a Postdoctoral Fellow at the Petroleum Institute, who provided the team with some pointers to relevant research.

The team also had a meeting with Victor Aarre. Aarre is an employee at Schlumberger not directly involved in the project, but had extensive knowledge about neural networks, geology, and previous attempted methods for salt delineation. Aarre

gave feedback and advice on the progress of their project.

Knut Reitan Backe was successfully contacted to get a hold of licenses for the Petrel software. Erlend Våtevik was successfully contacted to acquire licenses for the Ocean Framework.

Machine learning

The team also contacted several people with knowledge about neural networks and image recognition. These were Frank Lindseth, Theoharis Theoharis, Aleksander Rognhaugen, and Igor Barros Barbosa. Unfortunately, none of them had the time to assist the team.

Via the customer, the team was also in contact with Nader Salman, another Schlumberger employee. Salman was involved in another project at Schlumberger working with tracking of subsurface structures. Salman gave the team some tips as to the structure of the neural network, and how to best test and validate it.

Liyuan Xing is a postdoc from IDI working with Schlumberger, contacted by the team for possible advice or input for the project. Xing did not have the time to assist the team.

PROJECT OVERVIEW														
Phase	Month	August			September			October			November			
	Week	34	35	36	37	38	39	40	41	42	43	44	45	46
Research		Yellow	Yellow	Yellow	Yellow	Yellow	Grey	Grey	Grey	Grey	Grey	Grey	Grey	Grey
Implementation		Grey	Grey	Grey	Grey	Yellow	Yellow	Yellow	Yellow	Yellow	Yellow	Yellow	Grey	Grey
Testing and Finalization		Grey	Grey	Grey	Grey	Grey	Grey	Grey	Grey	Grey	Grey	Grey	Yellow	Yellow
Presentation		Grey	Grey	Grey	Grey	Grey	Grey	Grey	Grey	Grey	Grey	Grey	Grey	Yellow

Figure 2.1: Project plan

2 Planning

Planning can in many cases make or break a project; a good plan is a boon to any project, while a bad one may well bring a project down. Section 2.1 shows the overall project plan. Section 2.2 covers how the project was organized. Section 2.3 explains the team’s communication methods. Section 2.4 lists all the roles and who is assigned that role. Section 2.5 talks about the templates and standards used in this project. Section 2.6 explains the team’s procedures for version control. Section 2.7 describes how the team ensured a high quality of work throughout the project. Section 2.8 presents an analysis of the risks involved in the project.

2.1 Project Plan

The project plan was as described in Figure 2.1. This project plan was made at the start of the project and shows a high-level overview of the different phases and tasks involved. It was updated a few times during the project based on its progress.

The report was written over time throughout all phases except the presentation phase. The effort spent on the report gradually ramped up over time. Especially in the research phase, not much was done beyond devising the overall structure of the report, as the research itself was considered more important.

The first phase of the project was the research phase. The team did not have any prior knowledge about petroleum or seismic data. There was also new software to be learned, such as Petrel and Ocean. It was therefore decided to spend four weeks on research. Additionally there was a transitional phase between the research and implementation phases. This was not in the original plan, but was added when it

was discovered that the team was not completely ready to start implementing the solution yet at the end of the research phase. The research phase included learning about petroleum and seismic data, Petrel, neural networks, and already existing solutions for automatically delineating salt bodies. More information about the research done can be found in Section 3.

The implementation phase started after the research phase was finished. The deadline for the implementation was initially set to the 26th of October, but was moved to the 2nd of November near the end of the phase. This was done so as to give time to achieve interaction between the Python and Petrel halves of the solution, but this turned out to not be feasible even within the extra time. During these 7 weeks the neural network was set up, trained, and improved. Beyond that, functionality for importing seismic data to the network was created, image saving added, rudimentary post-processing of the output set up, and the ability to write the output in a format readable by the Petrel plug-in added. A Petrel plug-in was created with the ability to import the neural network output visually into Petrel, and export seismic data in a format readable by the Python program. More about the implementation phase can be found in Section 10.

The phase after the implementation phase was used for testing the product and finalizing the report. Work done in this phase included executing component and system tests. The test procedures can be read about in Section 9. On the report side of things, all remaining sections were completed and the report proofread and edited.

The last phase was the presentation phase. Work on the presentation, beyond rudimentary planning, started after the report was delivered and lasted until the presentation on Thursday in week 46. In this time the whole presentation was made and rehearsed.

2.2 Project Organization

Project organization is important for ensuring everyone knows what to do, and maintaining a high quality of work. The steps the team took to achieve structure included giving every team member one or more role, and having regular meetings every week. Table 2.1 is an overview of all the team members' roles. Detailed descriptions of each role can be found in section 2.4. There were three regular meetings in the week, including customer and advisor meetings. The following sections are about the different types of meetings.

Meetings

The team had three meetings each week, all of which were attended by all members of the team except when illness or other obligations prevented individual members from attending. Only one of these meetings (Tuesdays at 13:15) were purely

Roles	Team member
Project Leader	Y Nhu Camilla Tran
Software Architect and Developer	Tobias Ask
Test Manager	Martin Mølne
ScrumMaster and Developer	Magne Skjæran
Plug-in Developer and Secretary	Thomas Drabløs Frøysa
Report Manager	Ida Wold

Table 2.1: Overview of different roles

internal, but after each customer and advisor meeting the focus would switch to internal matters. The internal meetings had a fixed meeting template consisting of Scrum standup meetings, report status, code status, and delegation, with a section for anything that did not fit in these categories. More about meeting agenda and Scrum standup meetings can be found in section 2.5 and 5.2.

Customer Meetings

The customer meetings were with only a few exceptions held on Wednesdays at 12:15. In the few cases this was not possible, the meeting was moved to another day. During each customer meeting, the team would present what had been done since the previous customer meeting. The customer would then provide tips, feedback, and comments to help direct the team's efforts in the following week, as well as answering any questions the team might have.

The stakeholders that joined these meetings was primarily the team's Schlumberger contact persons, Qiang Fu and Olesya Zimina. Other engineers and geologists were sometimes invited to join the meeting to give the team feedback. An overview of who the team was in contact with can be found in Table 1.5.

Advisor Meetings

The advisor meetings were held on Fridays at 10:15. The advisor checked the team's progress and issues, and gave feedback for the work and tips for improvements.

2.3 Communication

Communication is very important in a team and is one of the keys to a good group dynamic and making progress in a project. There is a lot of different communication technology on the market today, and the project team chose to use Slack, *Skype for Business*, and *appear.in*. The next sections are about these technologies and how the team used them.

Slack



Source: <https://brandfolder.com/slack>

Slack is an app for messaging for teams. It allows the team to create different channels for individual topics, as well as communicating privately with individual team members. More information about Slack can be found at slack.com [2]. The team used Slack as its primary internal communication channel, and had six different channels: code, meetings, report, research, general, and random.

By splitting topics into different channels the team ensured that conversations could stay focused. The report channel was used to communicate about report writing, and the meetings channel was used to report about absence or meeting rooms. The developers used the code channel to communicate, and the research channel was used for posting articles and other research material to achieve more domain knowledge in the team. Anything project related that did not fit these categories was posted in general. Everything that had nothing to do with the project was posted in random, ensuring the project-related channels stayed on topic.

Skype for Business and appear.in



Source: <http://www.fusecollaboration.com/technologies/skype-for-business>

Skype for Business was used primarily to communicate with the customer. An invitation link was needed for joining the conference, and an user account for using the program. The team also used appear.in, a website where it is possible to have video conversations, in cases where members of the team could not physically

attend a meeting. More about Skype for Business and appear.in can be found at skype.com[1] and appear.in [3].

2.4 Roles

Roles in a project are important so that every team member can feel they have responsibility and have a clear way to contribute to the project. The coming sections are about each of the roles in this team and which team member held the role. Table 2.1 is an overview of the different roles.

Thomas Frøysa volunteered to be the team's software architect and wanted to use this opportunity to learn more about software architecture. Because of the short duration of the project, after a while Frøysa came to the conclusion that he would not be able learn enough in time and therefore resigned from this role. Tobias Ask was, based on their previous experience with software architecture, elected to take over the role and continue Frøysa's work. Frøysa took over Ask's extra role of being the secretary in addition to being a developer in the team. The switch took place in week 41. In the upcoming sections, members who formerly held a given role are highlighted with italics.

Project Leader: Y Nhu Camilla Tran

The project leader is responsible for making sure that all team members have something to do, and managing the team. Delegating tasks and making sure that they are being conducted are some of the main tasks. In the team, the project leader led all of the group meetings by following a specific meeting agenda. Administrative work such as sending mail to the customer, setting up different documents in Google spreadsheets and Google docs were also largely handled by the project leader.

Software Architect: Tobias Ask, *Thomas Drabløs Frøysa*

The main task for the software architect is to plan how the system is going to be structured. The planning was based on customer meetings where requirements were discussed. The architect had to interpret the requirements and design the system based on this.

Test Manager: Martin Mølnå

The test manager is responsible for making the test plan, making sure that testing gets conducted, and analysing the results.

ScrumMaster: Magne Skjæran

The ScrumMaster held the Scrum stand-up meetings in the project. More about the ScrumMaster and Scrum stand-up meetings can be found in section 5.2.

Developers: Magne Skjæran, Tobias Ask

A developer's task is to make sure that the architect's vision gets implemented. The system has to be programmed, and deadlines upheld. The developer is responsible for delivering code in time for the customer meetings. Another important task is to make sure that the code has a minimal amount of bugs.

Plug-in Developer: Thomas Drabløs Frøysa

The main task of the plug-in developer is to implement the plug-in using the Ocean framework. It is important to have good communication with the developers and the customer to achieve a good result. More information about the Ocean tool can be found in section 4.3.

Report Manager: Ida Wold

The team needed a report manager to make sure that the report stayed on schedule. The main tasks of the report manager was to delegate report writing tasks and keeping track of the report's status. This ensured that the report would not leave out any important sections or pieces of information.

Secretary: Thomas Drabløs Frøysa, Tobias Ask

The secretary's task was to write down everything that was said during the meetings. It was an advantage to have a permanent secretary so that the language and layout in the report meeting documents was consistent.

2.5 Templates and Standards

Templates and standards are important for the team. It improves the team's work environment and introduces structure to the meetings. The coming sections are about the templates and standards the team used.

Meeting Minutes Templates

The team's meeting minutes were written down by the secretary every meeting. The meeting minutes had a fixed template for each type of meeting and can be read in Appendix A. Between week 34 to 38 there were no fixed templates, thus the meeting minutes were written down arbitrarily rather than being based on a pre-planned meeting agenda. At the end of the research phase and the start of the implementation phase, the team wanted to have more effective meetings, and meeting minutes that would be easier to refer back to in the future. The solution was to have fixed templates. The use of the templates started during week 39 with trial and error, but the fixed templates did not reach their final form until week 40.

The fixed templates were different for each type of meeting, but would contain the same main topics each week. A topic was simply erased if there was nothing to cover related to it that meeting. As an example the team did not always have questions for Schlumberger or the advisor. The internal meeting minute topics were Scrum stand-up meetings, report and code status and delegation, and an "optionally" section. The "optionally" section was used to discuss other topics that did not fit under the other topics, such as planning the presentation. The team members were free to write down their own topics in this section before the meetings.

The customer and advisor meeting minutes templates consisted of the team's status and feedback. There were sections that were named "during the meeting". After each customer and advisor meeting there was an internal meeting where the template from purely internal meetings was reused. In addition, the team discussed the customer and advisor meeting after it was done.

Google Drive structure

Google Drive[12] is an online cloud storage solution and gives users the ability to store documents and create Google Documents, Spreadsheets and more. These can all be edited collaboratively by multiple people at the same time.

The folder structure and file names in the team's Google Drive were set up so as to ease navigation. Figure 2.2 shows the main folders in Google Drive. "Administrative files" contained documents such as contact info and presentations, "Images" contained images of the output from the neural network and mock-ups, and "Report" contained report related documents such as draft-reviews.

The "Software files" folder contained files that the team received from Schlumberger, and the "Meetings" folder contained all of the meeting minutes, further subdivided by week. The "Autotrack Salt Body Project" folder was a common folder accessible by both the team and Schlumberger. It contained requirements, acceptance criteria, hours per person, risk analysis, and the architecture. Section 2.6 explains the version control procedures for the team's Google Drive.

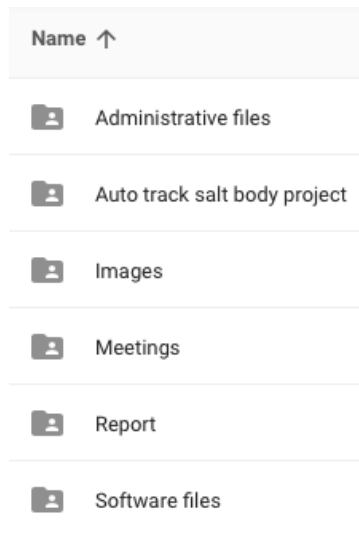


Figure 2.2: Google Drive folders

Absence and Tardiness

Every team member is an important resource for the project and therefore it is important to notify the rest of the team of absence or tardiness before meetings so that it can be compensated. The team had effectively no issues with absence; unplanned absence never occurred, and in the cases where team members were prevented from being physically present they still participated via Skype for Business or appear.in. If a team member was tardy, they would alert the rest of the team via Slack. More about Skype for Business, appear.in, and Slack can be found in Section 2.3.

Hour Tracking

It is important to write down working hours to see the workload in the project and ensure it is fairly distributed. Each team member wrote down the hours they worked in a Google Spreadsheet. The document contained sheets for each individual team member and a summary of the team as a whole.

2.6 Version Control Procedures

During the project, version control was key to ensure that no work would be lost due to hardware failure, user error, or similar. To facilitate this, a number of tools were used, each with associated procedures to ensure their proper functioning.

Google Drive



Source: <https://developers.google.com/drive/v3/web/branding>

Most project documents were stored on Google Drive, where any change is near immediately backed up to Google's servers. Google Drive also provided version history for all such documents, allowing the easy restoration of previous versions at any time. Due to the automated nature of Google Drive, it did not require any particular procedures. For more information about Google Drive, see Section 2.5.

GitHub



Source: <https://github.com/logos>

The code for the project was stored on GitHub. This allowed changes to be bundled together into commits, making it easier to reason about what members of the team had been doing. For code this was more effective than the version history tools like Google Drive provide. This was important so that a high standard of code could be enforced; knowledge of the ongoing changes was more important here than for other documents. The procedures for the use of GitHub for the code were as follows:

- Commit changes whenever an atomic task has been completed so as to ensure minimal code is lost due to unforeseen circumstances
- Sync changes to the GitHub servers as often as possible
- Use clear commit messages to ensure a common understanding of the code's progress

- Avoid the use of branches, as this can increase overhead

The project's backlogs were also stored on GitHub using GitHub's recent "projects" feature. Research, report, and implementation backlogs were stored separately. However, it is worth noting that unlike the code storage on GitHub, this did not actually provide any sort of version history. As such members of the team were not to delete any items from the backlogs without first consulting other members, as that would be difficult to reverse.

ShareLaTeX

ShareLaTeX



Source: <http://2014.ruleml.org/images/ShareLaTeX.png>

Finally, the project report itself was stored on ShareLaTeX. ShareLaTeX[34] is a website allowing for collaborative editing of \LaTeX files without the need for installing any software. It allowed for collaborative, simultaneous editing, while also providing a secure storage location and version history. Much like Google Drive, this was automated. Due to the sheer importance of the report, it was also synchronized to one of the team member's Dropbox[10] account so that if ShareLaTeX should somehow fail, the team would still have a copy of the report.

Overall

The overarching procedure beyond the tools themselves was that all artifacts with continuing relevance to the project must be stored on one of the three tools; nothing should be stored solely on any team member's computer except very temporarily (e.g., between Git commits). Data generated from version controlled data (e.g., neural network output images) was exempted from this, as that could be regenerated as needed. This all happened automatically once set up, thus requiring no particular procedures.

2.7 Quality Assurance

The purpose of quality assurance was to ensure the work conducted in the project was of a high quality. The focus for quality assurance is the process rather than the product itself. By deciding on common standards for meetings, documentation, and implementation of the program, it was easier to make sure the work being done was beneficial for the end product. This section covers most of the measures implemented to ensure good quality, and includes both internal and external routines. Routines concerning coding can be found in Section 8, testing in Section 9, and version control in the previous subsection. More information about meeting templates can be found in Section 2.5.

Meetings

The purpose of the meetings was to evaluate progress, make decisions concerning the entire group, and plan the work ahead. These meetings are discussed in more detail in Section 2.2.

Advisor Meetings

The team had weekly meetings with the advisor, to keep him up to date on the progress of the project. The meetings were also used for any questions from the group, and feedback from the advisor to the group.

Customer Meetings

There were weekly Skype for Business meetings between the team and the company's representatives. This meeting fulfilled two important purposes. It let the customer know about the progress being done, and the team could get input on the way forward.

Work Review

Throughout the project a lot of documents were made, research conducted, and people contacted. To ensure these activities were done in a satisfactory manner, documentation and review were important. Any correspondence between the team and external sources was stored to make sure it would be easy to check on the previous work conducted. All documents created for external use were written as a collaborative effort, or reviewed by team members other than the original author, to ensure quality.

2.8 Risk Analysis

It is important to know all the risks that could impact the project, especially those which might affect it to such a degree that the customer is not satisfied with the delivery. The risk analysis include plans for preventing risks. These plans lower

the probability or consequence of this risk. There are also plans for mitigating risks; reducing their impact once they have occurred. Each task is assigned to the whole group or to the person with a specific role. The person assigned to a task is responsible for checking if the risk has become an issue. If it has, this person is responsible for undertaking tasks to mitigate the risk within the given deadline or determining if it is best to ignore it. This person also makes sure that the preventative actions are completed. The activities affected by these risks are also listed in the tables.

All the risks have a probability of becoming an issue. The probabilities are divided into:

- Low - This is not very likely to happen. In most projects it will not happen.
- Medium - This is risks that are moderately likely to happen, but usually only once.
- High - Risks that are very likely to happen. It is also a probable that they could occur more than once.

The consequences for each risk are divided into:

- Low - The risk will not impact the project much.
- Medium - It is a high probability that the project could be delivered after schedule or be missing some features.
- High - This risk could make the delivered project very different from what the customer expected. This can also include a solution that is not finished.

The identified risks for this project can be found in Tables 2.2-2.10.

ID	R1 Misunderstanding the project description
Description	The project description can be misunderstood and the delivered solution can be very different from what the customer wanted
Activity	All
Probability	Medium
Consequence	High
Prevention of the risk	To prevent this from becoming an issue it is important to have honest and clear communication with the customer. It is also important to do solid research on all subjects and questions that come up in the project description or the customer meetings. The customer can also give additional information about difficult subjects, like a presentation or demonstration.
Mitigation of the risk	If it is clear that there is a misunderstanding between the customer and the team he team can try to change the delivery so it satisfy the customers expectation, or ask for additional information so as to resolve it.
Deadline	Continuous
Responsibility	All

Table 2.2: Risk 1 Misunderstanding the project description

ID	R2 Missing training data
Description	The neural network needs already delineated seismic cubes for training the network.
Activity	Training the neural network
Probability	Medium
Consequence	High
Prevention of the risk	It is important that the team ask the customer about receiving delineated seismic cubes for training the neural network
Mitigation of the risk	If the team cannot get a hold of delineated seismic cubes, interpreting their own data is a possibility. This is very time consuming and the team do not have enough training to delineate salt bodies very accurately.
Deadline	One week after it has become an issue
Responsibility	All

Table 2.3: Risk 2 Missing training data

ID	R3 Technologies used takes longer to learn
Description	Many of the technologies used in this project are new to the team. Estimating how much time it takes to learn these new technologies can be difficult. The estimates might differ significantly from the actual time spent.
Activity	Planning and implementation
Probability	High
Consequence	Medium
Prevention of the risk	It is important to research all the technologies that are used in the project well, so that the estimated time to learn these technologies are as close to correct as possible.
Mitigation of the risk	If it looks like the time to learn the technology was significantly underestimated, the team member that understand this technology best should explain it to the rest. Alternatively it is also a possibility to ask the customer for additional information if it is developed by Schlumberger. Reducing the scope is also a possibility if there is limited time left.
Deadline	At next group meeting(max 4 days) after it has become an issue
Responsibility	Team leader

Table 2.4: Risk 3 Technologies used takes longer to learn

ID	R4 Conflicting views of how to solve the problem
Description	Members of the team can have different views of how to solve the problem. This can lead to using a lot of time to agree on one solution or team members can use slightly different solutions on different tasks.
Activity	All
Probability	Medium
Consequence	Medium
Prevention of the risk	It is important to try to build consensus in the early phases of the project.
Mitigation of the risk	If it is impossible to get a consensus a vote should be held to decide the path forward. The team can also get additional information from the customer before the vote.
Deadline	Start of implementation phase
Responsibility	Team leader

Table 2.5: Risk 4 Conflicting views of how to solve the problem

ID	R5 Not able to find suitable neural network framework
Description	There are many neural networks frameworks available. The team has to find a neural network that is suitable for the project and has a satisfactory licence-agreement.
Activity	Planning and implementation
Probability	Low
Consequence	Medium
Prevention of the risk	The team should search extensively for fitting software.
Mitigation of the risk	If the team cannot find a suitable neural network framework the team can write their own or change to another solution.
Deadline	Start of implementation phase
Responsibility	All

Table 2.6: Risk 5 Not able to find suitable neural network

ID	R6 Technical Difficulties
Description	The team might encounter problems with different technologies. Internet problems can affect writing documents on both Google Drive and ShareLaTeX. The team is also dependant on an Internet connection for starting Petrel. Communication with other members and the customer often happen with programs like email, Skype for Business, Slack, and Appear.in which all need an Internet connection. The team is also very dependent on computers for programming, writing documents and some communication.
Activity	All
Probability	High
Consequence	Low
Prevention of the risk	The team should make sure they have access to multiple computers and Internet connections.
Mitigation of the risk	All the members of the team own at least one computer. In addition they have access to computers at NTNU. At the petroleum institute the team also have access to computers with Petrel and Ocean installed. All members of the team have access to Internet at home and at NTNU.
Deadline	Continuous
Responsibility	All

Table 2.7: Risk 6 Technical Difficulties

ID	R7 Approach to the problem turns out not to work
Description	It is a possibility that the chosen approach to solve the problem does not work as well as the team and the customer wants.
Activity	All
Probability	Medium
Consequence	High
Prevention of the risk	Attempt to create a basic proof of concept as early in the implementation phase as possible. It is also important to have alternate approaches in case the first approach do not work well enough.
Mitigation of the risk	If the team find out that the solution do not satisfy the customer's expectations the team will change their approach if there is still time to implement and deliver an alternative solution, but the scope might be reduced.
Deadline	Continuous
Responsibility	All

Table 2.8: Risk 7 Approach to the problem turns out not to work

ID	R8 Team member is unavailable for a short period
Description	Members of the team is temporarily unavailable for a short period of time (illness, obligations, etc.). This can lead to small delays of important work
Activity	All
Probability	High
Consequence	Low
Prevention of the risk	The team should ensure that there's redundancy in skill sets so the rest of the group can handle their tasks
Mitigation of the risk	It can become necessary to reassign tasks that are important or should be finished soon to other team members. Can also consider reducing the scope.
Deadline	Continuous
Responsibility	All

Table 2.9: Risk 8 Team member is unavailable for a short period

ID	R9 Team member unavailable for a long period
Description	Members of the team is temporarily unavailable for a long period of time (serious illness, major events, etc.). This can lead to large delays of important work
Activity	All
Probability	Low
Consequence	High
Prevention of the risk	The team should ensure that there's redundancy in skill sets so the rest of the group can handle their tasks
Mitigation of the risk	Reassign tasks to other team members. Reducing the scope needs to be seriously considered.
Deadline	Continuous
Responsibility	All

Table 2.10: Risk 9 Team member unavailable for a long period of time

3 Preliminary Study

This section documents the team's findings from the research phase of the project. Since the project assignment requires technical knowledge of the geological and seismic domain, the team spent substantial effort trying to acquire this knowledge. Section 3.1 captures the essence of the problem. Section 3.2 talks about what the customer wanted from the team's product in terms of quality. Section 3.3 explains the most common methods used to tackle the problem today. Section 3.4 gives a theoretical overview of neural networks, a critical concept in the solution. Section 3.5 describes the team's solution in a conceptual manner. Section 3.6 looks at alternate solutions to the problem the team could have chosen to pursue.

3.1 Salt Bodies and Delineation

Most easy-to-find oil and gas reservoirs have been found already, but the major remaining structural traps are generally related to salt bodies. There is major interest in processing salt bodies so as to discover deep-water oil and gas deposits. It is very important to identify and delineate the salt accurately in order to identify locations where oil and gas reservoirs are likely.

The shape of a salt body helps indicate whether oil and/or gas is present, and has thus been a research topic for decades. The main method used for the detection of salt begins with subsurface seismic imaging. This consists of producing sound waves that transfer down below the Earth's surface. Part of the wave gets reflected when it hits a boundary between two materials, and the timing and direction of this can then be detected by equipment on the surface. This results in a "trace" showing the strength of the return wave over time, and thus the depth of geological

boundaries. Done at numerous locations this eventually results in a seismic cube showing the detected boundaries in a large 3D area. This however does not on its own say where salt bodies are located; the data needs to be interpreted.

Due to the unique properties of salt, salt bodies are generally more clearly defined in a seismic cube than other geological features. However, salt bodies also tend to have complex structures in three dimensions, while most other geological features are relatively gradual in any changes in position.

Current methods are struggling with accurate automatic delineation of salt bodies. The tops of salt bodies tend to be flat and are therefore easy to detect, but the flanks often contain complex curving shapes that are difficult to detect. In addition the often completely vertical structure of the flanks can produce weak seismic reflection. The data around the flanks will consequently have a lot of noise. Since the flanks are not being correctly tracked, a seismic interpreter has to supply a lot of manual interpretation. This process can be very time consuming. A surface created from a salt delineation of the F3 data set (see Section 4.1 for further details about the data set) can be found in Figure 3.1.

Salt bodies can be very large. Usually salt bodies are between 0.5-5 miles (0.8-8 km) wide [46]. The F3 data set shows one salt body and this data set is 24x16km.

3.2 Business Requirements

The customer wants an automated form of tracking that performs better than the current solution in three dimensions; accuracy, usability, and performance. A detailed description follows.

3.2.1 Accuracy

Current semi-automatic interpretations are not of the quality the customer wants. In particular, they're not effective at delineating the flanks of salt bodies.

3.2.2 Usability

Today's solution requires that a skilled interpreter manually delineates most of the salt structure. An automated solution able to complete parts of the delineation would ease the workload involved in delineating a salt body, but has to be easy to use in order to be useful.

3.2.3 Performance

Manual interpretation is a very time consuming process, so a fully automated solution would increase the efficiency of the interpretation process.

