NATIONAL TECHNICAL UNIVERSITY OF ATHENS

DATA SCIENCE AND MACHINE LEARNING

Few-shot learning for semantic segmentation on seismic data

Theodoros Papadopoulos

supervised by
Dr. Konstantinos KARANTZALOS

April 8, 2021

Εθνικό Μετσόβιο Πολυτεχνείο

Επιστήμη Δεδομένων και Μηχανική Μάθηση

Μάθηση με λίγα δεδομένα για σημασιολογική κατάτμηση σε σεισμικά δεδομένα

Θεόδωρος Παπαδόπουλος

επιβλέπων καθηγητής
Δρ. Κωνσταντίνος Καράντζαλος

April 8, 2021

Abstract

During the last ten years, in alignment with the resurgence of interest in machine learning algorithms driven mainly by the revolutionary discoveries of deep learning, the field of hydrocarbon exploration has received a huge boost in efficiency, productivity and time saving. The task of seismic interpretation, the process of identifying objects of interest in the earth's subsurface, can be effectively studied under the semantic segmentation domain, a field that so far presents very promising results on the topic.

However, contrary to the more traditional semantic segmentation tasks, seismic data present a major challenge with the amount of quality labeled data available which is a crucial component of most current state of the art supervised learning algorithms and this shortage originates primarily from two factors. The first one is the fact that proper seismic data labeling can be performed almost exclusively by geologists, experts in their field which makes the task slower since it can not be easily outsourced to non experts and at the same time, the process also becomes more expensive. The second major factor is that due to the earth's heterogeneity, the labels created from one location are not always transferable to different locations even when referring to similar objects of interest. This represents a significant additional challenge in terms of available data and corresponding labels, compared to applications for common everyday objects.

The purpose of this thesis is to acknowledge and embrace this data shortage, and try to pivot the answer to this problem from the industry standard supervised learning solution to a new promising sub-field called few-shot learning. To tackle this problem we build upon an existing few-shot learning approach, which we adapt to the specific requirements of a multi-class subsurface segmentation problem. Our proposed solution includes converting the network logic from binary classification