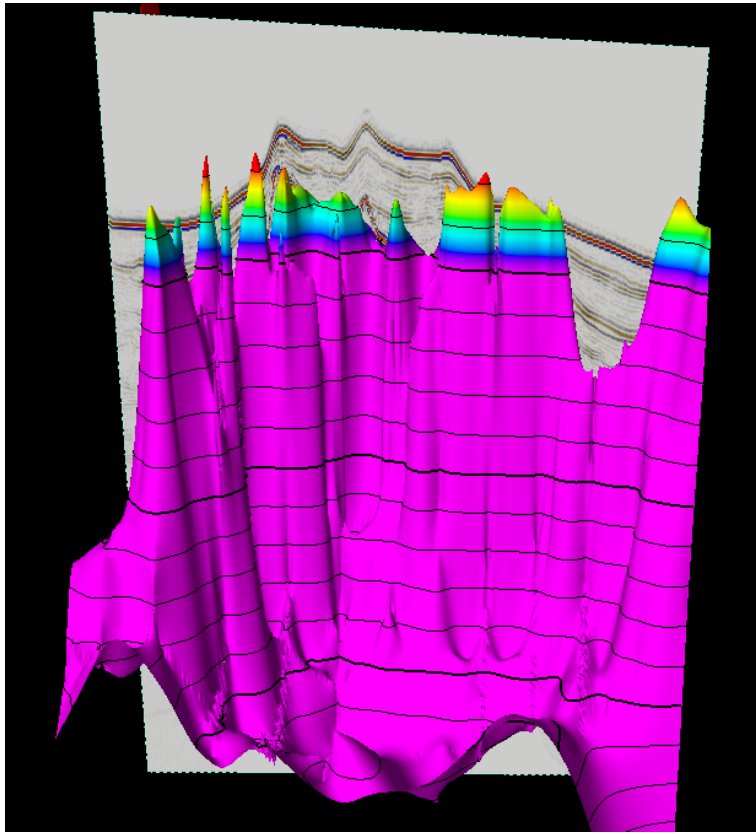


Figure 3.1: A surface created from a salt delineation

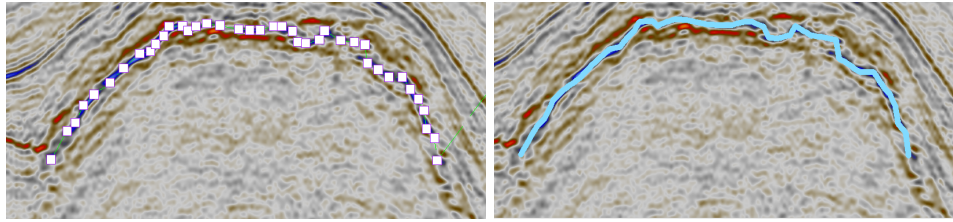


Figure 3.2: Manual interpretation in Petrel

3.3 Salt Delineation Methods in Petrel

Petrel offers several different methods for interpreting seismic data. Four of the basic methods are manual interpretation, guided auto tracking, seeded 2D auto tracking and seeded 3D auto tracking. In addition, Petrel has a method called Multi-Z interpretation. It is a process that handles multiple Z values at the same XY locations, within an interpretation.

Manual Interpretation

This allows the user to draw a seismic horizon, which corresponds to the edge of the salt. This is either done point by point or by clicking and pulling to draw. This is a very time consuming method, since nothing is automated. An example of a manual interpretation can be found in Figure 3.2. This is from the F3 dataset. Manual interpretation does not look at any seismic features; it simply draws a line between all the selected points. This method can be used to interpret weak seismic signals where the other automated methods are powerless.

Guided Auto Tracking

Almost the same as manual interpretation, but instead of naively drawing a line between the user specified points, the software uses an algorithm to determine the best line between the points based on the seismic data.

Seeded 2D Auto Tracking

With this method the user clicks a seed point, which the software expands into a two dimensional seismic horizon. It does this by finding and selecting similar points to the ones already chosen. It starts with only the seed point, and keeps expanding this until it can no longer find any other similar points within the seismic data.

Seeded 3D Auto Tracking

Similar to the 2D auto tracking, but instead of just expanding within the cross-section, it will expand in all directions. By far the most computationally expensive method. Figure 3.3 shows the top of a salt body interpreted using the seeded 3D auto tracking method. Figure 3.4 shows the same salt body in a 3D window, but the figure also includes an intersection. The dataset used in this interpretation is the SEAM dataset.

Multi-Z

Multi-Z is a manual interpretation method, but unlike the previously mentioned manual interpretation method, Multi-Z handles multiple Z values in the same XY coordinate. A triangle mesh can be made from the Multi-Z lines. Figure 3.5 shows interpretation of crossline 4305 in the SEAM dataset. The method used here is multi-Z and is shown in an interpretation window. Figure 3.6 shows a multi-Z interpretation of the same seismic cube, but in a 3D window. Figure 3.7 shows a 3D mesh made from the Multi-Z interpretation from Figure 3.5 and 3.6.

3.4 Neural Networks

An artificial neural network is a network inspired by its biological counterpart, the brain. They can be used to approximate an unknown mapping from input to output, hereafter called function. This is done by gradually modifying the network (known as neural network training) to replicate known data as closely as possible.

Structure

Such a network consists of so-called "neurons". Neurons have four components:

- Inputs - Neurons normally take one or more inputs, usually each consisting of a numerical value on the range 0 to 1, though this can vary significantly. A neuron with no input will always output the same value, which is useful for shifting the activation function. This form of neuron is known as a "bias"
- Weights - Each input is assigned a real valued weight, indicating how important the input is. This can potentially be negative. Each input is multiplied with its weight. Weights are normally initialized to random values within some range
- Activation function - An activation function is a mathematical function mapping the weighted input to an output. Usually this is a monotonically increasing function outputting a value between 0 and 1, but like the input it may also fall outside this range

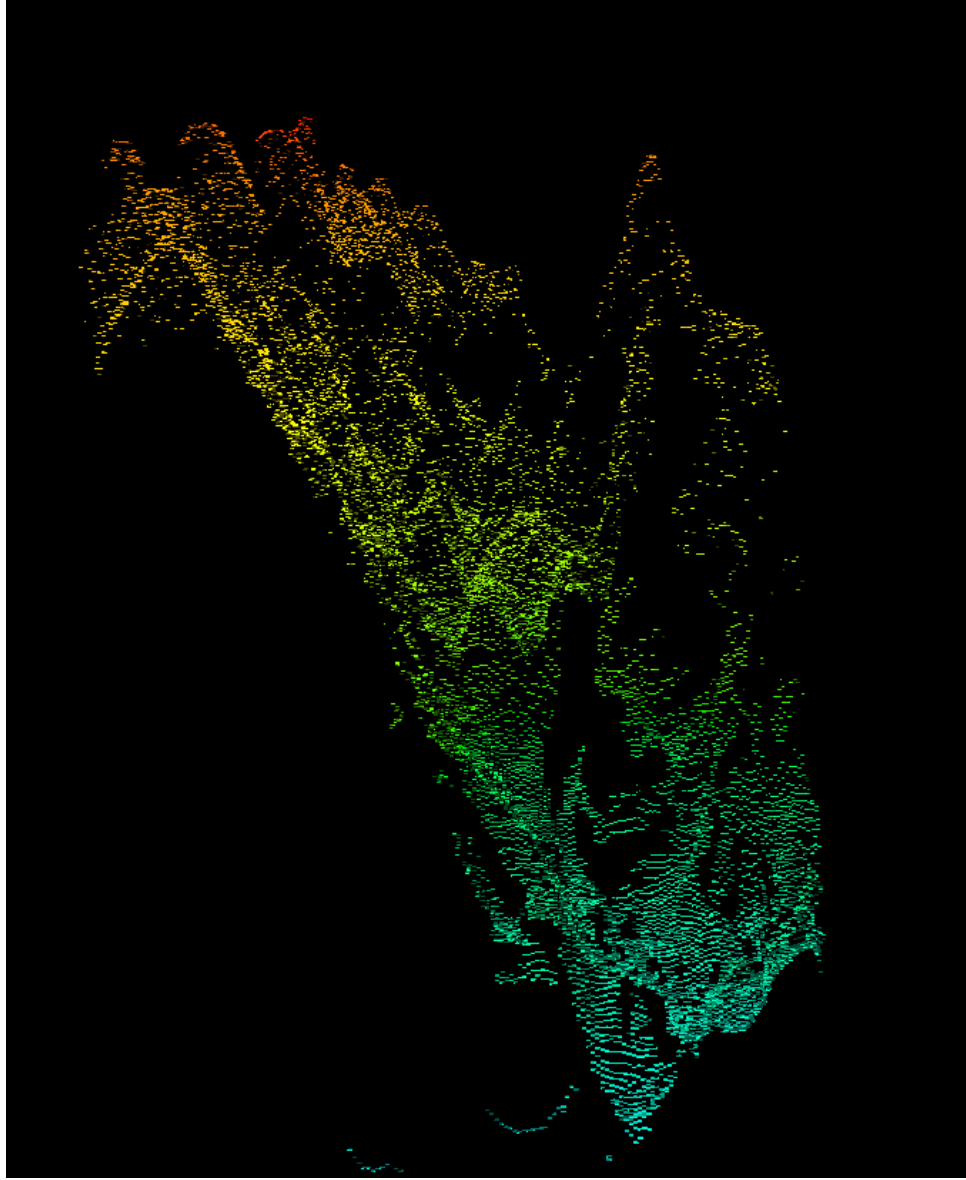


Figure 3.3: Seeded 3D auto tracking shown in a 3D window.

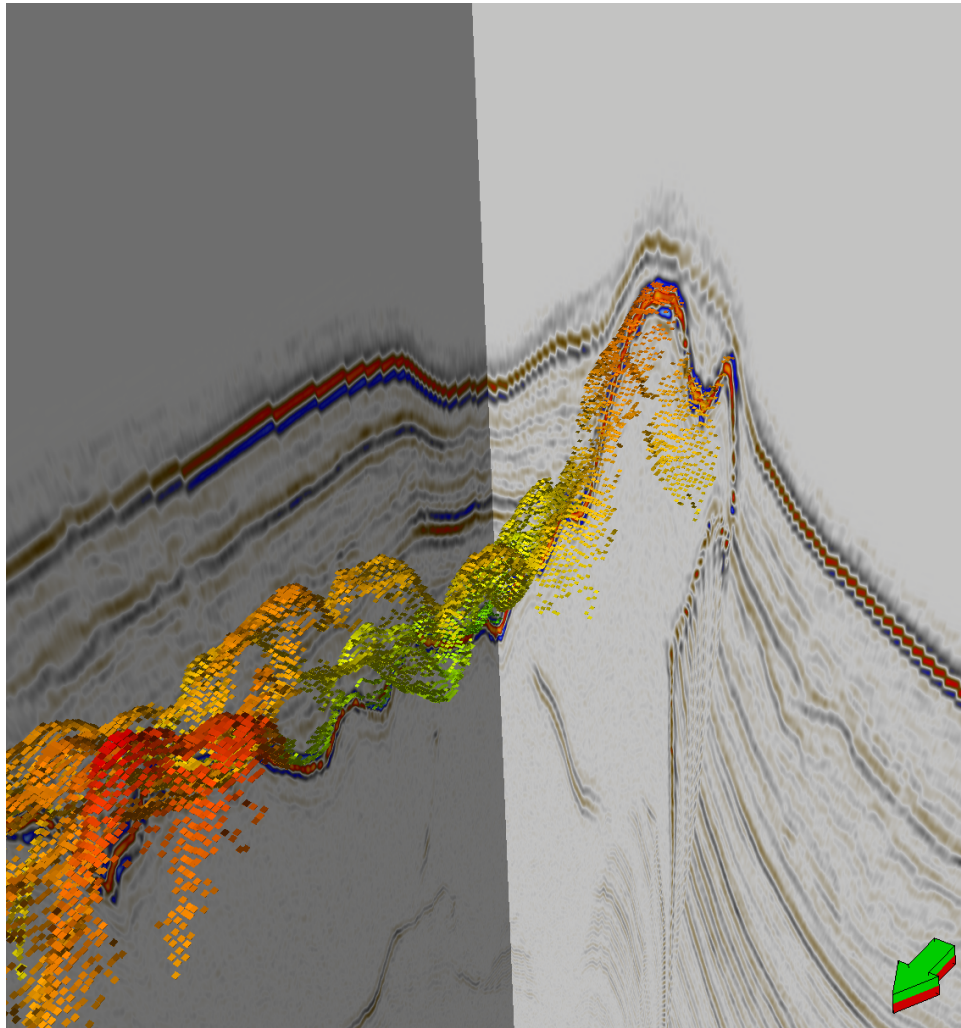


Figure 3.4: Seeded 3D auto tracking shown in a 3D window with an intersection included.

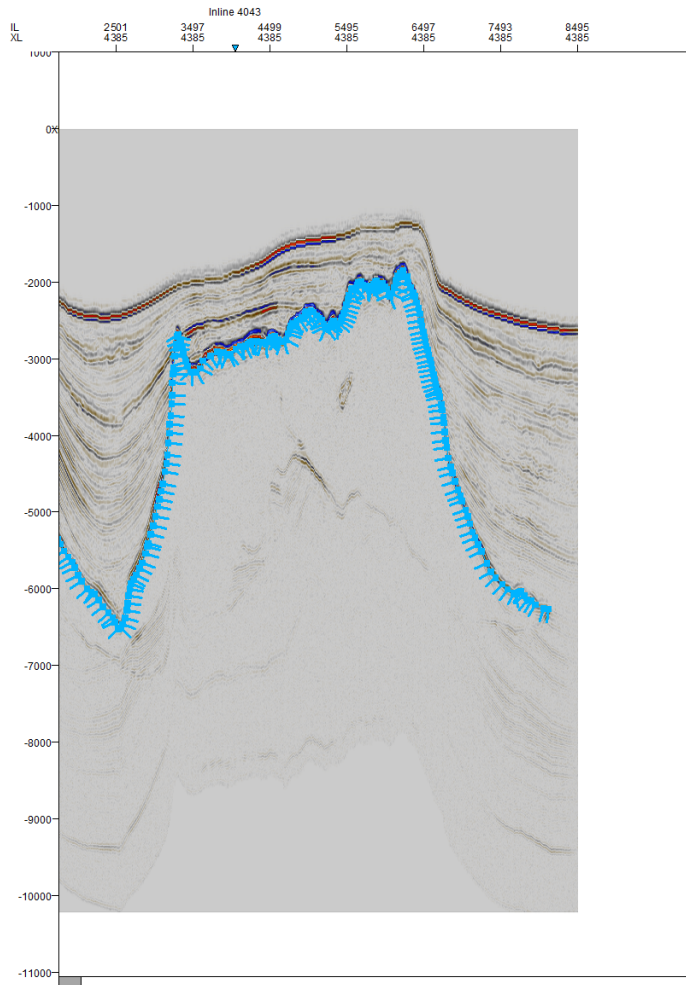


Figure 3.5: Multi-Z points in an interpretation window

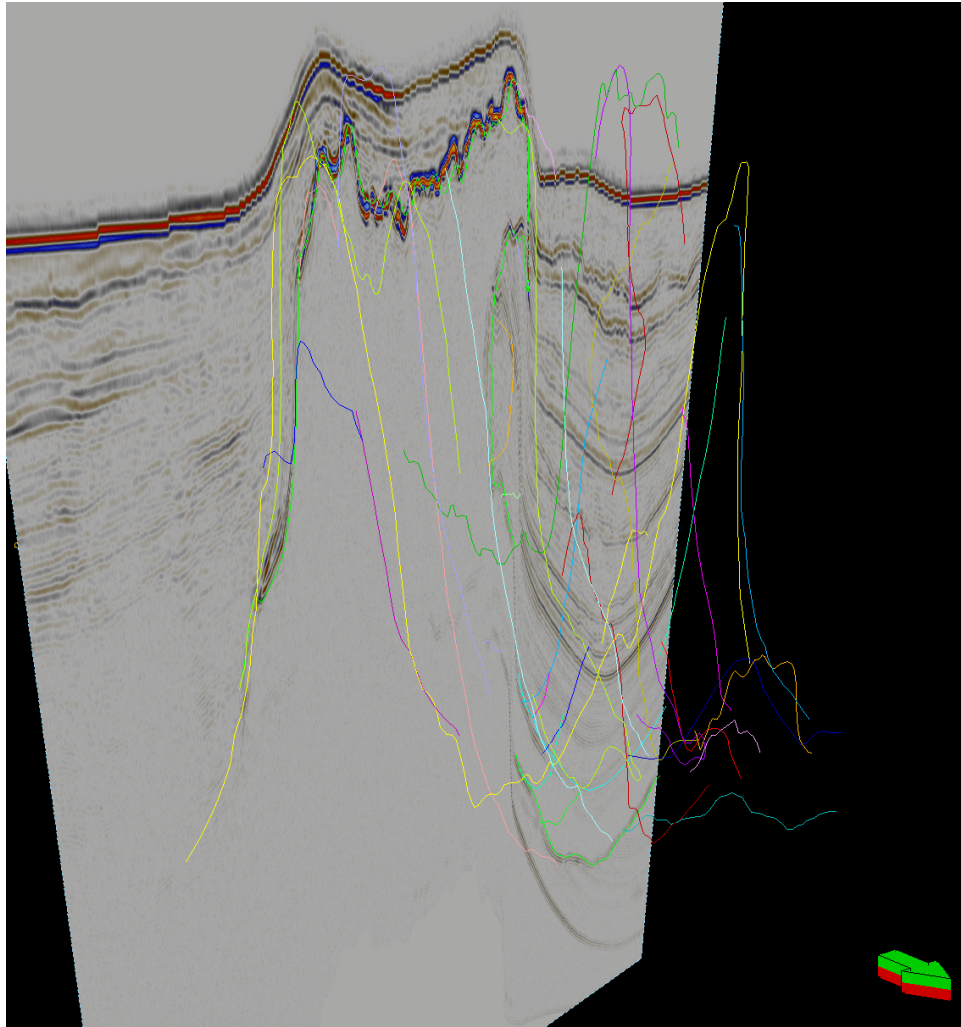


Figure 3.6: Multiple crosslines interpreted using multi-Z shown in a 3D window

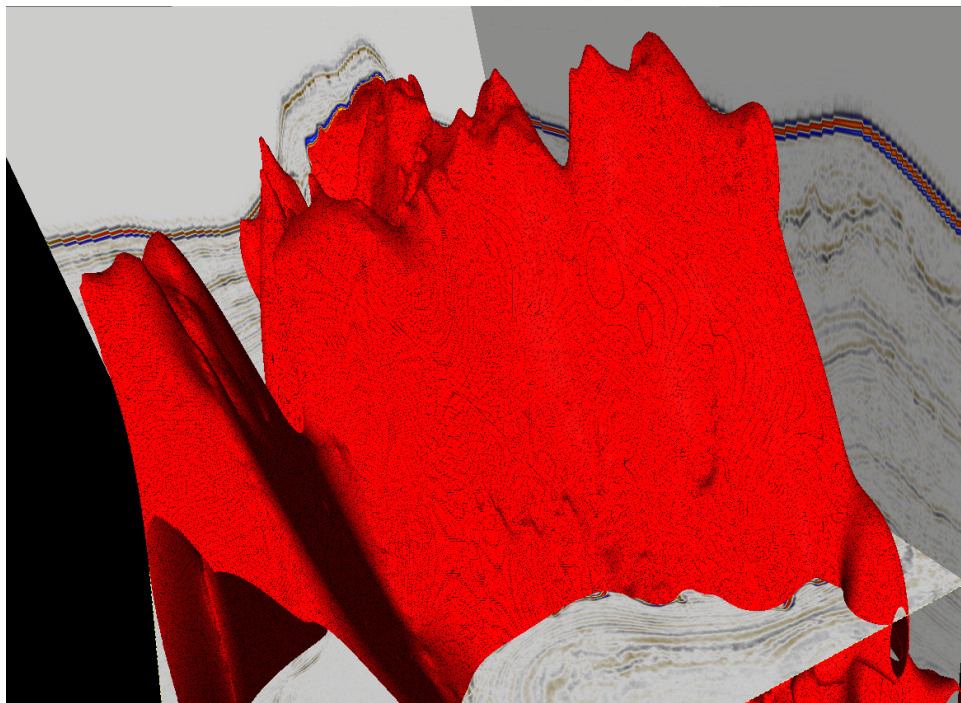


Figure 3.7: Mesh created from the Multi-Z interpretation in Figure 3.6

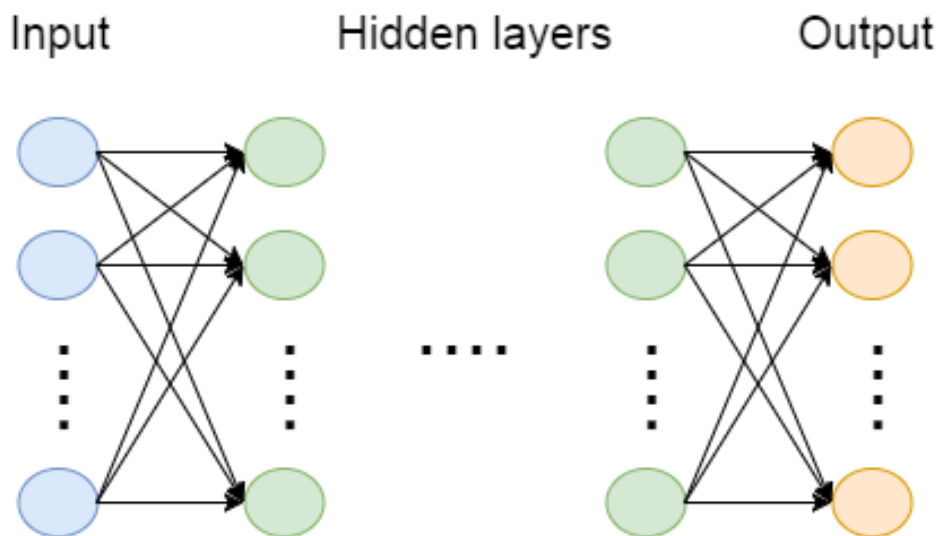


Figure 3.8: General neural network structure

- Output - The result from the activation function is sent to one or more other neurons, or as an output from the network itself. Theoretically it can be unconnected, but such a neuron serves no practical purpose since it will have no effect on the network result

By combining neurons into a network, a layered structure is formed that takes one or more inputs and provides one or more outputs. See figure 3.8 for a general neural network structure. Nodes are generally collected in a layered structure. The first layer takes input from outside the network, while the last layer provides the result. The layers between the first and last layers are known as "hidden" layers since they cannot be directly observed when running the network. Their presence however is critical to approximating complex functions.

Training

The resulting network can then be "trained" to approximate an unknown function, such as whether a given image is of a dog or a cat. This is done by running labeled data through the network, and using a learning process to update the network to better approximate the labeling. A learning process is simply some way of updating the network based on how right or wrong its output was. This can be as simple as shifting each weight in the network a fraction of the way towards outputting the expected value.

By repeatedly training a network, it will gradually approximate the training data provided it more and more closely. The extent to which it can approximate it

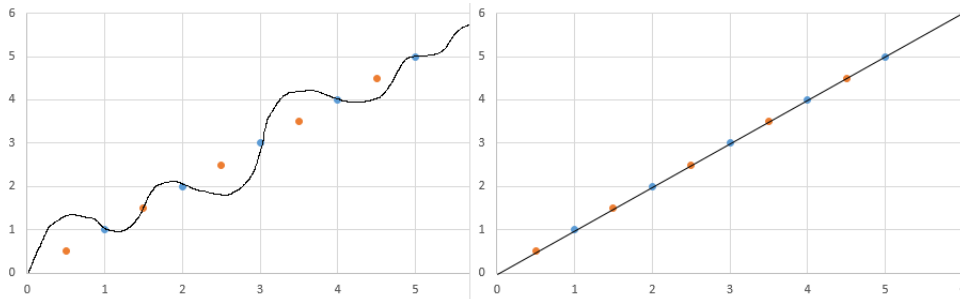


Figure 3.9: Overfitting of training data. The blue points represent training data, while the orange points represent unseen data

depends largely on two factors: the complexity of the network, and how complicated the function being approximated is. The more complex a network, the more closely it can approximate a function. The more complicated a function is, the harder it is to approximate.

Overfitting

A problem that can easily occur is that while the network output for training data is very accurate, its output for new data is not. This is known as overfitting; the points of the function that are known can be approximated in any number of ways with an output function complicated enough, yet this may be completely wrong for unknown points, as the network ends up finding spurious relationships that happen to produce the correct result for the training data. A simple example is shown in figure 3.9. Both proposed functions fit the known blue data exactly, yet the one on the left manages to miss most of the unknown data due to being overly specific. This issue is even more common if the data is noisy. That is, not all the data is completely correctly labeled. Another common issue is that the function itself isn't deterministic; two points of data that look identical have different outputs. This is typically due to inputs that simply aren't available. In these cases the function is simply impossible to replicate exactly, and overfitting can easily occur. This does not necessarily preclude approximating it closely enough to achieve good enough results.

To reduce overfitting, one may utilize a method known as "dropout" during training. For each training step, each node had a given chance (e.g., 50%) of being turned off and thus not affecting the output at all. This means that spurious relations are less likely to creep into the network as the influence of individual nodes is reduced; if some node initially dominates the output, without dropout the other nodes on the same level would not be usefully updated, while with dropout they will get updated in iterations where the dominant node is not included.

Convolutional Networks

Of particular interest is so-called "convolutional" neural networks (CNN). These utilize mathematical convolution, which is a way to combine two separate functions. Convolutional networks use this technique to separately process small parts of larger inputs that have some form of spatial structure. By utilizing the spatial structure of the input, it is possible to achieve great results in a relatively short time, as neurons don't have to also adjust for the input of distant inputs, but instead only a small cluster of inputs. This is often done in several layers, each layer reducing the input. It is normally combined with more traditional neural network layers. Convolutional networks excel at image processing as information in images is normally spatially clustered.

An example of the use of convolutional neural networks can be seen in the article "Deep Machine Learning provides state-of-the-art performance in image-based plant phenotyping" [23]. A group of scientists used deep learning to identify and locate plant features in images, an important part of the phenotyping process. "Plant phenotyping is the comprehensive assessment of complex plant traits such as growth, development, tolerance, resistance, architecture, physiology, ecology, yield, and the basic measurement of individual quantitative parameters that form the basis for the more complex traits." [22]. The plant features of interest were roots and shoots. The scientists trained a convolutional neural network (CNN) to classify portions of a bioimage. This classifier could then be scanned over entire images, outputting the location of the features. The article reports great results, capturing the final ten percent accuracy needed for fully automated systems. For the root CNN, two groups of convolution and max pooling layers were used, followed by two convolution and three fully connected layers. For the shoot CNN, an extra group of convolution and max pooling layers were used.

Summary

In summary, neural networks ability to approximate almost any function make them useful for a large variety of tasks. They particularly excel in cases where the developer has incomplete domain knowledge, the domain is simply too large to encode traditionally, or the domain can change over time. They've been used to great success for image recognition, speech recognition, computer vision, and a variety of other tasks.

3.5 Planned Solution

Based on the pre-study, the team decided that the approach with the greatest chance of success was to use a convolutional neural network. While neural networks had been tried out in the past without producing good enough results, these were relatively shallow and ran on far weaker hardware, and did not utilize convolution.

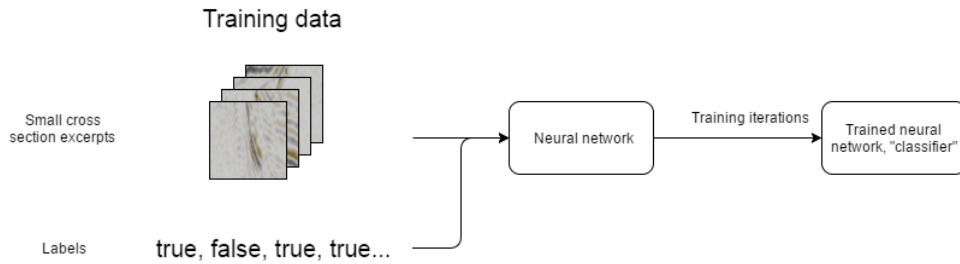


Figure 3.10: Training the neural network

The team therefore believed that with more modern hardware powerful enough to run deep convolutional networks good results might be achieved.

The most significant advantage of this approach is that it did not rely on domain knowledge, which the team as a whole lacked. Instead it stood out as a more general approach, and allowed the team to treat the salt delineation as an image recognition task rather than a domain specific task. Using a *convolutional* neural network also has the advantage of utilizing the seismic data's spatial structure.

Trying something that had not recently been investigated also seemed to the team like it would be more likely to produce significant results than trying to improve on an existing method, since the latter would likely only yield marginal improvements at best.

The team planned to first present a simple proof of concept consisting of a single-layer network, then if it showed promising results, build upon the proof of concept by deepening the network and tweaking its parameters to optimize the results. This would then be integrated into Petrel as a plug-in.

Conceptually the solution consists of two steps. First, train a neural network to be able to output the probability that a small excerpt of a cross section is an edge of a salt body. This is done by feeding the network with a lot of these excerpts, along with labels attached to each excerpt to indicate whether or not it contains an edge. The process is shown in Figure 3.10. When the training process is done, it has produced a salt body edge classifier. This classifier can then be used to scan cross sections to locate the salt body edges, as shown in Figure 3.11.

3.6 Alternative Solutions

There are many approaches for delineating a salt edge, or finding salt bodies in seismic data. Many of these do not require heavy computation, and have been tested thoroughly through many decades of research. Most of them require a lot of knowledge in fields like geophysics, acoustic resonance, and petroleum geosciences.

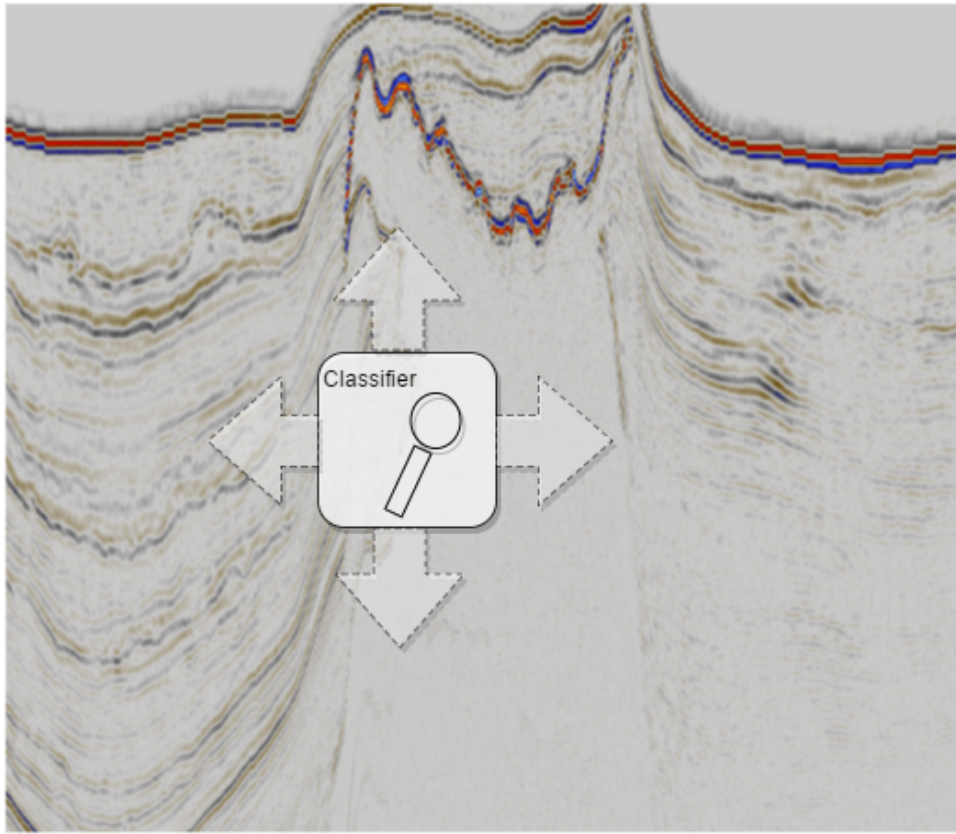


Figure 3.11: Running the classifier over a cross section

The team decided to go with the planned solution rather than these alternate solutions due to two main considerations. Firstly, the team lacked the domain knowledge to properly understand these alternate solutions. Secondly, the team believed that trying something novel rather than attempting to improve on a traditional method had a greater chance of producing significant results.

3.6.1 Bayesian attribute-based salt-detection

This is an approach where different attributes are created from seismic data, and Bayesian classification is applied. This often involves the seismic data being transformed in numerous ways to create enough attributes. These transformations, usually described as *filters*, often try to accentuate salt while reducing noise. A problem with this approach is that if a filter produces redundant information, which is not always easy to avoid, the classifier will become biased.

”Texture attributes for detection of salt” [38] is an article where Bayesian classification is applied to different attributes produced from a seismic cube to detect a salt body. The attributes are based on different texture qualities in the seismic data. Initially a large number of attributes were tested, and by removing those that either produced redundant information, or did not isolate any seismic feature through visual inspection, the results became progressively better.

The seismic data used in the article was the *North Sea 3D* data set, and both cross-sections and timeslices were used. Two sets of cross-sections and timeslices were used. Both sets were delineated manually to label the data. One was used to train the classifier and another to test the classifier. In the end, the best results were achieved by using 15 attributes. The classifier achieved an accuracy of 97.5%.

The difficulty in implementing a Bayesian classification comes from the domain knowledge required. Choosing which attributes to create, and what they actually mean required a lot more knowledge about geology than the team possessed. It would also be difficult to create a better solution than the one outlined in the article.

3.6.2 Edge Detection

Edge detection aims to find the salt-edge using mathematical methods combined with seismic data. By scanning through the image looking for discontinuities in the seismic data caused by abrupt changes in resonance, the salt edge can be found because of the difference in the seismic profile of salt versus the other types of rock surrounding the salt. There are many different ways to implement edge detection, but a major problem is noise in the seismic signal. The edge may be hidden by the noise, and this is not unlikely as the shape of salt body flanks tend to make them less visible in seismic data.

Edge detection is a method which requires extensive seismic knowledge to find discontinuities in seismic data. The team lacked this knowledge, and in addition the method does not handle noise well. Therefore, it would prove difficult to create a system built on edge detection.

3.6.3 Level Set Method

The level set method uses differential equations to calculate a body's boundaries' reaction to forces. Three different forces are used: the normal force resisting deformation, a directional force, and a smoothing force. A salt body is created from a given set of seed points, and the different forces are used to mould the body based on attributes and characteristics to differentiate it from surrounding sediments.

One way to extract salt bodies automatically with the level set method is covered in the article "Automated salt body extraction from seismic data using the level set method" [14]. They created an automatic delineation based on the level set method. The method in the article expanded on the level set method by considering constraints given by the user, e.g. a user could input a previous salt interpretation, and the algorithm would honor those constraints. They also stopped further calculations in completed areas based on stopping criteria. This reduced the running time, because less calculations would be conducted. In addition, the divide between completed, and non-complete areas could be used as a basis for further investigations.

The Level set method is a rather complex method, requiring quite extensive mathematical knowledge. The team lacked the knowledge needed for the mathematical method itself. There exist software for Level set method [17]. However, the problem still remained of how to differentiate salt from neighboring sediments.

4 Technologies

A number of different technologies were used during the development of the product. Most of them are required in order to run the product. Section 4.2 presents Petrel and the SEG-Y and ZGY file formats. Section 4.3 talks about Ocean. Section 4.4 contains information about Python and the libraries used in this project. Section 4.5 talks about different operating systems. Problems related to these and how to solve them is also included.

4.1 Data Sets

The team used three data sets throughout the project. Listed below are the names of these sets, along with a description of them and what they were used for.

SEAM Data set

The SEAM data set is a public synthetic data set created for technology development purposes. More about it can be read at its website [32]. The team used the data set both to train the neural network, and to test its solution. Some images of the cube are also included in the report. While synthetic, it still provides sufficiently complex salt bodies of high quality.

F3 Data set

The F3 data set is a public seismic cube from the Dutch sector of the North Sea. Read more about it here [11]. The team used the data set for images in the report.

3D Seismic Data (CN193)

The 3D Seismic Data (CN193) is a confidential data set from the NTNU- NPD-SCHLUMBERGER PETREL READY Database. The team used the data set when receiving an introduction to seismic interpretation methods in Petrel.

4.2 Petrel



The Petrel Excavation and Production software platform created by Schlumberger can be used to solve petroleum-related subsurface challenges ranging from explo-

ration to development. It brings disciplines together with applied science in geosciences, reservoir engineering, production engineering, and drilling.

The team used Petrel 2016.1[21]. Dicky Harishidayat taught the group about Petrel, and it was used to interpret salt bodies. It was also used to run the plug-in. Petrel supports seismic data in both SEG-Y and ZGY format. The SEG-Y file format is a standard developed by the *Society of Exploration Geophysicists*[35]. This file format was originally for storing seismic data efficiently in magnetic tapes, but has been revised many times since. The ZGY format is Schlumberger's own format for storing seismic data and it supports random data access. The ZGY format is used to optimize 3D seismic performance by storing the data in a compressed format. The files also contain a pre-computed level of detail, which means that each sub-section exists with more than one resolution. Thereby, when a seismic cube is loaded, it will first be loaded in a coarse resolution, and become more and more fine-grained as the program is given time to higher resolution section into memory.

4.3 Ocean Software Development Framework



The Ocean Software Development Framework is an application programming interface created by Schlumberger that lets developers create plug-ins for Petrel. Ocean is based on .NET and C#, and plug-ins can be compiled and debugged through Microsoft Visual Studio. The project is automatically built directly into the folder structure of Petrel. This enables the developer to test the plug-in by simply building the project and starting up Petrel. Debugging can be done by attaching Visual Studio to the Petrel process.

4.4 Python



Source: <https://www.python.org/community/logos/>

Python was chosen as the language for the project for two main reasons. Firstly, the excellent TensorFlow library is written in Python. Secondly, Python is a language designed for quick iteration with minimal overhead, perfect for a project like this.

Initially, the program was written in Python 2.7. Halfway through the development the team switched to Python 3.3 as one library used during development (though not in the final product) was only supported on Python 3 or higher. Upgrading to Python 3.3 also allowed for type hinting, making development simpler (see Section 8.1 for more info).

Libraries

Listed below are the Python libraries used during development.



Source: <https://upload.wikimedia.org/wikipedia/en/7/74/TensorFlow.png>

TensorFlow

TensorFlow is a machine learning system by *Google Brain*, a deep learning research project at Google. The system is open source and can run on both CPUs and GPUs. It only supports UNIX systems (*OS X* and Linux). TensorFlow provides functionality for building, training, and running neural networks, allowing the team to quickly iterate on the network rather than reinventing the wheel by creating a neural network from scratch.

It was chosen due to being one of the leading neural network frameworks on the market, as well as being free and open source. It has the significant disadvantage of not running natively on Windows, which caused a number of problems for the team, but the team was unable to find any neural network framework up to the task that would run on Windows.

NumPy

NumPy[19] was used for arrays and array operations, storing the inputs and outputs from the neural network, as well as being used for post-processing.

SciPy

The "misc" module from the SciPy[29] library was used for image operations. It allowed loading images (PNG images in the case of this project) as NumPy arrays, and saving NumPy arrays to image files. This was used heavily during the prototyping of the product since it allowed for Petrel screenshots to be used as neural network input before the team figured out how to export data directly. It also allowed the team to visualize the results.

PIL

The *Python Imaging Library* (PIL)[26] is a dependency for the "misc" module's image operation. It was initially used directly, but later replaced by the "misc" module as that provided the results in an easier to use format.

SegPy

The SegPy[33] library was used for reading SEG-Y files and turning them into NumPy arrays. This made it possible to use data exported from Petrel as the neural network input. This was later in the project replaced with a custom format, as the Ocean API did not provide a good way to export individual seismic cross-sections as SEG-Y files. SegPy requires Python 3 or higher, which forced the team to upgrade from 2.7 partway through development.

Matplotlib

The Matplotlib[18] library was used during prototyping, though not in the final product, as it allowed visualising results even without saving the output to a file.

PyCharm IDE

To assist the implementation, the IDE PyCharm[25] was used. This helped ensure consistent code style, and allowed for type hinting, thus easing development.

4.5 Operating System

While most of the solution will run on any system that can run Python, TensorFlow itself is a notable exception. Without TensorFlow the solution cannot provide any functionality, so in order for it to be useful it must be run in an environment supported by TensorFlow. Windows is not natively supported, which posed a problem as Petrel is Windows-only. TensorFlow instead only supports UNIX systems, meaning emulation or similar is needed in order for it to run on Windows.

During development, two separate workarounds were used; the *Windows Subsystem for Linux* and Docker. Both provide a virtual Linux environment, are more lightweight than most virtual machines, and relatively simple to set up.

Windows Subsystem for Linux

One way to run TensorFlow is to use the experimental *Windows Subsystem for Linux* [4], which was recently introduced in Windows 10 (August 2016 for the general public) [49]. However, there's two significant drawbacks. It only exists for Windows 10, not earlier versions, and it does not at the time of writing support CUDA. This means that the network cannot be GPU-accelerated, and instead must run on the CPU which is significantly slower. Other than that, it is simpler to set up than Docker, making it the main alternative used by the team during implementation.

Docker



Source: <https://www.docker.com/brand-guidelines>

Docker is an open-source container platform [48]. Docker allows the configuration of images that once created are easy to use and run; the trickiest part is the actual image creation. The team ended up creating a Docker image based around the standard Python image, which installs the prerequisites and runs the Python program the team created.

5 Development Methodologies

”A software development methodology or system development methodology in software engineering is a framework that is used to structure, plan, and control the process of developing an information system.” [36]. A number of different methodologies exist; this section will cover the combination of three different methodologies used for this project: Waterfall, Scrum, and Kanban. Kanban was mostly used for the report.

The project methodology Scrum was first selected, and the project was attempt to be divided into different sprints with sprint backlogs. The team quickly discovered that it was difficult to split the project into different sprints. The reason for this was that the project was so specific and complex that it was impossible to plan more than a couple of days ahead in enough detail to be useful. One week sprints could have been used, but the problem was that it was difficult to plan those sprints. Instead of sprints, the solution was to use a combination of Waterfall and Scrum and divide the project into three main phases: research, implementation, and a testing and finalization phase. It was decided that a minimum viable product (MVP) should be made early in the implementation phase. This was made so that the team and costumer could test the proposed solution before agreeing on a final solution. Section 5.1 talks about Waterfall, Section 5.2 about Scrum and Section 5.3 about Kanban.

5.1 Waterfall model

”The waterfall model is referred to as a linear-sequential life cycle model” [31]. When using this model each phase must be completed before a new one can begin. Like a waterfall, one cannot go back and change something from a earlier phase. Figure 5.1 shows all of the phases in the waterfall model. It consist of these phases:

1. Requirement Gathering and Analysis
2. System Design
3. Implementation
4. Integration and Testing
5. Deployment of System
6. Maintenance

Work in a phase builds on what has been produced in the phase before the present one. So the design builds on the requirements, the implementation on the design and so forth. The project’s life cycle did not use the deployment of system and maintenance phases, but used the rest. These phases were not used since the final product is only a prototype.

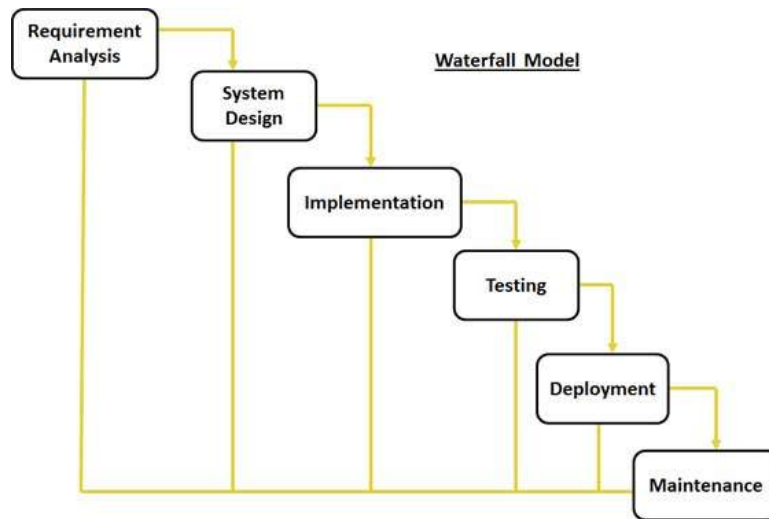


Figure 5.1: Phases of the waterfall model [31]

The advantages are that there are no overlapping phases, so all of the documents can be build on earlier ones with no worries that they may change. It is quite easy to use and works well on small and well defined project. This was a relatively small project, so in combination with scrum it worked well despite all of the uncertainties.

One of the disadvantages of the waterfall model is that it does not allow changes to the requirements and design after these phases are finished. In agile development methods a system is produced early, but this is not the case when using waterfall. This is a poor model for long projects that take some time to finish. The market or the costumers' needs can change during the later phases of the project, but the system cannot change.

According to the Waterfall method, more time should have been spent on documentation. The team also changed documentation from earlier phases and the costumer could decide which features they wanted to be the focus for the next week. This was done as a result of it being infeasible to plan the whole project before commencing implementation, and primarily affected design documents.

Below is a description of the Waterfall phases that had equivalentents in the project.

Requirement Gathering and Analysis

”Requirements convey the expectations of users from the software product” [37]. In this phase, requirements are elicited and collected in a requirements document. This phase corresponds to the research phase in the project. The team wrote requirements, user stories, and acceptance criteria. In addition to deciding the

requirements, the team also did a lot of research on petroleum related subjects, neural networks, and existing solutions to the problem.

System Design

Requirements are studied and a overall architecture is made. This architecture should not violate any of the constraints or requirements found earlier. This phase did not have a separate phase in the project. The team only had a simple overall plan for the architecture before the implementation phase started. The architecture was therefore expanded and changed when necessary during the implementation phase. This is not how it is suppose to be done in waterfall, but the team did not know enough about how to solve the problem in the design phase to make a complete architecture before the implementation phase.

Implementation

The system is developed in units. All the different units are based on the architecture designed in the previous phase. Unit testing is also done in this phase and not in the integration and testing phase. This phase corresponds to the implementation phase in the project.

Integration and Testing

All of the units are integrated and system test are run, both functional and non-functional. This phase corresponds to the testing and finalization phase in the project. Component and system test were executed by the team and the report was finalized.

5.2 Scrum-elements

Scrum is an agile project methodology that defines a workflow consisting of sprints as well as a variety of other artifacts and events. "Agile software development refers to a group of software development methodologies based on iterative development, where requirements and solutions evolve through collaboration between self-organizing cross-functional teams." [47]. Traditional Scrum with sprint planning and burn down charts was not adopted in this project, but some of the Scrum artifacts and events were used. More literature about Scrum can be found at scrumalliance.com[41].

ScrumMaster

The role of a ScrumMaster is to be responsible for making sure that the team understands and follows the practices of Scrum. "The ScrumMaster is often considered a coach for the team, helping the team do the best work it possibly can." [30]. In this project the ScrumMaster was used to run the Scrum standup part of every meeting. More about meetings can be found in Section 2.2.

Scrum Standup Meeting

The point of Scrum standup meeting is to check the status of each team member's work by asking specific questions led by the ScrumMaster. The Scrum standup meeting is in the Scrum methodology a daily meeting, but the team held it only three times a week because it was impossible to fit more than three weekly meetings into everyone's schedule. The duration of the standup meetings was capped at 15 minutes. [5]

The three questions that was asked of each team member during the Scrum standup meeting was:

- What have you done since last meeting?
- Are there any impediments in your way?
- What are you going to do until next meeting?

At first, the team had a fourth question: "What are you going to do during this meeting?". After a few meetings, the team decided that this overlapped too much with the "next meeting"-question to be useful, and therefore decided to drop it. Another reason was that not everybody was going to do some work during every meeting. If anyone had no plans for what to do until the next meeting, they would be delegated tasks by the other team members.

Product Backlogs

The definition of Scrum Product Backlogs is a simple list of tasks that should be done within a project. Hours of work needed and priority is usually estimated in the backlogs. More about backlogs can be found at scrum-institute.org [42].

The team utilized a roughly prioritized backlog with no time estimation. Workload was not estimated as a sprint structure was not used, and due to the difficulty of estimating the size of tasks in a domain the team was not experienced in.

Weekly Sprint Review

Sprint review is a meeting that is held at the end of each sprint. The goal of these meetings is to show what the team has accomplished during the sprint, by showing the participants which backlog items have been completed.

The team had no sprints, but did have a rough equivalent to sprint reviews. Each week the team had a meeting with the customer where they showed what had been accomplished since the previous customer meeting. This was accomplished by describing the work done and using screen sharing to show the results. What the team showed each week can be found in Section 10, which covers the implementation phase.

Weekly Meeting Retrospective

The retrospective is an important event in Scrum because the team can discuss how to improve the team and its process, rather than the product it self. Three questions were asked during each retrospective:

- What did we do well?
- What did we learn?
- What should we do differently next time?

The team held retrospectives during the last group-meeting each week. This worked very well since a lot happens within one week, and it takes time to improve the team. A weekly status update was therefore effective for driving gradual improvement. More about Scrum retrospectives can be found at scrumalliance.com [16].

5.3 Kanban

The Kanban methodology is a way to limit the amount of initiated work in progress and manage the workflow of a project. Its purpose is to avoid redundant work in a project and it is based on five basic principles:

1. Visualize the work.
2. Limit the amount of work in progress.
3. Manage flow.
4. Make Process Policies Explicit
5. Improve Collaboratively (using models and the scientific method)

The amount of work has to be visualized before a limit can be set. This can be solved by collecting all of the backlogs of the work and creating an overview with different columns representing the status of the work. The Kanban board can consist

of different columns like "to do", "in progress" and "done". After that, the team has to agree how much to limit the amount of work in progress. To manage the flow of the work, the team then has to analyze the flow of their project cycles, to see what bottlenecks exist. As part of this, the policies determining when a task flows from one state to the next have to be made explicit; without explicit policies, making lasting change is difficult. Once all this is in place, identified problems can be rectified, and further improvements can be made. Measurable results are especially helpful in this regard. [15] [44]

The team used all of the five principles mention earlier. GitHub's "project" feature was used for Kanban boards to visualize the work in the project (Figure 5.2), but was only used for the report writing and not for the implementation part. Kanban was attempted for the implementation part at first, but there were very few backlog items for the implementation part so it was meaningless to use Kanban since the amount of work in progress ended up limiting itself.

There were four different columns that represented the status of tasks. The columns were, "backlog", "to do", "in progress", and "done". The distinction between "backlog" and "to do" was that tasks in "backlog" were not yet possible to begin work on due to relying on tasks not yet completed, while tasks in "to do" would be possible to begin work on at any time. The amount of work in progress was set to a maximum of 10 tasks; once there were that many tasks in the "in progress" column, no more could be added until a task had been moved to "done". When it was decided to used Kanban, the team made explicit rules for when a task should be moved between the different columns. Tasks were also roughly prioritized within the "to do" column, and generally assigned based on priority.

In the beginning the tasks were few but large. Over time the team divided larger tasks into smaller ones when possible, as the larger tasks became bottlenecks, often leaving the team member assigned to any given task with too much to do. Kanban helped the team progress with the report, and overall worked very well.

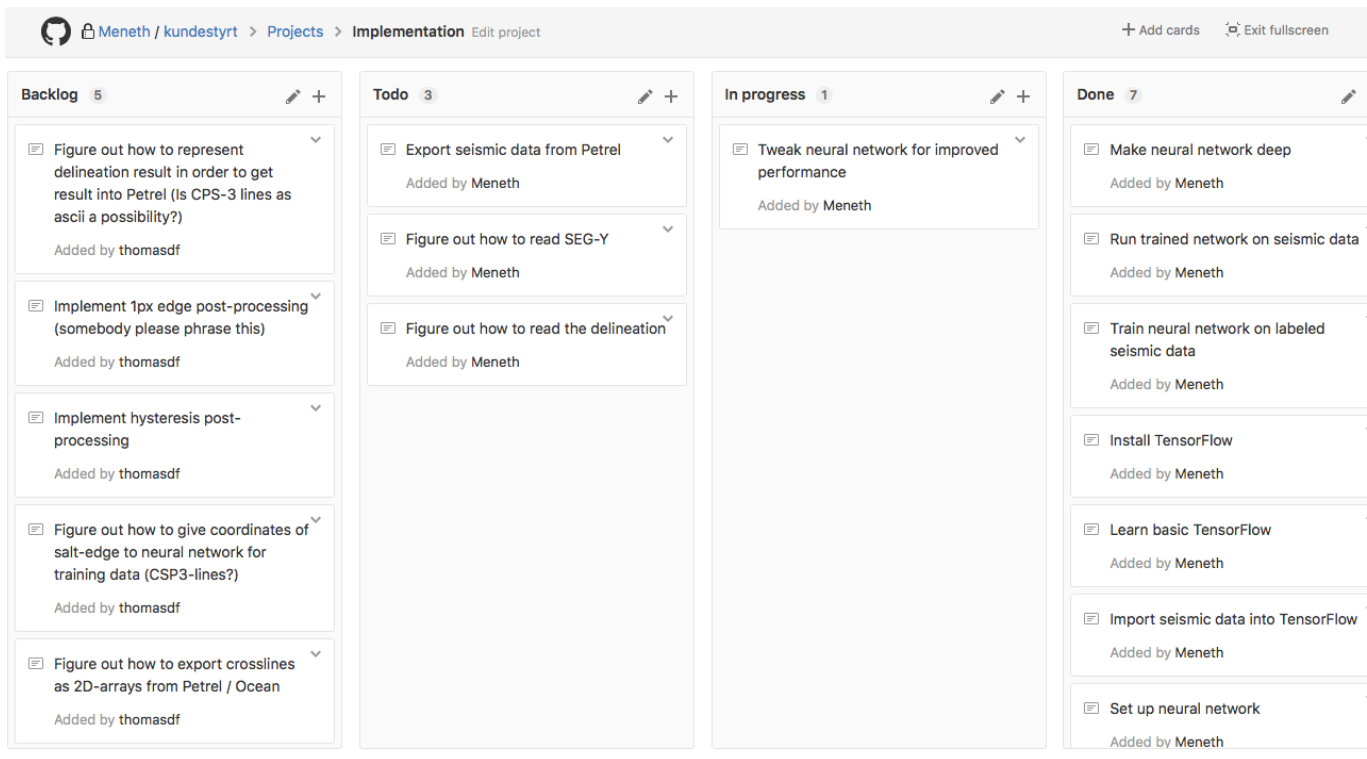


Figure 5.2: Kanban board

6 Requirements

This section contains the specification of the requirements for the system. The requirements were elicited from talks with the customer. It took some refinement across a few iterations to complete them. The customer wanted a threefold specification of each requirement with increasing level of granularity - the statement of the requirement, the user stories relevant for the requirement, and the acceptance criteria for the requirement. The system ended up having only a single requirement. This is because the project problem specification was quite narrow, in that the team was asked to develop a system to do a specific task. Thus, on a conceptual level, the system performs a single function.

6.1 Requirement - Delineate Salt Body

The plug-in will take a seismic cube as input and produce a delineation of salt bodies, if present, as output.

6.2 User story - as Petrel user

As a Petrel user I want to automatically delineate salt bodies from a seismic cube in order to extract/get an object which describes the body.

6.3 Acceptance Criteria

The acceptance criteria are what the customer expects of the final product in terms of "conditions of satisfaction" [45]. They are statements about the implementation that will either pass or fail. They are the most low-level and detailed part of the team's requirement specification. They are, however, still an important part of the vocabulary needed to reason about the system. Thus, they are written with simple and clear language, using a terminology familiar to the customer. The following acceptance criteria were derived:

- Fully automatic delineation of salt bodies in a seismic cube
- The plug-in will work with both SEG-Y and ZGY files
- Use the same or less time than the current method of tracking the entire salt body
- Does not need user interaction except in Petrel
- Delineation is close to being as accurate as auto tracked interpretation
- The plug-in will support Petrel

- The delineation will be deterministic
- The plug-in will support extremely large datasets
- The plug-in will support very complex geological shapes
- The plug-in will respect geological rules
- The plug-in will be easy to use
- The plug-in will not disturb the workflow of an interpreter
- The output will correspond with the input data, possibly scaled

7 System Design

This section will present the design of the system. Section 7.1 talks about the most relevant quality attributes for this project. Section 7.2 presents the architectural significant requirements. Section 7.3 shows the architecture for the system. Section 7.4 shows how the data flows through the system.. These were derived in the process of creating the system that will satisfy the customer’s requirements, stated in Section 6. The project had a narrow scope with minimal size in terms of functionality, and this is reflected in the system design.

7.1 Quality Attributes

Quality attributes are a way to assess a system using non-functional requirements. There are many different possible quality attributes to choose from, but most would not be relevant for this project. The following paragraphs describe the most relevant attributes for this project. Not all were actively pursued in the project.

7.1.1 Security

Software security is an important aspect of software development. When weaknesses in the security of software is exploited, there can be major consequences. Software is depended upon for so many tasks, and for storing valuable data, that security cannot be ignored.

The team used Protection Poker [24] to examine the security needs in the project. The goal of Protection Poker is to identify the most valuable assets, and most exposed features of a system. This is done by the team coming to an agreement on the score for different assets and features on a scale of 10 to 100. The scale is relative to the most and least valuable assets and features of the program. These can then be used to find the areas of the program where there should be extra focus on security.

The team played Protection Poker with an external moderator. The moderator guided the discussions, so they were as productive as possible. The team came to an agreement on the most exposed features, and the most valuable assets in the project, see Table 7.1.

Asset	Value
Seismic cube	100
Salt delineation	60
Neural network structure	40
Training data	70

Table 7.1: Assets identified during Protection Poker

The program developed in the project has some valuable assets. The underlying neural network contain quite some value, because it potentially can save a lot of man-hours delineating salt. The most valuable assets the program deals with are seismic cubes. Constructing seismic cubes requires specialized equipment and knowledge, and access to the areas surveyed. Seismic cubes can be used to find areas likely to contain oil, which means they can be used to obtain high values.

However, the seismic cube is not something stored by the program. It is given as input by the user, and does not persist within the program after the delineation has been completed. To get hold of a seismic cube from the program requires access to the program at runtime, meaning that the program does not increase the chance of theft, or loss of data. This project has a narrowly defined problem, where the developed solution has very limited functionality. Therefore, there are few angles of attack for people with malicious intentions. The end result of this project is a separate program with no connection to the Internet, or other pieces of software except Petrel. The information sent to and from Petrel is also very restricted. If the program has downtime, surrounding systems would not be directly affected by it, but it can lead to some delays in a user's workflow.

Because of the limited functionality, and the isolated nature of the developed software, measures for increasing software security were not a priority. Subsequently, a scenario for security was not developed. The program has valuable assets, but its functionality is limited and difficult to exploit.

If the program had been developed further into a plug-in with more functionality and connections to other software, then it would be pertinent to reconsider the approach to software security. There are multiple possible additions to the system which would lead to a more exposed system. One possible addition would be to move the delineation process to a remote server. Then the system would contain more angles of attack. Another addition could be to let the user give parameters and data for training. Incorrectly labeled data could lead to the neural network misinterpreting the salt, making the delineation wrong.

7.1.2 Performance

When discussing performance as a quality attribute, it is the responsiveness of the system that is the focus. This can either be measured in latency, or in throughput. Latency and response time for the program was not a valuable metric for this project, compared to throughput. The program does almost its entire job without taking input from the user. The only interaction between the user and the program after installation, is the user running the neural network. The wait time for the user comes from training of the network, and the delineation of the salt. These are best measured by throughput, because the time the program needs is entirely dependent on the amount of data provided by the user. The program uses seismic cubes, which can contain thousands of cross-sections. The number of cross-sections affects how long the system would need to complete its task.

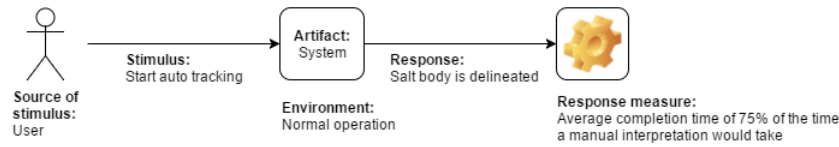


Figure 7.1: Performance scenario

Performance is the most important quality attribute for this project. The entire point of the project is to deliver a program which can lessen the time spent delineating salt for the end user. Therefore, how long the program uses is of utmost importance. If the program is too slow, there would be no use for it. See Figure 7.1 for a scenario for this attribute.

There are several ways to achieve high performance within the constraints of the system. Some are at the cost of accuracy of the system. Reducing the number of cross-sections the program delineates is a simple, but very effective method. Only delineating every other cross-section cuts the time needed more or less in half. The granularity of the delineation suffers, but the results should still be quite good. Another way to improve performance is to reduce the resolution of the cross-sections delineated. By either averaging or picking the maximum from groups of pixels, one can reduce the number of pixels labeled by a factor of four or more. One method that does not affect the accuracy is running the delineation in parallel. This can be done to a certain extent when running on a CPU. However, by running the program on a GPU this can be done to a much larger degree. This can potentially reduce the running time of the delineation by an order of magnitude.

7.1.3 Usability

Usability is how easy the software is to use for the user. A program designed with usability in mind, will most likely provide a better user experience. For this project the interaction between the user and the program is limited. After the user provides input, the program runs without user interaction until it provides a result. Therefore, usability plays a smaller role than it does in most projects, but should still not be neglected.

Steps to ensure usability usually involve streamlining the workflow for the end user. This is not difficult to achieve given the limited functionality of the system. The program needs minimal user input, and the interaction between the user and the program should therefore be quite easy to streamline as much as possible.

7.1.4 Maintainability

Maintainability as an attribute indicates to which extent software accommodates future changes. In general this is important because software rarely remains un-

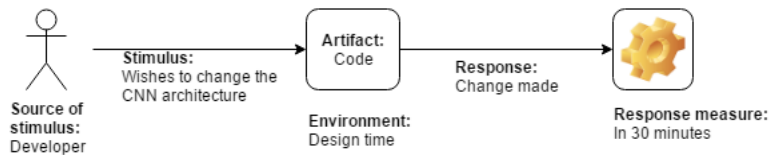


Figure 7.2: Maintainability scenario

changed after the initial development ceases. This will almost certainly be the case in this project. The task of delineating salt is a complex task, and the limited time frame of the project means that the customer almost certainly needs to continue work on the developed program.

Therefore it is imperative that the design of the system facilitates further development in the future. To accommodate future changes, the code has been developed according to the established code standards, see Section 8.1. The solution is dynamic by nature, because the system changes by the training conducted on it. See Figure 7.2 for a scenario for this attribute.

7.2 Architecturally Significant Requirements (ASRs)

Not all requirements of a system have an effect on the architecture. Some are more significant than others. This section lists the requirements and constraints of the system, in context of their significance on the design of the architecture.

7.2.1 Technical Constraints

These are the constraints relevant for the architecture imposed by the technologies the team employ.

Using TensorFlow as the Neural Network of Choice

TensorFlow is not supported on Windows, but the customer’s own software, that the team intends to develop a plug-in for, supports *only* Windows. This complicates the system integration process, because it means the two pieces of software cannot directly communicate on the same platform. The architecture needs to provide an alternate means of communication.

Ocean Framework

Plug-ins for Petrel need to be developed using the Ocean framework. This framework has an architecture in itself, to which any plug-in consequently will need to adhere.

7.2.2 Business Constraints

These are the constraints relevant for the architecture imposed by either external sources, or by sources over which the team has little control.

Development Time

The system architecture will reflect the fact that the time span of the project is merely twelve weeks. This means that the architecture will not be refined and improved over many iterations.

Team Members' Experience

The system architecture will reflect the fact that none of the team members are experienced architects. This means that the architecture could have some flaws a more experienced architect would spot, and that a more appropriate architecture could be possible.

7.2.3 Functional Requirements

These are the constraints relevant for the architecture imposed by the functionality the system needs to provide. Since the system only had a single functional requirement - delineate salt bodies (see section 6.1) - no significant constraints were imposed on the architecture by the functional requirements.

7.3 Architecture

The architecture of a system defines how the different software components in the system are related, how they communicate, and their properties. The goal of the architecture is to realize the quality attributes and requirements of the system. It also enables the team to reason about the system. In this project, it was clear from the beginning that the system would need only a minimal architecture. This is because of the functionally limited scope of the project task. The team therefore used an agile approach in the design of the architecture, and the development of the system was not based on any pre-defined architecture.

CNN Architecture

Figure 7.3 shows the architecture of the convolutional neural network.

Development View

It is useful to use different views of the same architecture when addressing the various, and often exclusive, concerns of the stakeholders of a system. The end-user

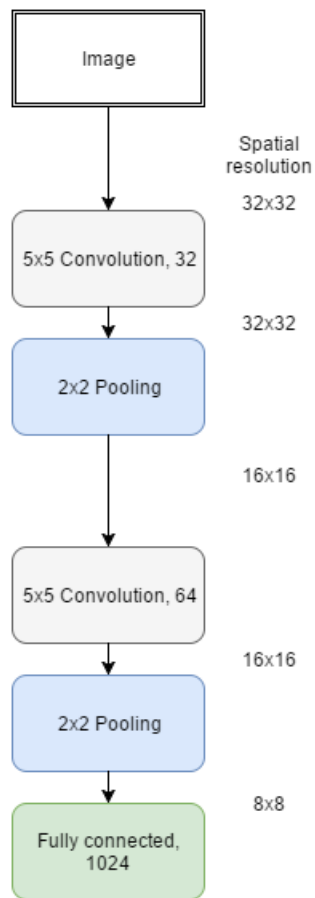


Figure 7.3: CNN architecture

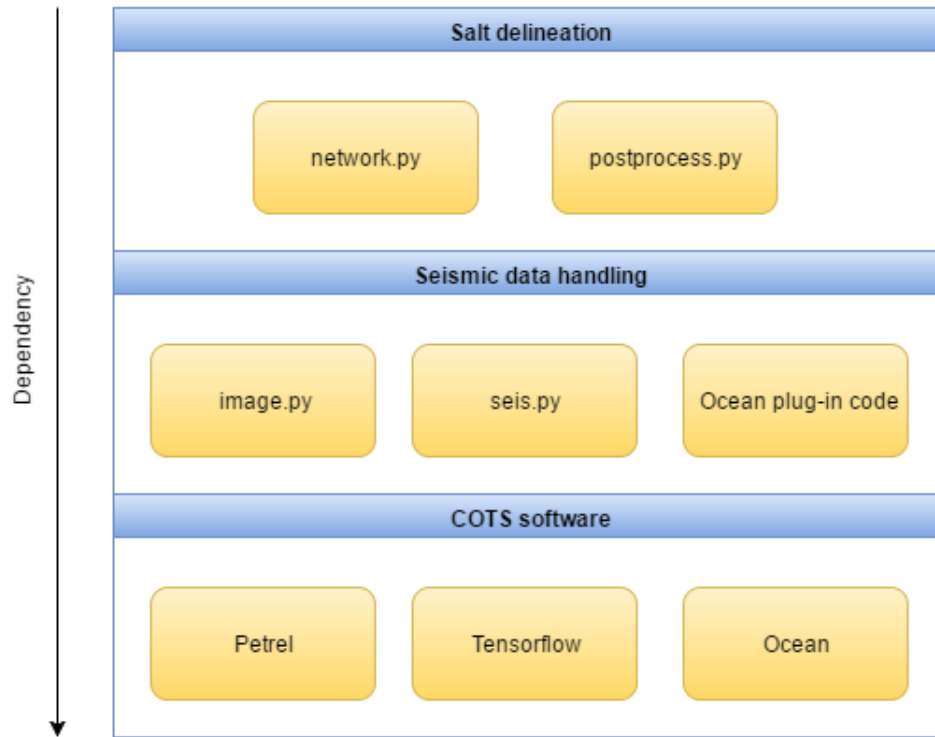


Figure 7.4: System development view

is for example only interested in the functionality, while the developers are also concerned with how the system should be developed to deliver the functionality. A view aimed at the developers is the development view. It splits the system into parts that can be implemented simultaneously, and defines which parts depend on which. The goal is to ease and increase the efficiency of the development process. Figure 7.4 shows the development view of the system.

7.4 Data Flow in the System

This section describes the route of data in the system. It shows two data flows - the flow when training the neural network, and the flow when delineating salt using a trained neural network.

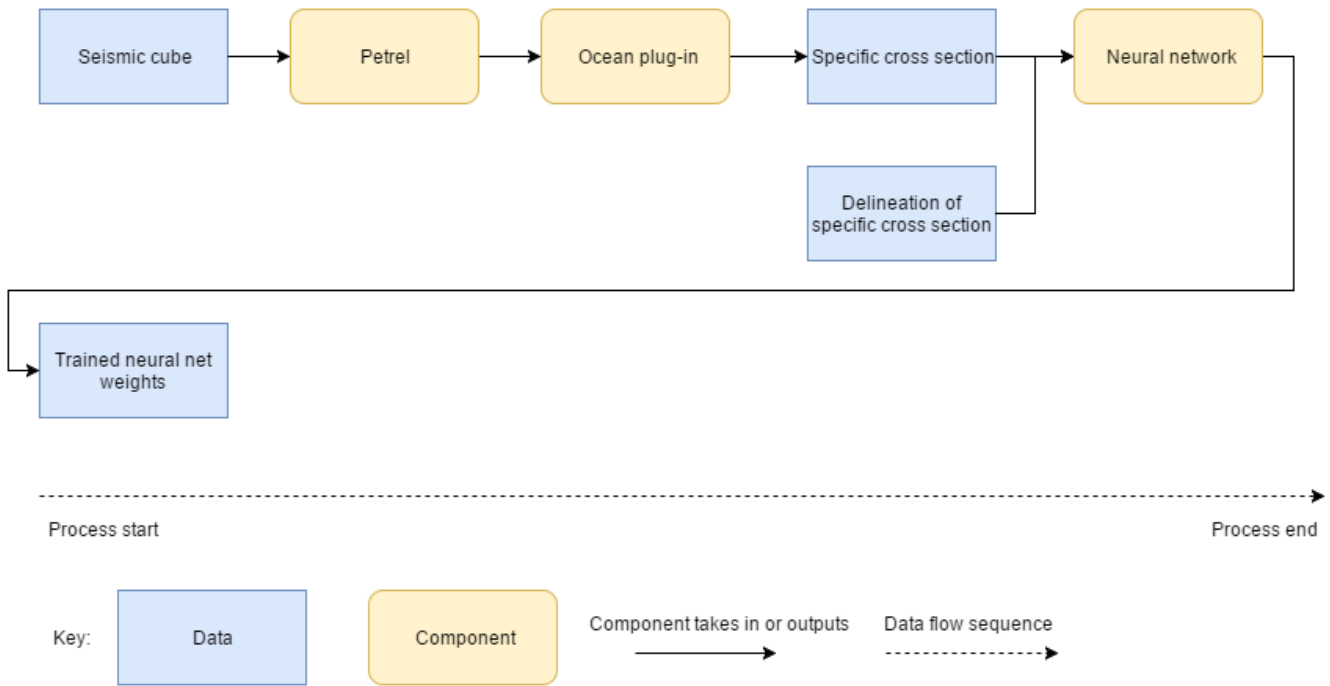


Figure 7.5: Data flow during training

7.4.1 Neural Network Training

For training, the neural network needs two separate sets of input data - the cross sections themselves, and the corresponding delineations. The cross sections are exported as individual text files using a custom Petrel plug-in. These text files get visualised, before a manual interpretation is drawn onto the visualisation. The cross section, along with its delineation, is then fed into the neural network. When the neural network is saved, the weights of the network are saved, and the classifier is ready to run. Figure 7.5 is a visualisation of the data flow during training.

7.4.2 Salt Delineation

To delineate a salt body, exported cross sections from the Ocean plug-in are fed to the previously trained neural network. It outputs a salt delineation for a cross section, which the Ocean plug-in transforms to a 3D point set. When imported to Petrel, this set represents a salt body. Figure 7.6 is a visualisation of the data flow during delineation.

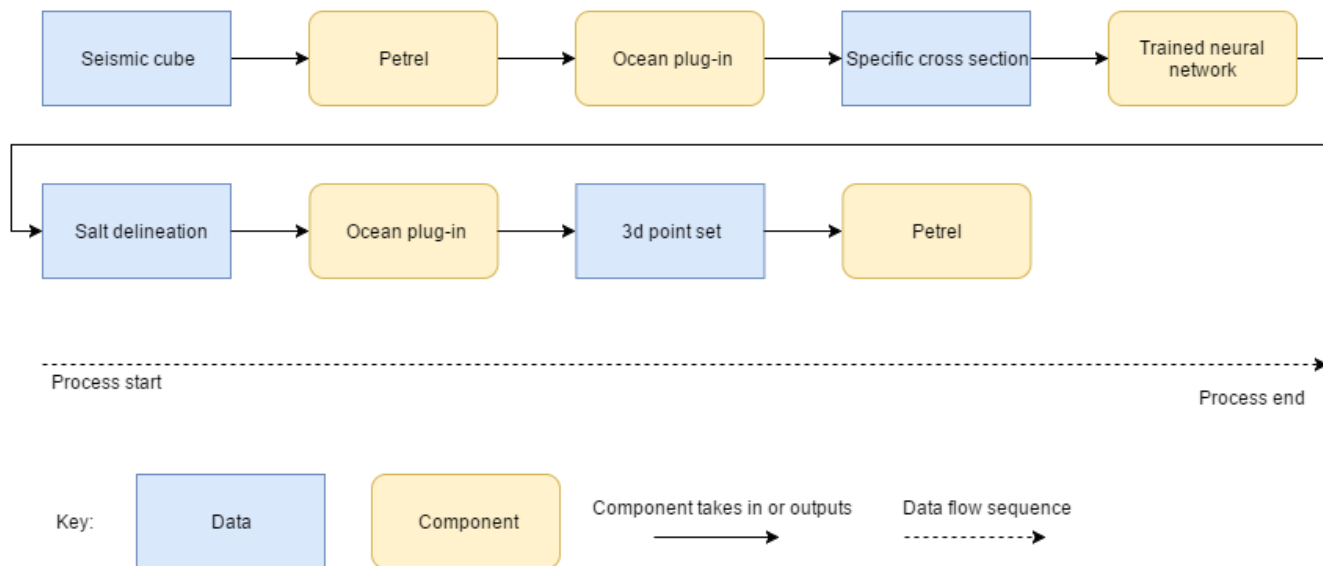


Figure 7.6: Data flow during delineation

8 Programming

At the beginning of the project, and throughout the project as the need arose, programming best practices were established. These consisted of code standards and code reviews. They were intended to help ensure quick development and a good resulting codebase. Section 8.1 talks about code standards and Section 8.2 about code review.

8.1 Code Standards

Code standards were established to ensure consistency in the code base, and to help ensure that the code would be easy to understand and update, even for future developers. Since Python was used, the PEP8 standards[20] were adopted as part of the code standards for the project. This ensured everyone was on the same page about how the code should be formatted. Beyond that, all function names should clearly indicate what the function did, and variable names were to indicate the purpose of the variable.

Code should be self-documenting when at all possible. Comments should be used when that is not possible to provide context for what is happening in the code, such as why a particular choice was made when there are other obvious approaches. To further assist this, all functions should have an associated docstring further explaining its purpose. Function parameters and return values should when rea-

```

def get_image_data(name: str, window_size: int, step: int=1, ratio: int=8) -> (np.array, np.array):
    """Returns labeled data for a given cross-section"""
    line = down_scale_labeling(
        misc.imread("training/TextCrosslines/Delineated/" + name + "_delineated.png", flatten=True))
    image = down_scale_data(seis.convert("training/TextCrosslines/Textfiles/" + name))
    image /= 2000.0 # Roughly the highest value found in any of the cross-sections
    image += 1.0
    return read_labeled_data(image, line, window_size, step, ratio)

```

Figure 8.1: Code sample

sonably possible use type hinting to make it clear what form the input and output takes.

Clearly distinct pieces of code should be separated out into different files so that when programming on one part of the program, it is made clear that the exact implementation details of other files is not relevant. Further, functions should never be longer than what can be comfortably contained on a single screen, defined as 60 lines in this project.

Global state should never be used; any information needed by a function should instead be passed to it as one or more parameters. The amount of code in the outermost scope (the "main" function) should be kept to a minimum; most should instead be contained within functions called by it directly or indirectly.

Figure 8.1 shows one of the functions from the solution, which exemplifies some of the above code standards. Notice for example how the parameters use type hinting to indicate what type of variable they are, that the function names clearly describe their purpose, and the docstring further describing what the function does. Figure 8.2 shows an example of how the TensorFlow library was used in the program.

8.2 Code Review

All code was to be reviewed by another member of the team to ensure that the code standards are followed, that there are no obvious bugs, and that there are no obvious missing improvements.

This helped ensure high-quality, easily understood code. Code was assigned for review by file from time to time. Combined with the version control procedures (see Section 2.6), this ensured that the number of lines of code included in each review stayed manageable so that the reviewer would not get overloaded.

Code review helped uncover a number of bugs, thus saving time spent debugging.

```

def setup_network(self) -> None:
    """Sets up the neural network, including the training mechanism and accuracy"""
    self.inputs = tf.placeholder(tf.float32, [None, self.num_inputs])
    self.correct_labels = tf.placeholder(tf.float32, [None, 2])

    # 5x5 pixel patches to determine 32 features
    W_conv1 = weight_variable([5, 5, self.channels, 32])
    b_conv1 = bias_variable([32])
    x_image = tf.reshape(self.inputs, [-1, self.window_size, self.window_size, self.channels])

    # Convolve and max pool
    h_conv1 = tf.nn.relu(conv2d(x_image, W_conv1) + b_conv1)
    h_pool1 = max_pool_2x2(h_conv1)

    # 5x5 feature patches to determine 64 features
    W_conv2 = weight_variable([5, 5, 32, 64])
    b_conv2 = bias_variable([64])

    # Convolve and max pool
    h_conv2 = tf.nn.relu(conv2d(h_pool1, W_conv2) + b_conv2)
    h_pool2 = max_pool_2x2(h_conv2)

```

Figure 8.2: Code sample

9 Testing

Testing is an important part of software development. The goal of testing is to verify that a program behaves as expected. This is done by conducting one or more tests on the program. A test runs the program with specific input, and sees whether the program gives the correct response. This information can be used to detect and correct errors overlooked by the developers. The testing process is a way to increase the likelihood that the product being developed works as intended. Section 9.1 presents the framework used. Section 9.2 lists all of the test methods considered. Section 9.3 describes the test plan. Section 9.4 explains the test case specification document that can be found in Appendix D. Section 9.5 shows the results and reflection for the testing conducted.

9.1 Testing Process Framework

The project test plan and other test documents were partially based on the ISO 29119 standard for software testing [13]. The standard is quite comprehensive so only a subset of the documents and processes outlined in the standard were used in the project and its report. The standard is also a generic standard meant for large projects. Therefore, some parts of its processes and documentation would lead to a large overhead, with little added value. The documents included from the standard is a lightweight version of a test plan, a test case specification document, and lastly the completion report. Static testing, testing were the product is analyzed without running code e.g. code review, is covered in Section 8.2.

9.2 Test Methods

There are several different ways to perform software testing, and the testing process of this project includes several of them. The different methods considered and used are described in this section. The exact specification of the tests conducted are located in Appendix D

Given the nature of the solution created during the project, there are some factors complicating testing of the software. It is possible to test most of the program using ordinary methods. The most difficult aspect is the testing of the neural network. The seismic data used in the program was divided into three parts. The training set was the data fed into the neural network to train it to detect salt edges. The validation set is used during the training of the neural net to tweak the parameters of the neural network. After finishing the training, the test set is used to see how well the finished neural net performs. This is used to see how well the neural network works on "general cases", because it could be overfitted to the validation set. It is important to not continue training after using the test set. This could lead to overfitting to the test set, and it would lose its value.

Unit Testing

Unit testing is a process where one tests the smallest possible parts of a program, usually individual methods. This is to make sure that the individual parts of the code work as intended. After the tests are created, they can be used during development to make sure changes in the code do not adversely affect its function. These tests should be automated so that they are as easy to make use of.

This project is heavily dependent on the framework TensorFlow. Most of the methods written contain calls to methods in TensorFlow, where the computations needed are performed. These are not suitable for unit testing, as those tests would then test the TensorFlow method called, and not the method written by the team. External code should not be included in unit testing, as it cannot feasibly be changed on the unit level.

Unit tests for this project were written in the Python standard library for testing. These tests call methods in the code with a certain input, and check that the output matches the expected result.

Component Testing

Component testing tests how some part of the program consisting of several units perform. One can test a component independently of other components by replacing the surrounding components with simplified imitations.

The component test used in this project is testing the accuracy of the neural network. The reason this project was started was to automate part of a manual task.

If the resulting delineation is too inaccurate, the program does not have a purpose. Therefore accuracy is a very important metric for the project. The testing done against delineated cross-sections is done by TensorFlow. Tests done with non-delineated cross-sections have to be manually assessed.

Integration Testing

The goal of integration testing is to make sure the developed product works as intended in conjunction with surrounding systems and software.

These tests are relevant because the program was developed into a plug-in for Petrel. It is important to make sure the geological data is correctly sent between Petrel and the neural network, and that the interpretation is correctly transferred back to Petrel. However, the data itself is not transferred directly. The data is temporarily stored in text files, and read by the other program. Therefore, integration tests were not included in the test efforts of the project.

If integration tests were included, they would simply check the transfer of data between Petrel and program. These tests would be difficult to automate, since as far as the team could find, there does not exist any suitable framework for this task. The tests would include the steps for running the program, and how to check that the transfer of data was successful.

System Testing

Testing done on the system as a whole is covered by system testing. This should be conducted as black box testing, where the tester does not know how the system works internally, only what its inputs and outputs are. There are many different tests one can use in system testing. Performance and usability testing are covered in this report.

System Testing - Performance

Performance testing is used to test how well an item performs with time or resource constraints. How important such tests are depends on the environment and the task the program shall perform.

The program should use as little time and resources as possible, while having a high degree of accuracy. Accuracy is covered in the component tests. The time is measured in the performance tests. The performance tests were extracted from the quality attribute, and its scenario, see Section 7.1.2. The test is to check if the system can perform its delineation in 75% of the time needed for manually interpretation. How long this takes depends on the cross-section to be delineated. Some cross-sections contains more complex shapes of salt bodies, and requires more careful work to retain the shape of the salt in the delineation. See Figure 9.1 for an example of a complex salt body. The time needed for the cross-sections used in the test were estimated by the customer.

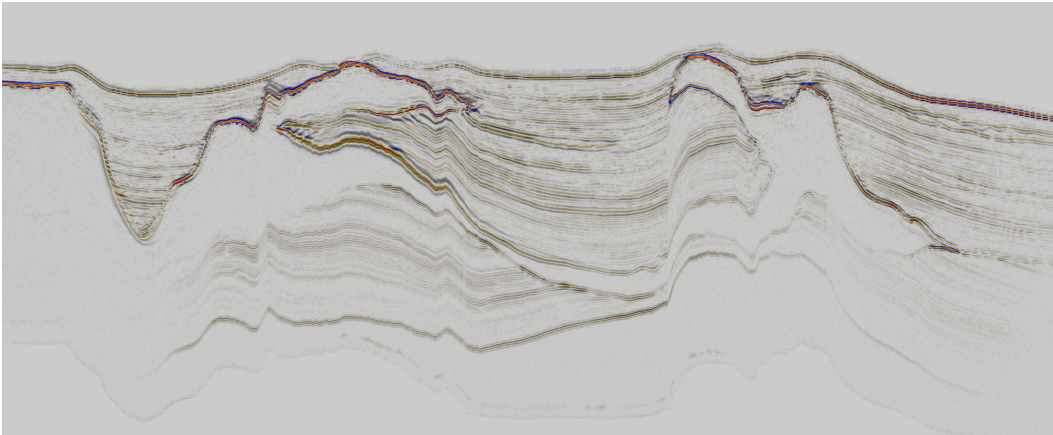


Figure 9.1: Cross-section with complex salt shapes

The performance tests are manual tests where the program is run on specified cross-sections, and the time it takes is noted. These tests are a little unreliable since the hardware affects the run time. In addition, the approximated time for manual delineation may be inaccurate.

System Testing - Usability

Usability testing is intended to test how well users can interact with the product. This is to make sure the program is understandable, and not more complicated than needed. User interaction is an important metric to use when developing software. If the software is too difficult or too time consuming to use it can lose a lot of its purpose.

The people involved in the development are not suited to assess the usability of the program they are making. Their mental model for the program tightly follows the development, and their understanding of the program is a lot higher than an end user starting to use the program. Therefore it is important to get feedback from people not involved in the development to see how easy the program is to learn and use in practice.

The part of the system where usability testing is relevant is the Petrel plug-in. For the team this testing would be difficult to execute. Petrel is a piece of very specialized software, and the team does not have access to suitable testers. In addition it would be difficult for the team to create good assessment criteria for the usability testing. In addition, the plug-in developed contains highly limited functionality, and the action the user can take is limited to exporting seismic data and importing a delineation. This is done by choosing an action from a context menu. Usability testing is thus not very applicable for the end product of the project.

Usability testing would be more relevant if the plug-in included more functionality.

The proposed functionality can be seen in the mock ups, see Section 11.2.1. Then the program would include more actions the user could take, and there would be more to test.

Usability testing of the system would be done with manual tests. The assessment would be done by external users, not related to the project. The team would describe the goals the user when running the program. Then it would be up to the user to assess how easy it was to reach their goal, and their experience using the software.

Acceptance Testing

Acceptance testing is used to see how well the finished product compares to the acceptance criteria developed in cooperation with the customer. This is the final test, and is not performed by the developers themselves. It is performed by the customer, and in many cases it determines whether the software will be used or not by the customer.

This project had many acceptance criteria, where some were directly connected to the envisioned Petrel plug-in. How the acceptance testing will cover the different criteria is up to the customer's discretion. The tests could be quantitative and/or qualitative. The running of the tests, and the assessment is conducted by the customer themselves.

9.3 Test plan

The test plan describes how the testing in the project will be conducted. The plan covers every planned test process for the project in its entirety. This includes what parts of the program that will be tested, and which test processes will be used.

Test Items

The testing outlined in this plan covers these parts of the product:

- Input Converter
- Classifier
- Post-Processing

Test Sub-Processes

The test plan for the project includes the following test sub-processes:

- Unit testing
- Component testing
- System testing

Test Deliverables

Each test sub-process produces these documents:

- Test Case Specification
- Test Completion report

Test Completion Criteria

The unit tests must cover 80% of the methods used, not including those that rely heavily on 3rd party libraries. The system test must cover 100% of the requirements.

Test Data Requirements

The system test requires a seismic cube with at least one cross-section.

Test environment requirements

The unit tests need a framework to run the code correctly, this is covered by the standard library "unittest" in Python. The component testing of the system needs feedback on its accuracy, this is supported by the TensorFlow framework.

Testing Activities and Estimates

The component tests used to evaluate the neural network accuracy are not covered in the estimate of test execution table 9.1, because they are a part of the TensorFlow framework, and its workflow.

Schedule

The unit tests should be developed in parallel with the development of the software. The system tests should be created after the implementation phase is finished. A final run of all the tests will be conducted in the testing phase, and will be the basis of the test completion reports.

Activity	Estimate
Specification of test cases	4 hours
Construction of unit tests	6 hours
Construction of component tests	2 hours
Construction of system tests	2 hours
Setup of test environment	2 hours
Test execution	1 hour
Test completion reporting	2 hours

Table 9.1: Testing Activities and related estimates

9.4 Test Case Specification

The test case specification document gives details about a single test case. A test case covers the necessary information needed to perform a test of the system. The parameters chosen for the test cases in this project can be seen in table 9.2. The actual test case specifications can be seen in the appendix D.

ID	ID for the test starting with the letter for the test type
Objective	What the aim of the test is
Preconditions	What is required for performing the test
Inputs	The inputs for the test
Expected results	What the results should be given the input
Actual results	What were the actual results
Test result	Did the test pass or fail

Table 9.2: Test case Template

9.5 Test Completion Reports

The test completion reports are documents showing the results from the testing conducted. In addition to the test result, they include reflections about the testing conducted. There are reports for each of the test sub-processes used: Unit, component and system tests.

9.5.1 Unit Tests Completion Report

The unit test sub process ended up involving seven different unit tests.

Deviations from Planned Testing

The unit tests were initially intended to be developed in parallel with the system. However, the focus during the implementation was to get a working prototype so

the testing was put on hold. They were instead developed during the later stages of the implementation.

Test Completion Evaluation

All of the planned tests passed. The unit tests covered only a small part of the total code. The part covered was the post-processing of the delineation. The rest included too much usage of third party libraries, or methods dependent on other methods. The test coverage was lower than intended, but the parts of the code suited for unit tests were included.

Lessons Learned

Creating the unit tests alongside the code can lead to less work overall than doing so at a later stage.

9.5.2 Component Tests Completion Report

The component test sub process ended up involving two different component tests.

Deviations from Planned Testing

The unit tests were performed as planned.

Test Completion Evaluation

Both of the tests passed. The quantitative test (see Table D.8) had a high accuracy (97.66%). This is in part because the test only checks for correct labeling without taking into account that there are far more negative than positive examples. There were few tests developed, because the team had very little access to labeled data. The team used all the labeled data as training data, except one cross-section used as a validation set. Therefore, there did not exist a labeled test set. To perform a proper test of the neural network the team needed to perform a qualitative test. This is less accurate than a quantitative test with labeled data. Table D.9 shows the result of the qualitative test. The assessment of the result is up to the discretion of the person doing the evaluation. The edge is almost entirely continuous, and there are a manageable amount of false edges.

The team did not see the lack of tests of accuracy as a big issue since the accuracy of the neural network will increase with a larger training set. Therefore, the accuracy at the end of development does not reflect the true potential of the product.

Lessons Learned

It is difficult to test a neural network extensively with a small set of labeled data. Qualitative testing suffers from tester bias. Accounting for the proportion of negative examples compared to positive examples would create a test more representative for the accuracy of the network.

9.5.3 System Tests Completion Report

The system test sub process ended up involving two different system tests.

Deviations from Planned Testing

The system tests were performed as planned.

Test Completion Evaluation

All of the tests passed. The tests were created to illustrate that the complexity of cross-sections does not affect the run time to any significant degree. In actuality the only aspect of the input that significantly affects the time taken is the size of the cross-section. Post-processing also takes longer based on the total amount of pixels identified as salt, but this is only a tiny part of the overall run-time.

Lessons Learned

The metrics for the tests are based on how fast a user delineates a cross-sections with existing methods. This is not deterministic, and relies on an estimate. However the results were so much quicker than the estimated manual interpretation, that the system passed the performance goal without a doubt.

10 Implementation Phase

The implementation spanned seven weeks. During this time, a Python program consisting of a neural network and various functionality for processing seismic data was created, as well as a Petrel plug-in to provide the network with data and show the user the results. Section 10.1 present the milestones. The remaining subsection shows the work done in that week, starting with week 38 and ending with week 44. Each week consists of an implementation part and retrospective.

10.1 Milestones

To aid the implementation of the project, the team periodically created milestones to serve as goals. These were determined on a week by week basis, revolving around the meetings with the customer. The milestones thus served as a way to ensure both the team and the customer were aware of how the project should progress between each meeting.

Across the seven weeks of the implementation, one such milestone was created each week:

- Week 38 - Finish research phase and begin implementation
- Week 39 - Implement rudimentary neural network
- Week 40 - Extend neural network to have a multi-layer structure, and implement rudimentary post-processing
- Week 41 - Move network to single-channel input. Improve the performance of the network. Learn about Ocean and compile an example Petrel plug-in
- Week 42 - Create a GUI mockup for a Petrel plug-in. Create a visualization of network architecture. Add the ability to save and load networks
- Week 43 - Load the neural network results back into Petrel and wrap up development
- Week 44 - Implement direct communication between the Petrel plug-in and the Python program

10.2 Week 38

Going into the first week of implementation, the team's main goal was acquiring an understanding of the TensorFlow framework and neural networks in general, as well as creating a simple proof of concept to show the customer.

Implementation

In order to learn more about neural networks and TensorFlow, all members of the team spent time installing TensorFlow and other required software, and then setting up a sample neural network to ensure everyone had a basic understanding of the concepts involved.

The team acquired the public SEAM dataset[32], which is a synthetic seismic cube designed to closely approximate real geological conditions. This was used throughout the project for testing the solution.

Towards the end of the week, the first version of the solution was created; a basic single-layer neural network that did only a middling job at the task, intended mainly as a proof of concept. At this point, a way to directly export seismic data from Petrel had not been found, so screenshots from the software combined with a manual delineation of the salt body were used for the training and test data. Considering these limitations, the results were quite promising at the time.

Figure 10.1 shows an example of the input and output of this initial version. The black line on the input image was the salt delineation the neural network was supposed to find. It was trained on four different seismic slices from the same seismic cube. Note that the actual visualization was not implemented until the following week.

Retrospective

Weekly retrospectives were not introduced until week 39. Week 38 served as a transitional week; going from research to implementation. A significant part of the week was spent on learning about the TensorFlow framework rather than coding, making the week a mix of both research and implementation.

As a result of being in such a transitional phase, the organizational aspects of the project were in a mild state of disarray. Combined with the actual implementation not starting until later in the week, this led to there not being a retrospective. It was instead decided that retrospectives were to start the week after, assisted by the introduction of templates for the meeting documents.

10.3 Week 39

In week 39 the main goals were visualizing the neural network results, and improving the quality of the output by deepening the network.

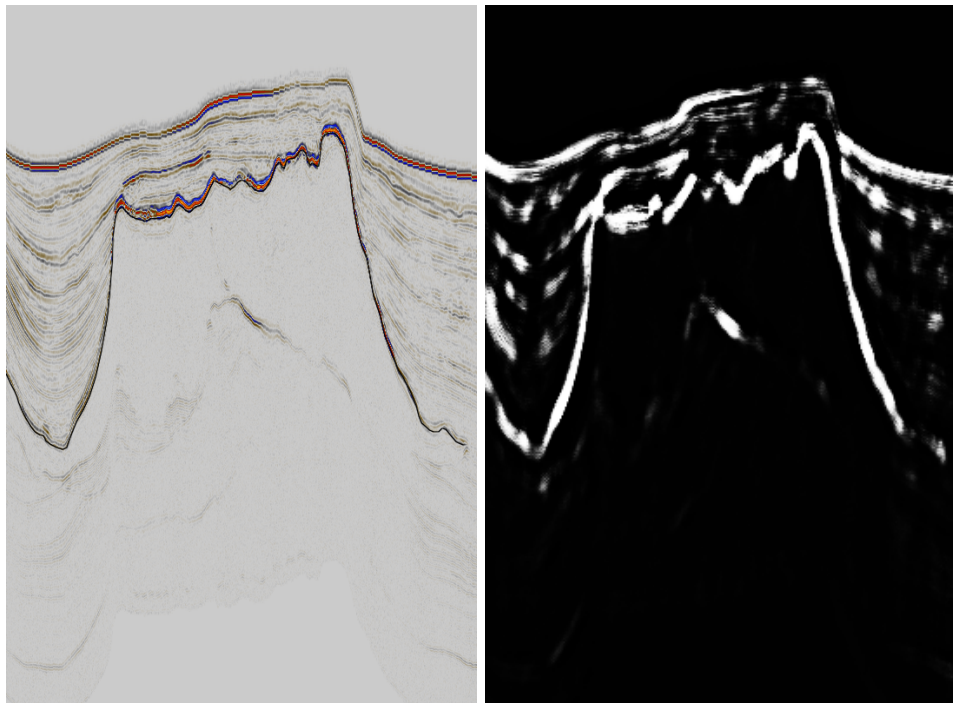


Figure 10.1: Early input and output

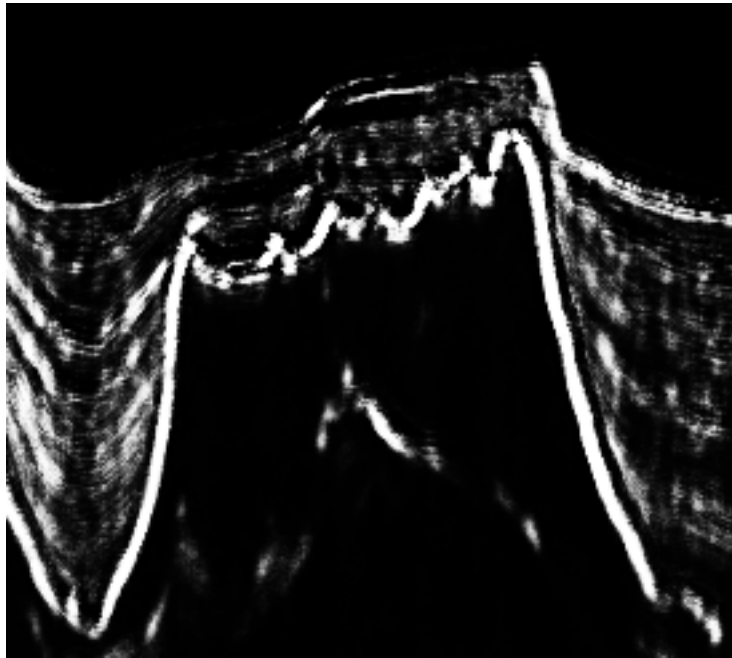


Figure 10.2: Early output

Implementation

The first order of business was to visualize the results from the neural network prototype so that it could be presented to the customer. This was achieved in time for the week's customer meeting (example output can be seen in Figure 10.1). It was then decided that the next goal should be to further pursue neural networks by making the structure more complex.

By increasing the complexity of the network's structure and introducing convolution layers the network was able to create better results, though at the loss of significant performance; instead of training taking half a minute, now it could easily take five or more. The sheer size of the network made running it at full scale infeasible without optimization, so it was instead run at half scale, meaning every other row and column in the input image simply was not classified. Still, the results were clearly better than before, as can be seen in Figure 10.2, which used the same input data as in Figure 10.1. As a temporary measure, the top and bottom of the slice were simply not run through the neural network since the result would be the same in those regions every time: no chance of salt. This cut a couple of minutes off the processing time, thus allowing more extensive testing.

Retrospective

The implementation of meeting minute templates helped ensure that all important topics got covered, and that the meeting notes would be easier to use when later writing the report. This also shortened meetings as less time was spent trying to figure out if any topic had been forgotten. Significant progress was made on the code, and the implementation milestone for the week was more than met. Group dynamics had by this point been clearly established, and everyone pulled their share of the load.

New insights included that asking new questions during the customer meetings was rarely effective unless the questions were very simple. For more complex questions it was decided it would be better to think them through, get them down on paper, and handle them over email. This was in part due to somewhat of a language barrier; there were instances where one party had trouble making themselves understood by the other as a result the accents of the team and the customer's representatives. This week also underscored how important it was to send any questions to the customer as soon as possible; even more could have been achieved if questions were always sent on the same day an issue occurred.

Future improvements would include better preparing for the advisor meetings, as so far they had not been particularly useful due to the team having very few clear questions prepared. Another point of improvement was to start including report writing in the project schedule to ensure progress continued to be made at a high enough rate.

10.4 Week 40

For the third week there were three separate goals: improving the code documentation, adding a post-processing pass to the network, and moving from using screenshots as input to using the seismic data directly.

Implementation

In order to make it easier to understand the code, variable names were revisited and descriptions were added to most functions describing what their purpose was. This made it easier for programmers on the team to work on code they had not personally written.

Furthermore, as the output from the network was quite noisy, a post-processing step was coded. This consisted of a double threshold. First all pixels where the output was low (below 66.7% chance of salt) were eliminated, and those where the output was high (above 99%) were kept. Any remaining pixel connected to a high output area was then kept. This eliminated much of the non-salt output, while preserving the actual edges. The result of this can be seen in Figure 10.3.

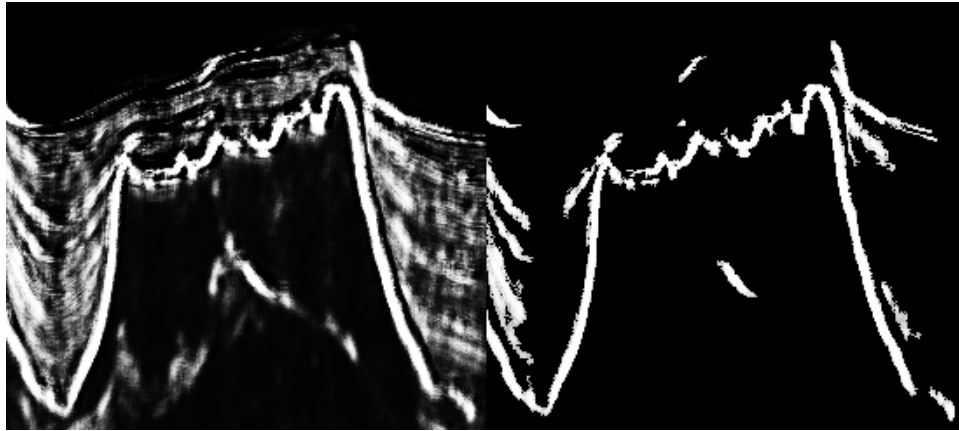


Figure 10.3: Left: Original output. Right: Post-processed output

The biggest piece of progress was moving away from using screenshots as input, and instead using seismic data directly. For one, this significantly reduced memory use, as now there was only one channel of input per pixel rather than three, and secondly it had the potential to significantly reduce overfitting as it is more difficult to overfit when there are effectively fewer inputs (in this case, 3 times fewer for the same window size). However, the results (see Figure 10.4) were not at a satisfactory level in time for the week's customer meeting, so presenting it had to be delayed to the week after. After the meeting however, the solution was improved by downscaling the input data before running it through the network. For each 2 by 2 pixel area, the pixel with the greatest absolute value was kept, thus preserving the most important data while effectively doubling the window size of the network. Combined with more training epochs, this ended up producing output more accurate than the three-channel network. See Figure 10.5 for an example.

The customer provided the group with Ocean-project files for creating a basic plug-in for Petrel, to help the team on the way to creating a plug-in capable of importing from and exporting to the Python program. These project files were set up and compiled successfully. The project files contained C# code which implemented an item in the context menu for right-clicking a seismic cube in Petrel. The item in the context-menu was called *run neural network*, though at this time it had no actual functionality. At this stage, the plan was that when it was clicked, the system should go through the cube cross-section by cross-section, and running the cross-section through the network and showing the result in Petrel.

Retrospective

The team was focused and productive, even though the solution was not quite at the stage planned for the week's customer meeting. This focus manifested in both

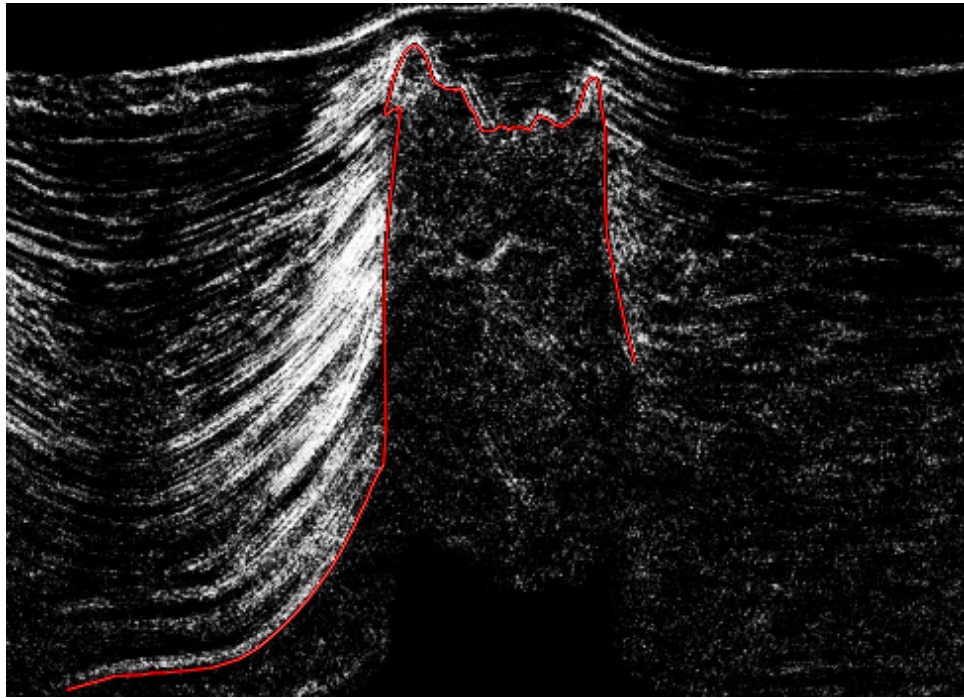


Figure 10.4: NN output from single-channel input

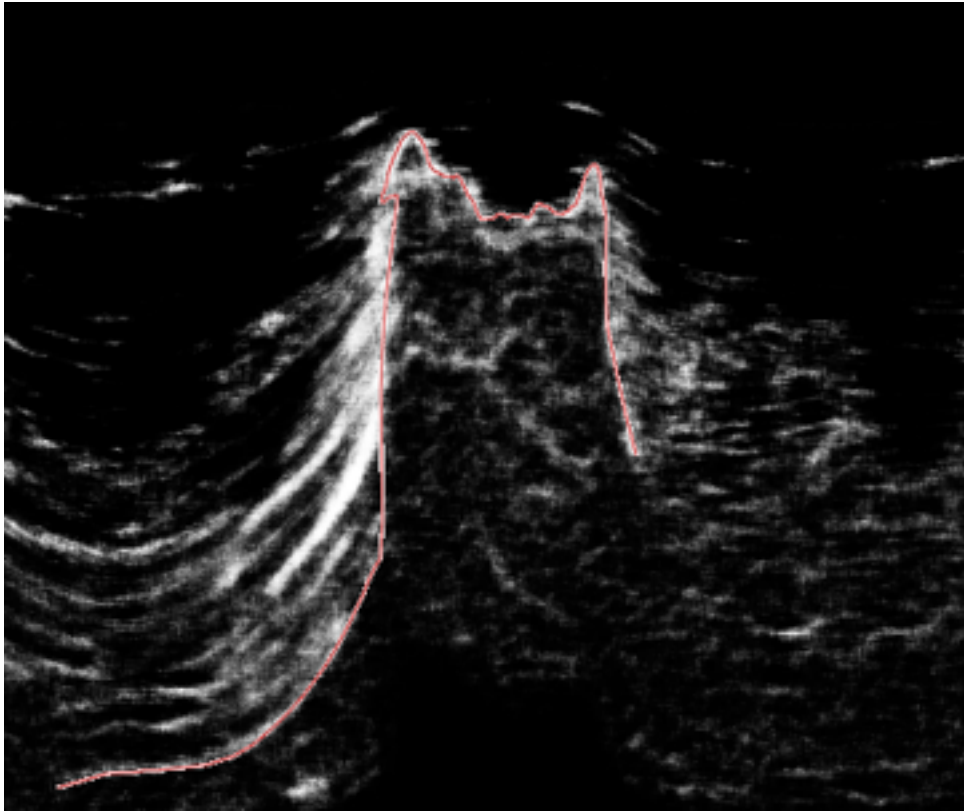


Figure 10.5: NN output from downscaled single-channel input

good progress being made on the report, and the prototype quickly being updated after the customer meeting to the state originally planned. The progress made on the two fronts helped ease the team's minds.

The use of meeting templates continued to be a boon to the team, ensuring effective meetings and that all topics were sufficiently covered. The previous week's introduction of weekly retrospectives helped address a few issues encountered that week, thus reducing the chance of frustration caused by recurring problems.

The team however again failed to prepare well for the advisor meeting. To improve on this, the content that should be covered in the meeting should be pre-planned. It was also decided that the advisor should be provided with a copy of the first draft of the report so that they could provide feedback. This was planned to be done at the end of week 43 or the start of week 44. It was also decided that the meeting before each customer meeting should always be used to plan the customer meeting so that no important topics are missed.

10.5 Week 41

This week, the team's main goal was to learn more about the Ocean framework and get a simple prototype working, as well as adding save functionality to the network.

Implementation

Changes to the core of the program (the neural network and supporting functionality) was limited this week; much of the focus was changed to integrating it with Petrel via Ocean. Work on the core was limited to some tweaking, refactoring, and save functionality.

The neural network was tweaked further, resulting in marginal improvements to its result (see Figure 10.6 for an example), though this also further increased the training time.

Most of the project code was refactored to make it easier to understand and maintain. Most significantly, the neural network itself was moved to a class-based structure rather than being set up and interacted with procedurally. This allowed reducing the number of parameters needed in functions, as these could now simply be fetched from the network object instead, and also allowed moving most of the code into functions, leaving the network's main function at under twenty lines compared to over a hundred before the refactoring.

Save and load functionality was implemented for the network, making it possible to save a trained network and later reload it. This was both useful for testing and the final product. Testing became easier as now a new network would not have to be generated every single time, while being able to store a network was critical to the

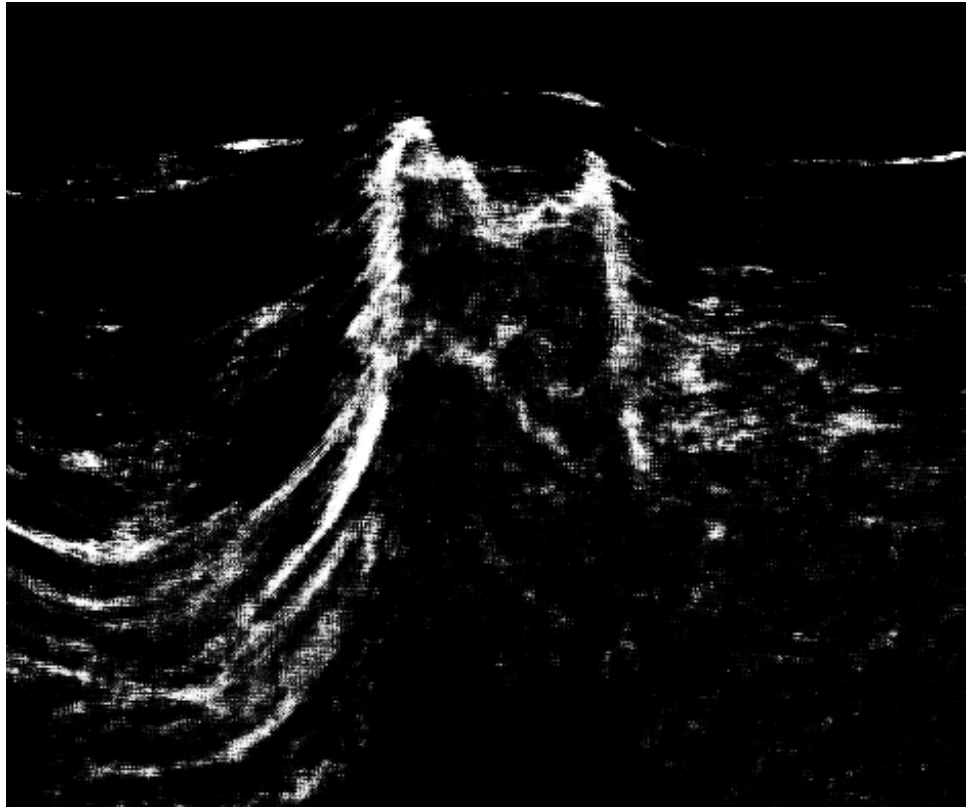


Figure 10.6: NN output

final product as having to train the network separately for every single run would have been a massive waste of time, and would significantly constrain the ways in which it might be utilized, ruling out possibilities like doing the training on a more powerful machine than the machine running the actual interpretation.

A lot of time was used reading about how to import and export seismic data to and from Petrel in a format that the neural network could read. This functionality was not implemented in week 41, and should be implemented in week 42.

The development of the plug-in did not progress particularly, and most of the time spent was used on reading more in Ocean-manuals, and understanding the code provided by the customer.

A mockup was created showing a potential user interface for the Petrel plug-in, including both planned functionality and theoretical future functionality. This can be seen in Figure 10.7. It is planned to implement the "run neural network" function, and if time allows it, the "mark area" function. The context menu shown in the figure would show up when right-clicking a seismic cube. For more detailed description of the mockup elements, see Section 11.2.1.

Retrospective

Efforts were made to decompose report-related tasks further so as to make it easier to observe progress, and to distribute tasks evenly between members of the team. This resulted in a good amount of progress being made towards the report. Planning the customer meeting the day before was quite successful, and ensured all important topics were covered with minimal confusion. In the past there had often been confusion over what team member should speak at any given time, but this was not an issue this time around.

Refactoring the code made further development simpler, and the code easier to understand. The project leader's participation in the bi-weekly project leader meeting made the team more aware of how despite the project task being one of the more complex ones, the group dynamics were solid; other project leaders shared issues with attendance, completion of delegated work, and scheduling conflicts. In comparison, the team has had no issues with attendance; every case where someone was unable to attend they notified the rest of the team at least a day in advance, and with only a few exceptions participated in the meeting over Skype for Business or appear.in. Scheduling of meetings was also mostly a non-issue; even with three weekly meetings there were very few scheduling conflicts. Completion of delegated work was also rarely an issue; when someone got a task they generally completed it by the time they said they would, and to the expected level of quality.

Hearing about the challenges other teams had experienced helped reinvigorate the team's resolve to overcome the complex problems they were faced with.

Salt delineation
Run neural network
Mark area
Add to training
Train neural network
Edit training set

Add to training		
Name	<input type="text"/>	
Weight	<input type="range"/>	<input style="width: 20px; height: 20px;" type="button" value="?"/>
Marked area	<input type="checkbox"/>	<input style="width: 20px; height: 20px;" type="button" value="?"/>
Delineation input	<input type="button" value="→"/> <input type="text" value="Interpretation"/>	
Skip undelineated slices	<input type="checkbox"/>	<input style="width: 20px; height: 20px;" type="button" value="?"/>
<input type="button" value="Cancel"/> <input type="button" value="OK"/> <input type="button" value="Apply"/>		









Edit training set			
Training set	Inspect	Toggle	Delete
SEAM		<input type="checkbox"/>	
F3		<input type="checkbox"/>	
.....		<input type="checkbox"/>	
.....		<input type="checkbox"/>	
<input type="button" value="Cancel"/> <input type="button" value="OK"/> <input type="button" value="Apply"/>			

Figure 10.7: Mockup of Petrel plug-in

It was decided that the secretary and architect were to switch roles, as the (former) secretary had a better understanding of the architecture of the program.

10.6 Week 42

The main goal was to further progress with the Petrel plug-in prototype, as well as further improving the maintainability of the codebase.

Implementation

In order to further improve the code base, type hinting was added to nearly all functions. Using an IDE (such as PyCharm), this provides auto-completion of function names and object properties, as well as checking whether the inputs and outputs of the function correspond with the type hinting. Additionally, short descriptions were added to each function, describing their purpose. These two steps should make further development and maintenance of the system simpler.

Beyond that, a bug was discovered where points with negative magnitude would be suppressed by the pre-processing of the neural network input, meaning the network had less information to work with than intended. Fixing this slightly improved the output of the network.

The plug-in development finally made some significant progress, and on Tuesday the plug-in was able to write a cube to a SEG-Y file. It turned out it was difficult to convert this file into cross-sections, and that the file size of approximately 4GB was unreasonably high, so the solution was found unfit. There was no API available in Ocean to export single cross-sections to SEG-Y files.

At Thursday, after a meeting with the customer, new project files based on the mock-up GUI were sent to the group, and these were set up and compiled successfully. The plug-in development made a lot of progress over the weekend, and writing of cross-sections to text files was implemented. The project files included code that implemented a sub-menu in the context-menu for a seismic cube, and this sub-menu had three buttons: *Run neural network*, *Mark area* and *Add to training*. The *Run neural network*-button should make the system read through each cross-section, output it to a text-file, call the network, wait for the network to complete, and then read the result back in. Further it should create a set of points to be shown on the cube. The *Mark area*-button should let the user select an appropriate region in the cube, a *sub-cube*, of where to look for the salt, before running the neural network on this part of the cube. The *Add to training*-button opened a menu where the user could insert a salt delineation as a *polyline* (a kind of line used in Petrel), and this would be added to the training set for the neural network. The *Add to training*-menu was implemented by the customer but with no functionality. The context menu can be found in Figure 10.8, and the dialogue box can be seen in Figure 10.9.

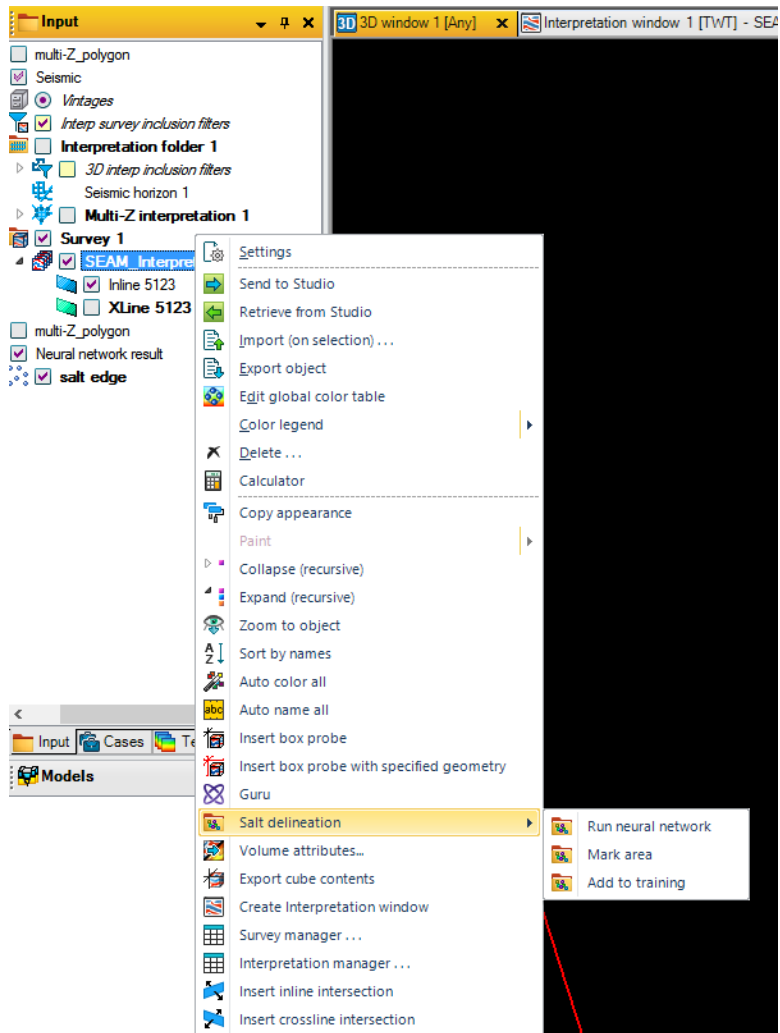


Figure 10.8: A seismic data context menu in Petrel

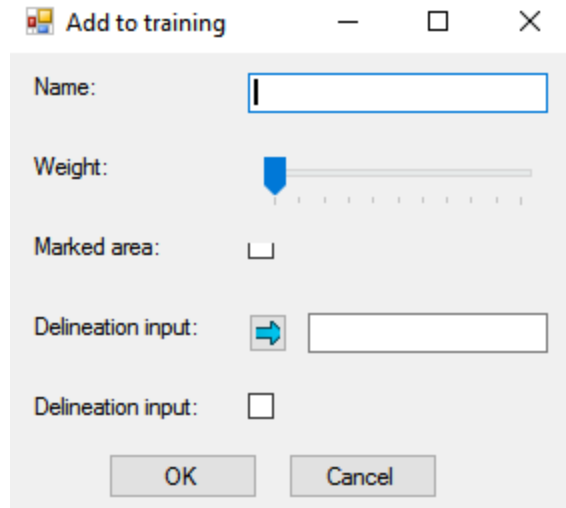


Figure 10.9: A dialogue box in Petrel for adding training data to the training set of the neural network.

Retrospective

Due to much of the focus having moved to the report rather than the program, progress was not as significant as it might have been. Still, the week's goals were achieved. The code base was now more readable than ever, and a Petrel plug-in proof of concept was created.

Swapping the role of secretary and architect worked well; good progress was made on the architectural aspects of the report as a result.

The advisor meeting was more useful than usual, thanks to the team having specific questions prepared. This gave the team more insight into what was expected from the report.

It was clear that in the final weeks of the project, it would be even more important for team members to be clear on when they would complete any given task, as time pressure now meant that failure to finish a task could now have far more severe consequences than earlier in the project. The main step taken to avoid this potential problem was a new rule that team members should never say they would do more tasks in a given time frame than they were absolutely certain they could manage, thus making minimum progress far more predictable.

10.7 Week 43

For the last planned week of implementation the main focus was to be able to load the neural network results back into Petrel and wrapping up development.

Implementation

The program was extended to output a file consisting purely of the coordinates of the points the neural network and post-processing had determined were part of the salt edge. This could then be read into Petrel via the Ocean plug-in. It was also updated to be able to read the seismic data output from the Petrel plug-in.

Plug-in development progressed further with the implementation of reading coordinate files from the neural network. Petrel has two different types of coordinates, one for the project space itself, known as global coordinates, and one for coordinates inside a seismic cube. Because of this, when reading the result from the neural network, the set of points has to be converted into global coordinates to be shown at the correct position. The ability to show these points in Petrel was also implemented. To find the local coordinates, built-in functionality in the Ocean framework was used. Without this step, the points could end up being shown far away from its corresponding cross-section. An example of this can be found in Figure 10.10.

In order to let the user decide if they want cross-sections to be sent to the network in inline or crossline direction, the possibility to read in either or both directions was added. The GUI does not yet have buttons to select this, but this can be changed in the plug-in's code. Further, the plug-in also supports the user selecting how many cross-sections to skip between each cross-sections to be sent to the neural network. This was implemented as the user will not always need a dense result, wanting instead to sacrifice density for quicker results. This is not yet supported in the GUI, but can be changed in code. This could have been implemented as a slide-bar in the GUI but system integration of the plug-in and the neural network was prioritized.

Because the coordinates of the traces in Ocean are not the same as the coordinates in Petrel, new training data based on Ocean's coordinates had to be produced. The delineation was reused and only re-named, but the seismic data had to be exported in the new format based on the new coordinate system.

To show the customer how the system looks and works, some specific methods for reading a few specific results from the network were implemented. These were used in a demo at the customer meeting on Wednesday. Figure 10.11 shows a similar result to what was shown in the demo.



Figure 10.10: A set of points showing the salt delineation for a given seismic cross-section, but in the wrong coordinate-system, making them appear in Petrel so far away from the cube that the cube itself is not visible when looking at the points

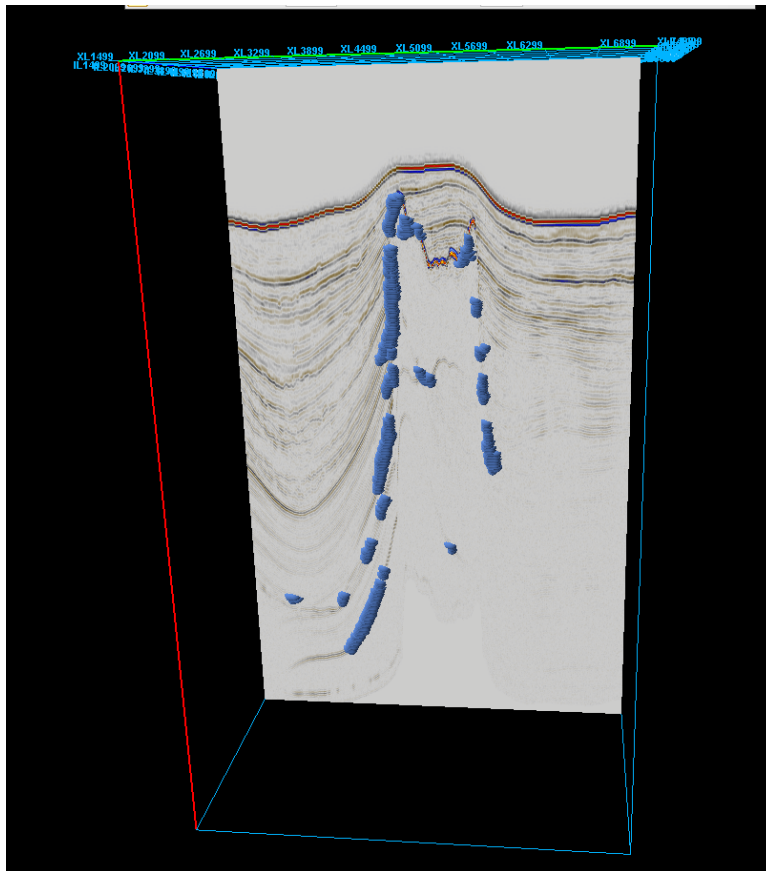


Figure 10.11: A cross-section with the delineation result from the neural network shown as a set of points in Petrel.

Retrospective

During the week it was decided that one last week of implementation was needed to get the solution to a state the team would be happy with. The system at this point simply was not smooth enough for an end user to use; they would have to run the Python aspects and Petrel aspects separately, while the team preferred for the Petrel plug-in to call the Python program without the user needing to be aware of the Python program's existence.

The customer stated they were satisfied with the product, but were especially happy to hear that one last week of implementation would be undertaken.

Communication between the programmers was especially good. Earlier little co-ordination had been needed, but now the Python program was reading data the Petrel plug-in output, and vice versa.

The team had hoped to have the first draft completed by the time of the Friday meeting, but this turned out to not be doable. However, at the Friday meeting significant process was made, and nearly the whole report reached first draft status. The extra week of implementation made the report writing more difficult, as sections like the documentation were thus harder to write.

It was decided that it was especially important to ensure all tasks were delegated in the few remaining weeks.

10.8 Week 44

The team wanted to further improve the product, and therefore underwent one last week of implementation. The goal for the week was to achieve direct communication between the Petrel plug-in and the Python program so that the end user would not have to run the two separately.

Implementation

A lot of effort was put into integrating the neural network with the plug-in, but no solution was found. Because TensorFlow only supports UNIX, and Petrel only supports Windows, communication between the two components was a lot more complicated than expected.

Many different approaches were tried, like starting a *Windows Subsystem for Linux* process via C# code and making this code run the neural network, and creating a batch-script that would run the neural network. None of these approaches worked, so the team arrived empty-handed at the customer meeting, instead presenting the first draft of the report.

Retrospective

Most of the team worked on the report, as this had to be prioritized at this stage. The plug-in developer was mainly focusing on system integration but was not able to get that functionality into the system.

Work on the report progressed further, and by having even more fine-grained delegation of tasks the team was even more productive than before. A first draft review was conducted, and a lot of improvements for the report came up. This made further delegation of tasks easy.

Beyond this point, the team focused exclusively on the report, making this the last retrospective held.

11 Result

The team produced three main results: A neural network, a Petrel plug-in, and an outline for further work on the solution. The neural network and the Petrel plug-in combined was named NetSalt. Section 11.1 shows the results from the neural network implemented as a python program. Section 11.2 presents the Petrel plug-in used to visualize the results. Section 11.3 is an outline of how the customer can develop the solution further.

11.1 Neural network

The primary result of the project was a Python program implementing a neural network for delineation of salt, along with supporting functionality handling importing and exporting seismic data, pre and post-processing of the data and results, and saving and loading of networks. This provided the ability to automatically delineate salt bodies within a seismic cube. However, there were a number of caveats relating to quality and performance.

Included with the Python program was a network pre-trained for 20 000 steps, as well as delineated training and test data.

11.1.1 Quality

The quality of the output is simply how well its delineation corresponds with the actual boundaries of the salt body, as defined by the level of false positives and false negatives. False positives can be more severe than false negatives, as the subsequent manual interpretation then has to include figuring out which parts of the data are spurious, while for false negatives one simply has to fill in the resulting gaps.

The quality of a neural network is largely limited by the quality and variety of its training data. The team's network only had training data from a single seismic cube, and thus would be unlikely to work well outside that cube. This would likely be alleviated by a more varied set of training data.

It was early discovered that for the network output to be reasonable, the training data had to be skewed significantly towards input windows containing a salt edge by skipping a large portion of negative examples, as the vast majority of a seismic cube will not contain the edge of a salt body. Without such a skew there would be so much negative input data that the best course of action for the network would be to simply claim that *nothing* was salt, since that would be correct in the vast majority of cases. The team therefore skewed the input so that there was a roughly equal amount of positive and negative examples.

In order to produce good results, significant time has to be spent on training. The more training is applied, the more accurately the network is able to classify the

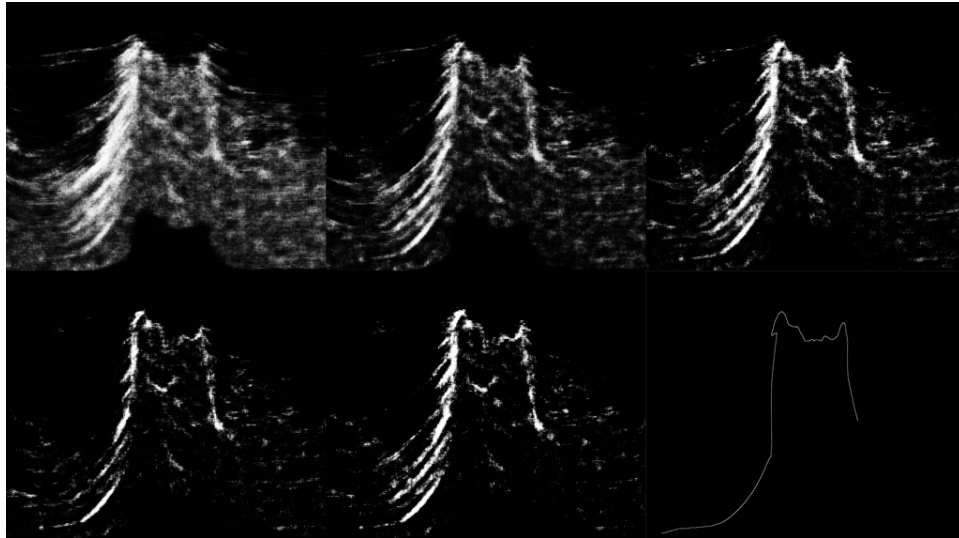


Figure 11.1: The effect of training on the neural network results compared to a manual delineation

training data. However, if trained too much this can cause spurious relationships to be established that happen to be accurate for the training data but not unseen data. A good middle-ground therefore has to be found. Figure 11.1 shows the impact of training on the neural network results. Each successive image has been trained for 5000 more steps than the previous one, starting at 5000 steps, while the last image shows a manual delineation. One can clearly see that more training improves the result until around 20 000 steps, after which the network becomes increasingly inaccurate. To reduce overfitting, the network utilizes dropout during training, as explained in Section 3.4.

The network outputs the set of points it estimates to be part of a salt edge. This set of point is not necessarily contiguous, though the post-processing used does reduce the incidence of non-contiguous false positives. This can then be imported into Petrel using the team’s plug-in.

Due to the sheer amount of relatively uniform space in a seismic cube, the accuracy of the neural network is difficult to quantify. A network simply outputting ”not a salt edge” for every single pixel would have an accuracy (as defined as the portion of the data correctly classified) well over 99%, yet clearly be producing a terrible result. The team’s network had an accuracy of 97% (see Table D.8 in the testing Appendix) as some false positives were allowed for; most of these false positives manifested very close to the real edge, and thus did not significantly detract from the result. Much of this could be eliminated by an interpreter by thinning the edge, something that would be simpler than doing manual delineation from scratch.

Overall the quality of the result was very promising in the team's tests, but time limited those tests to only a single seismic cube. As discussed in Section 11.3, there is significant potential for further improvements to both the network itself, and to how it is post-processed.

11.1.2 Performance

In order for the network to be useful, it has to be possible to run it in a reasonable amount of time. Due to the sheer number of points to be classified, the run-time of the network is significant.

In the team's tests, running a single seismic cross-section through the network took 8 to 9 minutes on an i7 4770K CPU. This means that if one were to classify every 10th cross-section in both axes in a cube 1000 by 1000 traces large, it would take 1000 to 2000 minutes; 16 to 33 hours. Effectively, this means that one would have to either distribute the work over several computers in order to get a classification to this level of detail completed between two workdays, let it run across a weekend, or reduce the number of cross-sections classified.

Another aspect of performance is the training of the network. 20 000 steps of training, which is what provided the highest level of quality in the team's tests, generally took the team about an hour to complete. Compared to the time involved in classifying a seismic cube this is insignificant, but it does mean that any tweaking of the network is time consuming, slowing down further development.

Overall, the performance is good enough that the network could be used in practice using multiple computers, but it is slow enough it cannot effectively be used *during* a workday except for individual seismic cross-sections. As mentioned in Section 11.3, there's a number of ways in which the performance of the network could be improved, both in regards to training and in regards to classification.

11.2 Plug-in

For exporting seismic data to the neural network and visualizing the result of the salt delineation, a plug-in for Petrel was implemented. Its user interface was based on a mockup sent to the customer.

11.2.1 Mockup

In order to showcase what the team might be able to create, and what might be developed in the future, the team created a mock-up of a Petrel plug-in. This included both planned functionality and functionality the team clearly would not have the time to implement. The customer created a framework for a plug-in based on this mock-up, and sent this project to the group for further implementation and

integration with the neural network. The mockup can be found in Figure 11.2. In the coming paragraphs, the functionality of the plug-in is explained based on the mockup.

Clicking on "Run Neural Network" would run the network on the currently marked area in the seismic cube, or the whole cube if nothing is marked, outputting a salt delineation.

"Mark area" would be used to mark which part of the cube to delineate so that time wouldn't be wasted on areas that clearly have no salt.

"Add to training" would open the window seen in the second part of the figure. With this, one would define how heavily weighted the area should be, allowing certain sets of data to be considered more important, e.g., due to being an especially complex area, or especially good delineation. This could be achieved by including more training data from more heavily weighted areas. The "marked area" checkbox would determine whether the whole cube or the currently marked area should be used. The "delineation output" would be the multi-Z delineation (or similar) used for the interpretation. "Skip undelineated slices" would instruct it to only use slices with at least some delineation on them for training; ideal for situations where only a subset of slices have been manually delineated.

"Train neural network" in the context window could be used to train a new network or further train the existing network.

"Edit training data" in the context window would open the last window shown in the figure. This would allow deleting specific training data, or disabling data temporarily, as well as viewing the data (would probably open a seismic cube consisting of the training data and delineation).

11.2.2 Reality

Because of the limited time frame of this project, most of the functionality from the mockup in Section 11.2.1 was not implemented. Further, because of the problems with integrating the plug-in with the neural network, using the plug-in is generally not a user-friendly experience. It requires editing the code, manually running the neural network for each cross-section, and renaming the files from the result, to be read back in to the plug-in and to be visualized in Petrel. The only button that has any function is the "run neural network"-button, but changing parameters like which cross-sections to be exported has to be done in code.

11.3 Potential improvements

While the team was able to produce what they believe to be a decent solution, there are still many clear avenues for improvement, both to the network itself, and to the Petrel plug-in, as well as to the communication between the two.

Salt delineation
Run neural network
Mark area
Add to training
Train neural network
Edit training set

Add to training		
Name	<input type="text"/>	
Weight	<input type="range"/>	<input style="width: 20px; height: 20px;" type="button" value="?"/>
Marked area	<input type="checkbox"/>	<input style="width: 20px; height: 20px;" type="button" value="?"/>
Delineation input	<input type="button" value="→"/> <input type="text" value="Interpretation"/>	
Skip undelineated slices	<input type="checkbox"/>	<input style="width: 20px; height: 20px;" type="button" value="?"/>
<input type="button" value="Cancel"/> <input type="button" value="OK"/> <input type="button" value="Apply"/>		









Edit training set			
Training set	Inspect	Toggle	Delete
SEAM		<input type="checkbox"/>	
F3		<input type="checkbox"/>	
.....		<input type="checkbox"/>	
.....		<input type="checkbox"/>	
<input type="button" value="Cancel"/> <input type="button" value="OK"/> <input type="button" value="Apply"/>			

Figure 11.2: Mockup of Petrel plug-in

11.3.1 Network

The network ended up producing decent results, but was not especially quick. There are several ways in which both the results and performance could be improved.

Performance

On the performance side, the most obvious improvement would be to utilize GPU acceleration. GPU accelerated neural networks can be up to several times faster than networks that have to run on a CPU, as GPUs excel at parallel computations, and neural networks are relatively easy to make parallel. Such a performance increase would allow the program to be run overnight in a high level of detail, having a result ready by morning. This would also improve the training time, not just the classification time. The team briefly tested GPU acceleration near the end of the project, and found that it was roughly 20 times as fast on a high-end GPU (GTX 1070) as a high-end CPU (i7-4770K). The team was unable to find a way to run the network with GPU acceleration on Windows, instead running it on Ubuntu. As Petrel is Windows-only, this makes it difficult to transfer the data and results between the network and Petrel.

One way to utilize the sheer speed of GPUs would be a server-based solution where the neural network runs on a different machine from the one running Petrel. The actual data involved takes far less time to transfer than it takes a GPU to process it, so by offloading it to specialized machines significant time could be saved. The main concern involved is the fact that seismic data is highly valuable, so companies would most likely want to host these servers themselves so that they would not have to share their seismic data in any way.

The training time could likely be improved by using a decaying training rate. The final solution simply used a constant training rate, but a network could likely be trained to equivalent levels of quality in a shorter period of time if the training rate started high and was gradually reduced over time. This also has the potential to improve the output somewhat, as towards the end a very low training rate can achieve minuscule tweaking to the network weights that simply cannot be achieved with a high rate.

A third step that could be taken would be to train several networks at different resolutions. Currently the seismic data is scaled down in both dimensions by a factor of two, as it would take four times as long to classify a seismic cross-section at full resolution. This concept could be utilized further by training networks at even lower resolutions to eliminate areas with virtually no chance of containing salt edges, thus allowing the slower but more accurate networks to only classify the areas that have a real chance of containing salt edges. This could be done in several layers so as to ensure high performance without being detrimental to the quality of the output.

Quality

There are also a number of changes that have the potential to further improve the quality of the result. The most traditional way to improve a neural network is to simply provide it with more training data, and more varied data. This makes it less likely for the network to end up overfitted to its training data, and teaches it a greater variety of situations.

It is possible an even deeper network might produce better result. Someone with a higher level of experience than the team with neural networks would likely be able to make structural changes to the network that further improve its output.

The post-processing of the network output could also be improved. The solution tends to create very thick edges, while it would be preferable for these to be thinned to where the actual salt edge is most likely to be. This can be achieved in a number of ways. One way would be to draw "best fit" lines through the output, weighted by the certainty of the result. This could also potentially be done with three dimensional polygons rather than lines.

It might also be possible to take into account the output from adjacent cross-sections; if an edge is almost certainly present in one cross-section, it is highly likely to also be present nearby in the current cross-section. In a similar vein, it might be possible to use seismic data from adjacent cross-sections as inputs, thus taking into account the three-dimensional nature of salt bodies.

11.3.2 Plug-in

The plug-in could be improved upon. The biggest immediate improvement would be to get the two components integrated into one solution, where running the network would be done automatically by just clicking "run neural network". This would significantly improve the user experience, as no longer would it be necessary to run the Python program and the Petrel plug-in separately. One way this might be achieved would be to run a web server on the UNIX (virtual) machine where the Python program runs, and using that to communicate with the Petrel plug-in.

Once this has been done, it would be possible to implement all the missing functionality from the mock-up. Namely the ability to training add data from Petrel, letting the user select which area to delineate, and how many cross-sections to skip between each delineation.

12 Evaluation and Reflection

Many aspects of the project went well, while other aspects did not go as well as they could have. Section 12.1 talks about the evaluation of the project work. This is divided into different section based on the phases in the project. Section 12.2 evaluates how the group worked together and present difficulties related to team work. Section 12.3 talks about the work with the customer. Section 12.4 contains suggestions for improvement for the course. Section 12.5 talks about the advisor.

12.1 Project work

The work itself had both positive and negative sides.

The task quickly turned out to be an especially challenging one, relying heavily on domain knowledge the team had no pre-existing understanding of. This was a problem throughout the project, though especially in the early phases. A lot of time was thus spent acquiring some of this knowledge, but the team's understanding remained incomplete. The team had varying experience with neural networks, image processing, project work, and software architecture. This meant that many tasks were divided based on previous knowledge, making task delegation more rigid than it may have otherwise been.

12.1.1 Research/planning

The team started off the first phase well. Three weekly meetings were quickly set up, ensuring everyone stayed in contact, and the progress of the project was often reviewed. However, due to the meeting schedule (Tuesday, Wednesday, Friday), the Wednesday meeting was often not very useful as most of the time little happened between it and the previous meeting. The team would likely have been better served with a Monday, Wednesday, Friday schedule.

It was also often difficult for the secretary to make detailed notes on what each member of the team had done since the last meeting, as the meeting stand-up usually included quite a lot of information in a short period of time. It might have been better for each member of the team to fill out their section before or during the meeting. Doing so would likely have avoided situations where it was later difficult to determine exactly what had been done in a given period of time.

The early meetings were also inefficient due to a lack of a standardized meeting template. This meant that occasionally important topics would be forgotten, and that the meeting minutes from the early meetings lacked detail and were thus hard to use later on. This could have been avoided by making the meeting template in the very first week.

The group started off with two missing group members. One was on exchange in Taiwan, and thus never showed up at all. The other had not received the email from NTNU with the start-up information for the subject, but the group was able to quickly get in touch with this member and they showed up for the group's second meeting.

Initially, only two roles were defined: project leader and secretary. It took the group some time to settle into clearer roles, which may have slowed down initial progress somewhat.

Since the team lacked domain knowledge, it had to spend a long time on research. All told, the first four weeks were dedicated to understanding the problem and the domain. During this progress, the team was quick to contact persons at NTNU with relevant knowledge. Otherwise this phase would have taken even longer.

Due to the complexity of the task, the team was unable to come up with a clear project plan, and instead had to focus on planning only a short time in advance. This meant that the team failed to plan how to test the solution early on, instead having to figure this out much later, and that no clear architecture existed until much later in the project.

This also meant that the life style model of the project was in a state of flux during the phase. Initially the team had hoped to use Scrum, but this proved infeasible. The team instead ended up with no distinct model, but rather a combination of Waterfall, Scrum, and Kanban elements.

The team quickly decided what type of solution to go with. It is possible it would have been better for the team to keep a more open mind to alternate solutions, though it is likely that even so the same approach would have been chosen in the end.

All told, even though so much time was dedicated to this phase, the team could not acquire a complete understanding of the domain, which often caused difficulties later. Making the phase any longer however would have compressed the other phases too much to be advisable.

12.1.2 Implementation

The implementation phase for the most part went well. The goal set in each preceding week was met each time with only one exception; while single-channel output was implemented in time for the week 40 customer meeting, it was not yet high enough quality to be presented.

Coding devolved to only three members of the group; the remaining three never directly touched the code, though they had indirect input. To further divide up the work was difficult due to how connected the tasks were. This mostly worked well, but the programmers could have done a better job of keeping the non-programmers updated by having more detailed status updates.

Testing was not considered until the latter parts of the implementation, which resulted in more time having to be spent on it than if it had been actively considered throughout the implementation.

It was also difficult to figure out how much progress was possible each week, mostly due to a lack of domain knowledge and previous experience with the frameworks used. Especially difficult was the Ocean framework, as the documentation was not especially good and the framework highly complex. Many hours were spent on learning the framework rather than doing directly productive work.

In the end, the implementation phase had to be extended one week further than initially planned, as various tasks took longer to complete than expected. The team was however satisfied with the final result.

12.1.3 Report

The team was able to start on the report early, making an outline showing what was needed. This made it easy to observe progress. This was combined with a Kanban board tracking all report related tasks. A downside of this however was that this structure thus was rarely changed; it may have been more rigid than would have been ideal. Overall however, the visibility of progress was very useful for team motivation and as feedback as to whether the team was progress fast enough.

While the outline of the report was created early, the actual content was not. This was especially the case for the early stages, which could likely be better covered if it had been written about earlier.

The first outline of the report was created on Google Drive, but the team soon after realized that it would be better to use L^AT_EX, and therefore moved to the ShareLaTeX platform

Having one member of the group in the role of report manager was very useful; this ensured that someone was always aware of how well the project was progressing. During most of the project, report task delegation worked well on a voluntary basis. Members of the team simply chose the tasks they were most interested in, which resulted in good progress once this measure was instituted. However, towards the end of the project this meant that all the least interesting tasks remained and only minor progress being made, resulting in the planned date for first draft status being missed. Pressure had to be applied, which resulted in most of what remained first draft status being reached later the same day, though still leaving a handful of sections still unfinished.

One difficulty was the size of tasks. Many report tasks started as monolithic tasks representing entire sections. Breaking them down to smaller, more manageable tasks was sometimes difficult, but always eventually achieved. This also meant that it was easy to end up with too many sub-sections, as each resulting task generally

resulted in a sub-section. This meant that towards the end many sub-sections had to be merged together to make the report more readable.

None of the team members were native English speakers. This was addressed by making sure all members of the team proofread the report. It thus ended up not causing any significant issues for the report.

All in all, report progress started out a bit slow, but went well once the team realized significantly more effort had to be expended on it.

12.2 Group dynamics

The group for the most part functioned very well, quickly establishing a group dynamic that helped rather than hindered the project.

One significant aspect that made the execution of the project smoother than it would otherwise have been, was solid procedures regarding attendance. Every time someone had to be absent from a meeting due to illness or other obligations, the rest of the team were notified well in advance. The number of times someone was late without having notified the team could be counted on one hand. Illness was rare, with only three members of the team ever calling in sick, and being unable to attend a meeting due to other obligations was equally rare. This was achieved in part by determining meeting times suitable for everyone at the very start of the project.

The team members did a good job of completing assigned tasks. Only rarely did someone show up to a meeting not having done the task(s) they had promised they would do. This work ethic helped make the especially complex project manageable.

There were effectively no conflicts within the team. Disagreements were all settled with discussions without ever reaching conflict level. As a result, the team all felt they were pulling towards the same goal, which helped keep the members motivated.

One downside of the friendly relations of the team was that it could at times result in distractions; occasionally the team would end up discussing entirely unrelated matters for a period of time, rather than working on the project. This was especially an issue during long team sessions, as it could be difficult to maintain focus for many hours in a row.

The roles of the different team members were also somewhat unclear, especially early on. At the beginning only a few roles were formally specified. This was eventually resolved by more formally defining every role. However, while most team members were content with their roles, there was a bit of friction. The original architect eventually determined they did not have the necessary architectural experience to do a great job in the role, and thus switched roles with the secretary. Beyond that, the role of project leader at times ended up effectively shared between the official

project leader and another member of the team due to that person's dominant nature. This did not cause any significant issues, but it did cause a bit of chaos from time to time.

Towards the end of the project, the group held a "roast", where each team member was given positive and negative critique by the rest of the team. This helped highlight what each person could do better. Only relatively minor issues ended up being brought up, but the exercise was still useful in alleviating those. This should have been done earlier in the project so that the issues could have been addressed earlier. Still, it made the last stages of the projects smoother, and gave the members of the team useful pointers for the future.

Overall, the group worked very well together with only a few occasional issues. Given the complex nature of the task, this cohesion was key to the success of the project.

12.3 Customer

The customer was of course central to the project. They generally did a good job of directing the team in a suitable direction and communicating what they were expecting from the project. The customer had a highly positive attitude to the project, which was very helpful in keeping the team motivated. They did a great job of listening and understanding when the team had issues making progress, and needed to dial down the overall project goals.

Some issues did still occur. Some of these were as far as the team could tell outside the customer's control, and rather caused by the subject staff or other constraints not easily controlled by the customer.

The biggest issue encountered was that the customer lacked information about a number of aspects of the course itself. Most significantly, the customer initially thought the team had specifically chosen their task rather than having been randomly assigned to it. This resulted in them not providing the task description at the start, believing the team already possessed it. This was not discovered until a few weeks into the project, when the team received an introduction to the Petrel software the team noticed that the person giving the introduction had received a task description from the customer. The task description contained some information the team was unaware of as a result of the customer believing they already possessed it. As far as the team was able to tell, this lack of information on the customer's part was a result of miscommunication between NTNU and the customer.

The highly specialized nature of the project, combined with the aforementioned missing task description, sometimes resulted in the team being unable to understand the customer's explanations. The team often simply lacked the domain knowledge to make sense of the more detailed explanations the customer gave, and the knowledge gap between the team and the customer made it at times difficult for the customer's representatives to explain concepts. This was reinforced by all meetings

except the very first one being over Skype for Business rather than in person, thus making conversation more difficult by removing body language from the equation. It was also at times difficult for the two parties to understand one another due to different native languages. As a result of these factors, the team ended up talking to professors, PhD students, and similar at NTNU a number of times in order to get explanations that were easier to understand, though most of the time the customer's explanations sufficed. This was mostly an issue early in the project during the pre-study and research phase.

The customer was generally very quick to respond to email inquiries. Only in a few instances did slowdowns in the project work result from customer response times, most notably the initial prototyping of the Petrel plug-in. The team could have done a better job of quickly emailing the customer when issues were encountered, and generally keeping the customer more up to date on the project's progress between customer meeting. While the customer's representatives were knowledgeable about most topics the team might want to ask about, they largely lacked any insight into machine learning, which at times caused communication issues as the team's understanding was different from their understanding of the constraints involved in using a neural network.

In total the team was largely satisfied with the customer's guidance and expectations, even though there was some room for improvement. After the pre-study and research phase the two parties largely managed to keep their goals and expectations well aligned.

12.4 NTNU

NTNU was responsible for the subject itself, and organized a variety of activities to assist the students taking the course. As the team rarely felt the need to reach out to NTNU itself during the project, they mainly interacted with NTNU through mandatory activities.

The team experienced a number of organizational issues on NTNU's part. First of all, the course compendium was not released until a few weeks into the course, forcing the team to rely instead on last year's compendium. The compendium, once released, had a few inaccuracies that had to be corrected by the course staff.

The subject's use of the available communication channels could have been better. A number of students never received the introduction email, and thus missed the intro lecture. This affected one member of the team, but the team managed to get in touch with the missing member the following day and sort out the issue. This issue could have been avoided by NTNU by linking the subject's website on itslearning and the official course page[39], as the vast majority of students check one or both before a subject begins. The team sees no issue with not using itslearning, but it is normal to ensure that the itslearning page links the communication channel that's used instead.

As mentioned in Section 12.3, it appears that NTNU failed to ensure the customer was aware of important details regarding how project groups were set up. The customer initially thought that the team had specifically asked for their project rather than having been randomly assigned to it. The team hopes that communication between NTNU and the customers will be better in the future.

This year, security was a focus for the subject, which the team applauds as security is clearly an important issue that warrants increased emphasis. However, most of the projects selected (including this project) had few to no security concerns, near invalidating the increased security emphasis. The team suggests NTNU makes sure to select projects for which security is a significant concern. Considering the computer science program at NTNU has no mandatory security classes, it would also be wise to distribute students in a way that ensures all groups had at least one member who had previously taken a software security course so that they could help the rest of the team learn about the topic. As this was not done the team had no members with any software security experience.

NTNU provided several lectures about a number of topics related to the subject. The contents of these lectures was generally high-quality. The ability of the lecturers to keep the students engaged varied, ranging from outstanding to mediocre. The group dynamics lecture had a group dynamics survey that was supposed to be provided three times so as to plot the changes in group dynamics over time, yet it was only provided once. As the actual results were not to be visible the first time, this meant the team never got to see and take into account the results of the survey. There was also no standard for how the lecture slides were provided. One was put on the course's page. At least one was emailed to only the project leader. Another was provided on paper. Ideally, these would all be uploaded to the course's website.

Another organizational issue the team experienced was that one of their initial seven members was on exchange to Taiwan and thus could not participate in the course. It should have been possible for NTNU to identify that the student would not attend and not assign them to a group.

Based on discussion with other groups, group leaders being pre-determined caused some friction. This team did not experience any significant issues caused by it, but this seems to have been largely a matter of luck.

Overall the subject was very much an interesting experience which the team learned a lot from. It still has room for improvement, especially on the organizational side. Many of the issues there could easily have been avoided. The team hopes NTNU will improve on this for next year's students.

12.5 Advisor

The team was appointed an advisor. According to the compendium, the mandate of the advisor is to "serve as a one-person 'steering committee' for your project.

His/her responsibility is to keep an eye on the main process of the work, and to oversee that sufficient contact with the customer is maintained”.

To accomplish this, the team had a meeting with the advisor each Friday. In these meetings the advisor would ask the status of the project and customer relations, but would say little beyond these few questions unless prompted. When the team had specific questions, the advisor’s responses were almost always useful, but what could be asked was limited significantly by the advisor’s understandable lack of knowledge of the domain of the project, namely petroleum and machine learning. The advisor meetings might have been of more use with an advisor who had some experience with these fields, though the team realizes that this might be difficult to arrange.

The main utility of the advisor for the team was a designated point of contact for questions the team had about the subject itself rather than the execution of the project, a role which they fulfilled admirably.

References

- [1] *About Skype for Business*. URL: <https://www.skype.com/en/business/skype-for-business/> (visited on 10/24/2016).
- [2] *About Slack*. URL: <https://slack.com/is> (visited on 10/24/2016).
- [3] *appear.in*. URL: <https://appear.in> (visited on 10/24/2016).
- [4] *Bash on Ubuntu on Windows*. URL: <https://msdn.microsoft.com/commandline/wsl/about> (visited on 11/07/2016).
- [5] *Daily Scrum Meeting*. URL: <https://www.mountangoatsoftware.com/agile/scrum/daily-scrum> (visited on 10/06/2016).
- [6] *Docker installation documentation*. URL: <https://docs.docker.com/engine/installation/windows/> (visited on 10/19/2016).
- [7] *Docker Toolbox installation file*. URL: <https://github.com/docker/toolbox/releases/download/v1.12.2/DockerToolbox-1.12.2.exe> (visited on 10/19/2016).
- [8] *Docker wikipedia entry*. URL: [https://en.wikipedia.org/wiki/Docker_\(software\)](https://en.wikipedia.org/wiki/Docker_(software)) (visited on 10/19/2016).
- [9] *Docker Windows 10 installation file*. URL: <https://download.docker.com/win/stable/InstallDocker.msi> (visited on 10/19/2016).
- [10] *Dropbox*. URL: <https://www.dropbox.com/> (visited on 11/03/2016).
- [11] *F3 Seismic data set*. URL: <https://opendtect.org/osr/pmwiki.php/Main/NetherlandsOffshoreF3BlockSeismicOnly494MB> (visited on 11/09/2016).
- [12] *Google Drive*. URL: <https://www.google.com/intl/en/drive/> (visited on 11/03/2016).
- [13] *ISO 29119*. 2013. URL: [http://ieeexplore.ieee.org/document/6588537/](http://ieeexplore.ieee.org/document/6588537) (visited on 09/15/2016).
- [14] Haukås Jarle et al. “Automated salt body extraction from seismic data using the level set method”. In: *First Break* 31 (2013), pp. 35–42. DOI: 10.3997/1365-2397.2013009.
- [15] *Kanban Explained*. URL: <https://help.rallydev.com/what-is-kanban> (visited on 10/07/2016).
- [16] *Key Elements of the Sprint Retrospective*. 2014-04-24. URL: <https://www.scrumalliance.org/community/articles/2014/april/key-elements-of-sprint-retrospective> (visited on 10/06/2016).
- [17] *Level Set Method Library (LSMLIB)*. URL: <http://ktchu.serendipityresearch.org/software/lsmlib/> (visited on 11/07/2016).
- [18] *Matplotlib*. URL: <http://matplotlib.org/> (visited on 11/03/2016).
- [19] *NumPy*. URL: <http://www.numpy.org/> (visited on 11/03/2016).
- [20] *PEP 8 Style Guide for Python*. URL: <https://www.python.org/dev/peps/pep-0008/> (visited on 11/01/2016).
- [21] *Petrel 2016*. URL: <https://www.software.slb.com/products/petrel/petrel-2016> (visited on 10/28/2016).
- [22] *Plant Phenotyping*. URL: <http://www.lemnatec.com/plant-phenotyping/> (visited on 10/17/2016).

- [23] Michael P. Pound et al. “Deep Machine Learning provides state-of-the-art performance in image-based plant phenotyping”. In: *bioRxiv preprint server* (2016). DOI: 10.1101/053033.
- [24] *PROTECTION POKER*. URL: <http://www.sintef.no/en/information-and-communication-technology-ict/sos-agile-blog/protection-poker/> (visited on 11/01/2016).
- [25] *PyCharm*. URL: <https://www.jetbrains.com/pycharm/> (visited on 11/03/2016).
- [26] *Python Imaging Library*. URL: <http://www.pythonware.com/products/pil/> (visited on 11/03/2016).
- [27] *Schlumberger Ocean*. URL: <https://www.ocean.slb.com/en> (visited on 11/04/2016).
- [28] *Schlumberger Petrel*. URL: <https://www.software.slb.com/products/petrel> (visited on 11/04/2016).
- [29] *SciPy*. URL: <https://www.scipy.org/> (visited on 11/03/2016).
- [30] *ScrumMaster*. URL: <https://www.mountangoatsoftware.com/agile/scrum/scrummaster> (visited on 10/06/2016).
- [31] *SDLC Waterfall Model*. URL: https://www.tutorialspoint.com/sdlc/sdlc_waterfall_model.htm (visited on 10/23/2016).
- [32] *SEG Advanced Modeling Program*. URL: <http://seg.org/News-Resources/Research-Programs/SEAM> (visited on 11/07/2016).
- [33] *SegPy*. URL: <https://github.com/sixty-north/segpy> (visited on 11/03/2016).
- [34] *ShareLaTeX*. URL: <https://www.sharelatex.com/> (visited on 11/03/2016).
- [35] *Society Of Exploration Geophysicists*. URL: <http://seg.org/> (visited on 11/09/2016).
- [36] *Software Development Methodologies*. URL: <http://www.itinfo.am/eng/software-development-methodologies/> (visited on 11/10/2016).
- [37] *Software Requirements*. URL: https://www.tutorialspoint.com/software_engineering/software_requirements.htm (visited on 10/24/2016).
- [38] Anne H. S. Solberg, Angélique Berthelot, and Leiv-J Gelius. “Texture attributes for detection of salt”. In: *Journal of Applied Geophysics* 88 (2012), pp. 52–69. DOI: 10.1016/j.jappgeo.2012.09.006.
- [39] *TDT4290 course page*. URL: <https://www.idi.ntnu.no/emner/tdt4290/> (visited on 11/05/2016).
- [40] *The pain and gain in market shares for oilfield service companies!* URL: <http://www.oilvoice.com/n/The-pain-and-gain-in-market-shares-for-oilfield-service-companies/c92910916ffe.aspx> (visited on 09/27/2016).
- [41] *The Scrum Guide*. 2014. URL: <https://www.scrumalliance.org/why-scrum/scrum-guide> (visited on 10/06/2016).
- [42] *The Scrum Product Backlog*. URL: http://www.scrum-institute.org/The_Scrum_Product_Backlog.php (visited on 10/06/2016).
- [43] *Visual Studio Installation Guide*. URL: <https://msdn.microsoft.com/en-us/library/e2h7fzkw.aspx2> (visited on 11/04/2016).
- [44] Julia Wester. *What is Kanban*. URL: <http://www.everydaykanban.com/what-is-kanban/> (visited on 10/07/2016).

- [45] *What Characteristics Make Good Agile Acceptance Criteria?* 2015. URL: <http://www.seguetech.com/what-characteristics-make-good-agile-acceptance-criteria/> (visited on 11/02/2016).
- [46] *What is a Salt Dome.* URL: <http://geology.com/stories/13/salt-domes/> (visited on 11/10/2016).
- [47] *What is agile? What is scrum.* URL: <https://www.cprime.com/resources/what-is-agile-what-is-scrum/> (visited on 11/11/2016).
- [48] *What is Docker?* URL: <https://www.docker.com/what-docker> (visited on 10/19/2016).
- [49] *Windows 10 Anniversary Update SDK Now Available! Windows Store Open for Submissions.* URL: <https://blogs.windows.com/buildingapps/2016/08/02/windows-10-anniversary-update-sdk-14393/#Q267Kd5MWB3rv0j.97> (visited on 11/07/2016).

Glossary

Petroleum

Auto tracking: One way to delineate salt and other geological formations is "auto tracking". This is a simplistic method that tries to automatically follow a pattern in a seismic cube. The results are generally correct, but will usually stop if there's any gap in the pattern, significantly limiting the utility for geological structures that don't clearly stand out, as is often the case for salt body. See Section 3.3 for more information.

Crossline: See *Cross-section*.

Cross-Section: A vertical 2D-section of seismic data from a seismic cube. Can be oriented in either *Inline* or *Crossline* direction. The inline-direction is orthogonal on the crossline-direction, and both directions are parallel to one of the boundaries of the seismic cube.

Delineate: To indicate the exact position of (a border or boundary). Specifically: to indicate the border of a salt body.

Inline: See *Cross-Section*.

Interpretation: A seismic cube has to be interpreted in order for it to be useful. Interpretation is the process of using automated tools and manual work to indicate where objects of interests lie within the cube. In the case of projects, this means indicating the edges of salt bodies

Intersection: One specific cross-section. See *Cross-section*.

Multi-Z: Multi-Z can refer to two different concept. Firstly, it can refer to a subsurface structure (e.g., a salt body) that intersects the z-axis multiple times; that is, under a single surface point the boundary of the object will be encountered several times. Secondly, it can refer to a Petrel tool for delineating 3D bodies that exhibit the aforementioned behavior

Ocean A software development framework used to create a plug-in for Petrel. For more information, see 4.3.

Petrel: A software platform that can be used to solve petroleum-related subsurface challenges. For more information, see Section 4.2.

Salt body: Geological processes can result in large concentrations of near pure salt. These are known as "salt bodies". They have a tendency to have complex three-dimensional structures, making their delineation difficult.

Seismic: "Seismic" refers to the propagation of sound waves through sub-surface structures like rock and salt, and the results utilizing these concepts can give via *seismic imaging*

Seismic cube: A collection of trace can be combined to make a three-dimensional structure known as a "seismic cube". By loading this into a seismic imaging program (e.g., Petrel), users can see the detected geological boundaries in three dimensions. See Section 3.1 for more information.

Seismic horizon: A transition between sediments. Can be seen in seismic imaging because the transition between the sediments cause reflections which are detected. See Section 3.1 for more information.

Seismic imaging: In seismic imaging, sound waves are generated that propagate down below the surface of the Earth, and are partially reflected as a result of the changing properties of the structures there. These reflections can be used to detect the structure of objects below ground. See Section 3.1 for more information.

Trace: When utilizing seismic imaging, each pressure wave will produce a so-called "trace" showing the strength of the reflected pressure wave over time. See Section 3.1 for more information.

Computer Science

Convolution: "Convolution" is a mathematical operation for combining two separate functions. It is heavily used in a variety of areas including image processing, as it allows the effective combination of different functions. It is used in machine learning by convolutional neural network, which are especially good at image recognition. See Section 3.4 for more information.

Neural network: An artificial neural network is a network inspired by their biological counterpart, the brain. They can be used to approximate an unknown function by gradually modifying the network (known as neural network training) to replicate known data as closely as possible. See Section 3.4 for more information.

Neural network training: Neural network have to be trained on known data. This consists of sending data with a known "correct" output into the network, then if the network output is wrong, modifying the network in some way to more closely approximate the correct output. See Section 3.4 for more information.

Appendices

A Meeting Minutes Templates

Template for Tuesdays: Internal meetings

Start:

End:

Attendance:

Scrum stand-up:

What have you done since last meeting?

Camilla:

Ida:

Magne:

Martin:

Thomas:

Tobias:

Are there any impediments in your way?

Camilla:

Ida:

Magne:

Martin:

Thomas:

Tobias:

Report

Status

Delegation

Code

Status

Delegation

Until next time (Wednesday)

What are you going to do until next meeting?

Camilla:

Ida:

Magne:

Martin:

Thomas:

Tobias:

Optionally

Questions to Schlumberger

Template for Wednesdays: Customer and Internal Meetings

Start:

End:

Attendance:

Meeting with Schlumberger

Tell what we have done

Skype meeting

Discussing Schlumberger meeting

Goals for next time:

Scrum stand-up:

What have you done since last meeting?

Camilla:

Ida:

Magne:

Martin:

Thomas:

Tobias:

Are there any impediments in your way?

Camilla:

Ida:

Magne:

Martin:

Thomas:

Tobias:

Until next time (Friday)

What are you going to do until next meeting?

Camilla:

Ida:

Magne:

Martin:

Thomas:

Tobias:

Optionally

Questions to Schlumberger

Template for Fridays: Supervisor and Internal Meetings

Start:

End:

Attendance:

Meeting With Supervisor

Tell what we have done

During the meeting

Scrum stand-up:

What have you done since last meeting?

Camilla:

Ida:

Magne:

Martin:

Thomas:

Tobias:

Are there any impediments in your way?

Camilla:

Ida:

Magne:

Martin:

Thomas:

Tobias:

Report

Status

Delegation

Code

Status

Delegation

Until next time (Tuesday)

What are you going to do until next meeting?

Camilla:

Ida:

Magne:

Martin:

Thomas:

Tobias:

Optionally

Questions to Schlumberger

B Installation Guide

This guide will cover the installation of everything necessary for the execution of the solution other than the prerequisites.

B.1 Prerequisites

Some pieces of software are assumed to already be present, namely:

- 64-bit Windows 7 or higher
- Petrel
- Ocean
- Visual Studio

B.2 Python program

TensorFlow is built with Google's own build tool, Bazel, which currently lacks Windows support. This means that it is not possible to install TensorFlow on Windows, and it is necessary to use some kind of virtual machine if you want to run TensorFlow on a Windows machine. Quote of description of Docker from Wikipedia:

"Docker is an open-source project that automates the deployment of Linux applications inside software containers." [8]

Read more about Docker on it's website. [48]

B.2.1 Step 1 - Install Docker

For Windows 10 Professional or Enterprise 64-bit, the installation file can be found at [9]. For other Windows versions, Docker Toolbox must be used instead. Installation file at [7]. See Docker installation documentation [6] if any issues arise.

B.2.2 Step 2 - Build Docker image

Open the Docker Quickstart Terminal. This will launch a virtual machine server. Once up and running, it should look like figure B.1.

Next, navigate the terminal to the folder containing the solution and run the command (**NOTE:** Include the dot at the end in the command. Also: "img" is an arbitrary name, any lowercase word without space is fine)



Figure B.1: Docker Quickstart Terminal

```
docker build -t img .
```

Note that the image name needs to be lowercase. A successful build looks something like figure B.2.

B.3 Petrel plug-in

Prerequisites for installing the plug-in are Petrel, Ocean and Microsoft Visual Studio. For installation of Petrel, see [28]. For installation of Ocean, see [27], and for an installation guide for Visual Studio, see [43].

Installation

To install the plug-in do the following steps.

1. In the folder "Visual Studio" inside the Solution-folder, open the Visual Studio-project. (Double-click the file called "Testinterpretation.sln")
2. Find Visual Studio's *Solution Explorer* to find the files in the project.
3. Inside the file called "SaltMenuHandler.cs" find the class "RunNeuralNetwork-CommandHandler". Find the method called "execute". Change the variable called "filePath" to the folder you prefer to use. E.g. if you want to use the folder "Desktop" it should look like this:
var filePath = @"C:/Users/*yourUserName*/Desktop";
4. Build the solution. *Build Solution* can be found in the *Build*-menu in Visual Studio. **Notice that Petrel cannot be open while the solution is built.**

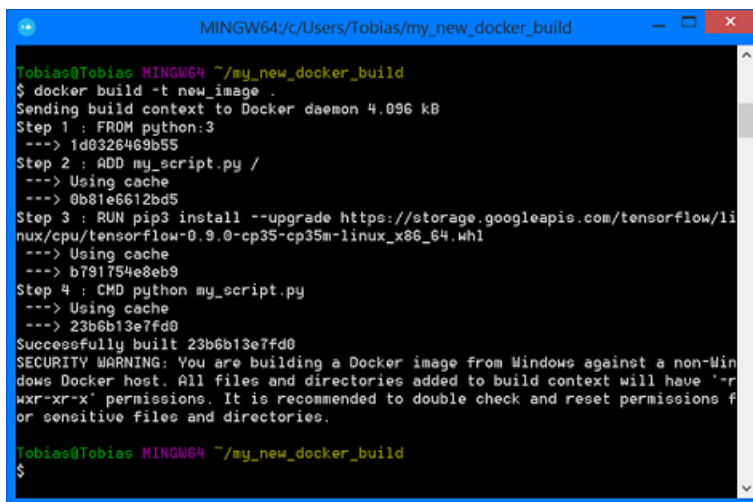
A screenshot of a Windows terminal window titled "MINGW64/c/Users/Tobias/my_new_docker_build". The terminal shows the execution of the 'docker build' command. The output indicates a successful build of a Docker image. The steps shown are: Step 1: FROM python:3; Step 2: ADD my_script.py /; Step 3: RUN pip3 install --upgrade https://storage.googleapis.com/tensorflow/linux/cpu/tensorflow-0.9.0-cp35-cp35m-linux_x86_64.whl; Step 4: CMD python my_script.py. The build is successfully completed with the image ID 23b6b13e7fd0. A security warning is displayed: "SECURITY WARNING: You are building a Docker image from Windows against a non-Windows Docker host. All files and directories added to build context will have 'rwxr-xr-x' permissions. It is recommended to double check and reset permissions for sensitive files and directories." The terminal ends with the prompt '\$'.

Figure B.2: Successful Docker image build

The plug-in is now built and can be found inside Petrel. For usage, see Section C.2.

C User manual

The Python program and Petrel plug-in are effectively run separately from one another.

C.1 Python program

This section will describe how to run the neural network. The program comes with a hard-coded default input file. In order to change this and run it on another input file, see the "Optional configuration" section below.

With Docker set up, it is possible to execute the script with the TensorFlow code from the Docker Quickstart Terminal. In the command below, replace "PATH_TO_FOLDER" with the actual absolute path to the solution folder, with forward slashes used rather than backslashes, while "c" represents the drive number.

```
docker run -it -v /c/PATH_TO_FOLDER/output/:/output img
```

For example, if the solution is located at "D:/Users/Name/Documents/Solution/", one would have to run:

```
docker run -it -v /d/Users/Name/Documents/Solution/output/:/output img
```

This will run a pre-trained network on an unseen seismic cross-section. This will generally take 5 to 15 minutes depending on the system, but should output progress throughout. The output represents the performance of the network on every 20th line of the cross-section. There are roughly 380 lines to process.

Output

The program will produce two images; one showing the salt delineation determined by the network, and the other will show the same after post-processing has been done to reduce false positives. For each of the two images, a text file containing the points identified as being part of the salt edge is also produced. These text files can be loaded into Petrel using the Petrel plug-in. These outputs will appear in the "output" folder of the solution folder.

Optional configuration

NOTE: After doing any changes to the code, the image needs rebuilding, with the

```
docker build -t img .
```

command.

In order to change the input, the code has to be modified. The slice to be evaluated is set in the "evaluate" function in network.py. It defaults to "Inline604". It can be switched to any other name, but note that in order for it to work there must be a seismic cross-section in training/TextCrosslines/Textfiles with the specified name.

The training data (not used unless the program is reconfigured to train a network rather than use a pre-trained one) is specified in the "load_images" function in image.py. They can be changed in the same manner as the evaluation file, though for them there also has to be a PNG file with the same dimensions as the seismic cross-section in training/TextCrosslines/Delineated named IDENTIFIER_delineated.png, where "IDENTIFIER" is the name specified in the code. This PNG should be white except along the edge of the salt, which should be black. It has to be a greyscale image. In the "load_images", one of the add_image lines has to be copied, and its name replaced with that of the new data.

Another configuration option is whether to use a saved network or creating a new one. By default a pre-trained one is used, but this can be toggled changing the variable "self.load = True" to False in network.py. If so, the network will load the last version saved. The network will by default be saved whenever it is done training, but note that this will overwrite whatever has already been saved unless

the previous save is renamed or moved. The solution also comes with a pre-trained network; the training/models folder contains a folder named "20000", containing the weights for a network trained 20 000 steps. This is also the network contained in the "training/models" folder itself, which is where saved networks are loaded from and saved to. Note that in order for training to happen, the pound-sign in the line "#self.train(20000)" has to be removed in network.py. Training can depending on the machine the program is run on take anything from half an hour to several hours. A smaller number than 20000 can be set in the aforementioned line to reduce the number of training epochs, making training faster at the cost of accuracy.

C.2 Petrel plug-in

How to use Petrel is expected to be known by the user. If the user needs more information about this, a good resource is the *Petrel Help Center* inside Petrel. Where to find this can be seen in Figure C.3

Open Petrel

To open Petrel do the following steps. A Petrel project called *SeamPetrelProject* can be found in the folder called "Petrel Project".

1. Open Petrel
2. Select *Ocean Framework* under *Core Licenses* and click *OK*.
3. Open a project and import a seismic cube. An example-project called *Seam-PetrelProject* can be found in the Solution-folder. This can be opened via the *File*-menu in Petrel. Select *Open Project...*
4. If the cube is shown with no seismic as in Figure C.1 you will have to reconnect with the seismic data. This can be done by doing the following steps:
 - Right-click the seismic cube. (Highlighted in Figure C.2). Select settings.
 - Go to the info-pane of the settings-dialogue
 - At the bottom a field called *Orig. filename* can be found. Press the button to the right of this field.
 - Browse to Solution/Petrel Project/Petrel Data and select the file called *SEAM.Interpretation_Challenge_1_Time*. Click Open to get back to the Settings-dialogue.
 - Click *OK*.

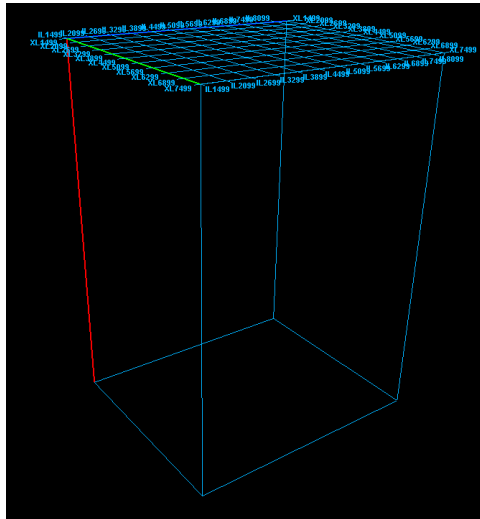


Figure C.1: A seismic cube with no data.

C.2.1 Run the Neural Network for the Example-Project

To run the Neural Network for the Example-Project do the following steps:

1. Find the folder called "Example Results" in the solution folder, and copy its contents into the folder you specified in the "filePath"-variable.
2. In the *Input*-pane of Petrel, right-click the seismic cube (the cube in the example-project is called *SEAM_Interpretation_Challenge_1_Time*. See Figure C.2). A context-menu similar to the one in Figure 10.8 should appear. In the menu, find *Salt Delineation* and select *Run Neural Network*.
3. A new folder in the *Input*-pane will appear. The folder should be called *Neural Network Result*. To view the results, click the check-box at the left side of this folder.

Tweak the Settings

All tweaking of settings has to be done in code. Remember to close Petrel, because the plug-in has to be re-built, and that requires Petrel to be closed. To tweak any setting do the following steps (See Section B.3).

1. Open the Solution in Visual Studio.
2. Tweak a setting according to one of the possibilities presented in the upcoming sections.

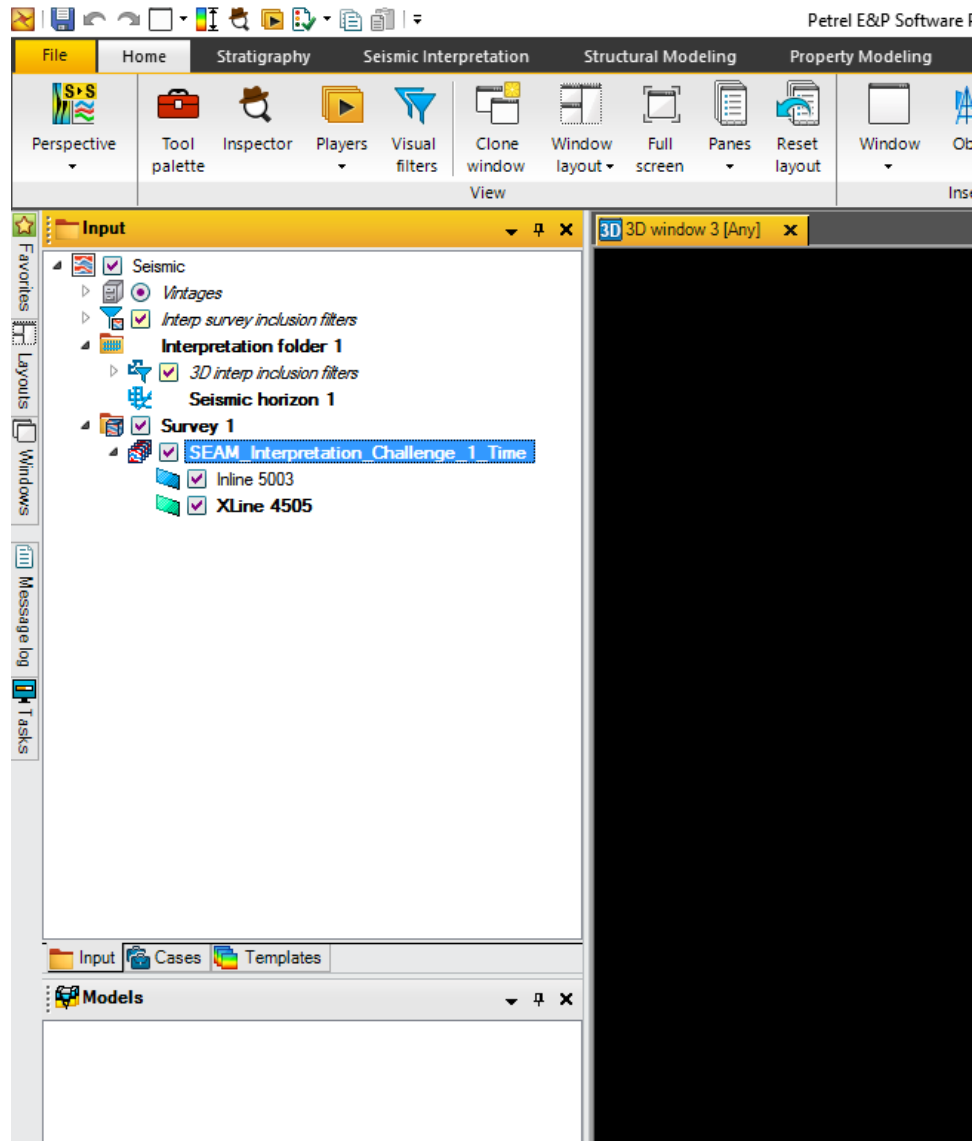


Figure C.2: The seismic cube is inside "Survey 1". Where to right-click is highlighted in blue.

3. Re-build the plug-in.
4. Open Petrel.

Run the Network for All Cross-Sections in Crossline-direction

If you want to export and import all Cross-sections in the Crossline-direction of the seismic cube do the following steps:

1. Open the Solution in Visual Studio
2. Inside the file called "SaltMenuHandler.cs" find the class "RunNeuralNetworkCommandHandler". Find the method called "execute".
3. Comment (that is add "//" in front of) the line with the following contents: "exportAndReadSpecificCrossLines(cube, filePath);".
4. Un-comment (that is remove "//" in front of) the line with the following contents: "runNeuralNetworkCrossLines(cube, filePath, 1);"
5. *Optional:* if you want the system to skip some cross-sections between each cross-section exported/imported, change the 1 at the end of the line to e.g. 10. This would make the system output every 10th cross-section instead of each cross-section.
6. Re-build the plug-in via the Build-menu in Visual Studio.

Open Petrel and run the plug-in like usual.

Run the Network for All Cross-sections in Inline-direction

If you want to export and import all Cross-sections in the Inline-direction of the seismic cube do the following steps:

1. Open the Solution in Visual Studio
2. Inside the file called "SaltMenuHandler.cs" find the class "RunNeuralNetworkCommandHandler". Find the method called "execute".
3. Comment (that is add "//" in front of) the line with the following contents: "exportAndReadSpecificCrossLines(cube, filePath);".
4. Un-comment (that is remove "//" in front of) the line with the following contents: "runNeuralNetworkInlines(cube, filePath, 1);"
5. *Optional:* if you want the system to skip some cross-sections between each cross-section exported/imported, change the 1 at the end of the line to e.g. 10. This would make the system output every 10th cross-section instead of each cross-section.
6. Re-build the plug-in via the Build-menu in Visual Studio. Remember that Petrel has to be closed when the plug-in is built.

Open Petrel and run the plug-in like usual.

Run the Neural Network for Specific Cross-sections

If you want to export and import some specific Cross-sections in Inline-direction, Crossline-direction or both, do the following changes:

1. Open the Solution in Visual Studio
2. Inside the file called "SaltMenuHandler.cs" find the class "RunNeuralNetworkCommandHandler". Find the method called "execute".
3. Un-comment (that is add "//" in front of) the line with the following contents: "exportAndReadSpecificCrossLines(cube, filePath);".
4. Comment (that is remove "//" in front of) the line with the following contents: "runNeuralNetworkCrossLines(cube, filePath, 1);".
5. Comment the line with the following contents: "runNeuralNetworkInlines(cube, filePath, 1);".
6. Find the method called "exportAndReadSpecificCrossLines". Inside this method do the following changes:
 - Edit the list called "indexIList" to contain the specific inlines you want to import/export. For example if you want the system to import/export crosslines 254 and 305 the numbers inside the "{ }" would look as follows: "{ 254, 305 }".
 - Edit the list called "indexJList" to contain the specific crosslines you want to import/export. For example if you want the system to import/export crosslines 128 and 269 the numbers inside the "{ }" would look as follows: "{ 128, 269 }".
7. Re-build the plug-in via the Build-menu in Visual Studio. Remember that Petrel has to be closed when the plug-in is built.

Open Petrel and run the plug-in like usual.

C.2.2 Run the Network for Other Projects

To create delineations, the following steps has to be performed:

1. Select which cross-sections you want to create a delineation for through the steps mentioned above.
2. Rebuild the plug-in via Visual Studio. Keep in mind that Petrel cannot be open while the plug-in is rebuilt.
3. Run the plug-in via the "Run Neural Network"-menu item.
4. In the folder selected in "filePath", the selected cross-sections will appear as files named by the following convention: "Inline{number}" or "Xline{number}".
5. For each cross-section you want to create a delineation do the following:

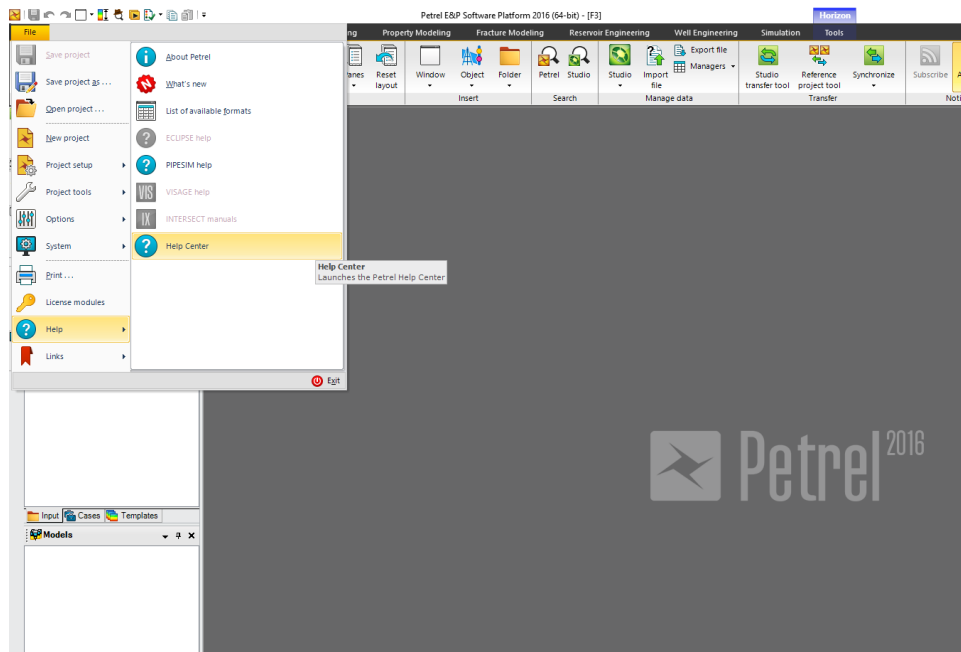


Figure C.3: The *Petrel Help Center* can be found in the *Help*-menu in Petrel.

- Copy the cross-section file into the correct folder for the neural network. (See Section C.1).
 - Run the neural network for this cross-section (See Section C.1).
 - If you want to use the post-processed output:
Rename the output from the neural network from "coords-processed.txt" to "coordsnumber.txt". For example if you ran the network on Inline504, this should be renamed to "coords504.txt".
 - If you want to use the non-post-processed output:
Rename the output from the neural network from "coords.txt" to "coordsnumber.txt". For example if you ran the network on crossline555, this should be renamed to "coords555.txt".
 - Copy this output to the folder specified in the "filePath"-variable.
6. In Petrel select "run neural network" again. The results should now appear in the folder called "Neural Network result".
 7. To see the results, click the checkbox to the left of the folder called "Neural network result".

D Test Case Specifications

ID	U01
Objective	Checking if edge-pixels are correctly identified
Preconditions	Not needed
Inputs	3x3 matrix consisting of only edges except center, 4 sets of coordinates along the edge
Expected results	true, true, true, true
Actual results	true, true, true, true
Test results	Passed

Table D.1: Test case U01

ID	U02
Objective	Checking if coordinates not on the edge are correctly identified
Preconditions	Not needed
Inputs	3x3 matrix consisting of only edges except center, 5 sets of coordinates not on the edge
Expected results	false, false, false, false, false
Actual results	false, false, false, false, false
Test results	Passed

Table D.2: Test case U02

ID	U03
Objective	Checking if pixel is correctly identified as bordering on an edge
Preconditions	Not needed
Inputs	3x3 matrix consisting of only edges except center, Coordinates for the middle pixel
Expected results	true
Actual results	true
Test results	Passed

Table D.3: Test case U03

ID	U04
Objective	Checking if pixel is correctly identified as not bordering on an edge
Preconditions	Not needed
Inputs	3x3 matrix consisting of no edges, Coordinates for the middle pixel
Expected results	false
Actual results	false
Test results	Passed

Table D.4: Test case U04

ID	U05
Objective	Checking if double threshold method preserves points when close to probable salt
Preconditions	Not needed
Inputs	3x3 matrix consisting of one clear edge point, and several quite probable points located in a line
Expected results	Points on the line is preserved
Actual results	Points on the line is preserved
Test results	Passed

Table D.5: Test case U05

ID	U06
Objective	Checking if double threshold method discards points when not close to probable salt
Preconditions	Not needed
Inputs	3x3 matrix consisting of one clear edge point, and several quite probable points located in a line, with a few unlikely points in between
Expected results	Points are not preserved after the unlikely points
Actual results	Points are not preserved after the unlikely points
Test results	Passed

Table D.6: Test case U06

ID	U07
Objective	Checking if double threshold method discards points when no clear salt is present
Preconditions	Not needed
Inputs	3x3 matrix consisting of no clear edge point, and several probable points located in a line
Expected results	Points are not preserved
Actual results	Points are not preserved
Test results	Passed

Table D.7: Test case U07

ID	C01
Objective	Quantitative test of the accuracy of the delineation, using the validation set
Preconditions	A trained neural network
Inputs	Validation Cross-section
Expected results	95%
Actual results	97.66%
Test results	Passed

Table D.8: Component test 01

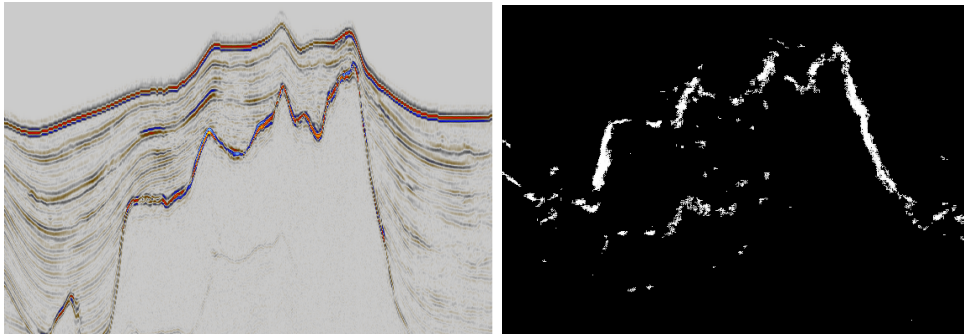


Figure D.1: Test cross-section from SEAM delineated

ID	C02
Objective	Qualitative test of the accuracy of the delineation, using an unused (test) cross-section
Preconditions	A trained neural network
Inputs	Test cross-section
Expected results	Most of the salt delineated
Actual results	See Figure D.1
Test results	Passed under doubt

Table D.9: Component test 02

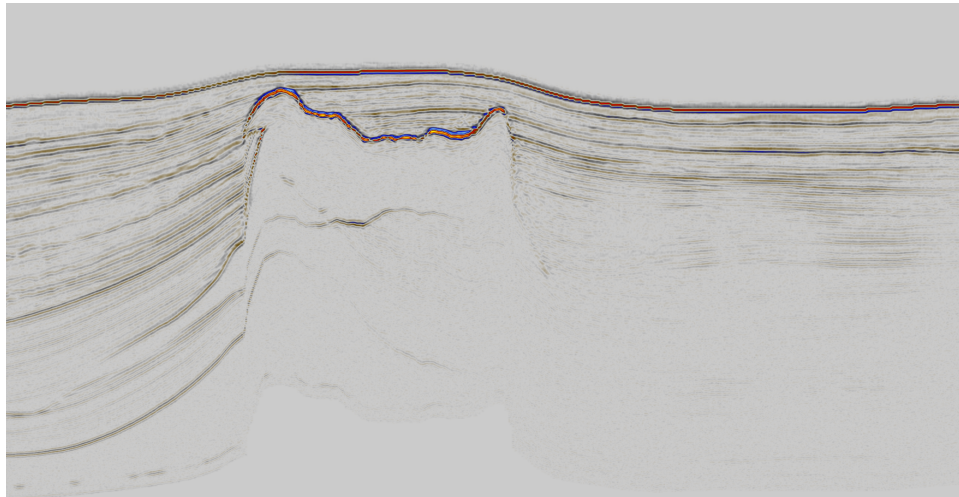


Figure D.2: Simple cross-section from SEAM Interpretation Challenge Time

ID	S01
Objective	Checking the time spent delineating one simple cross-section of a seismic cube see Figure D.2
Preconditions	A trained neural network
Inputs	Seismic cube
Expected results	18-23 minutes (75 % of the estimated 25-30 minutes)
Actual results	513.96 seconds
Test results	Passed

Table D.10: System test 01

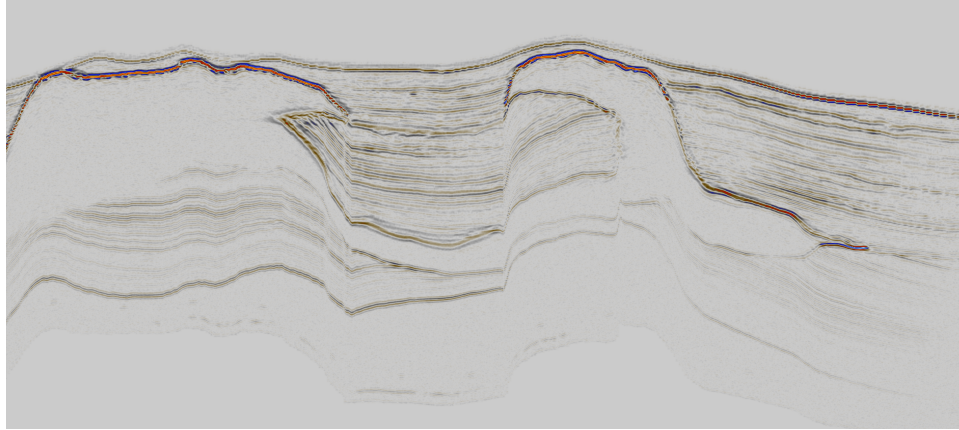


Figure D.3: Complex cross-section from SEAM Interpretation Challenge Time

ID	S02
Objective	Checking the time spent delineating one complex cross-section of a seismic cube see Figure D.3
Preconditions	A trained neural network
Inputs	Seismic cube
Expected results	30 minutes (75 % of the estimated 40 minutes)
Actual results	549.43 seconds
Test results	Passed

Table D.11: System test 02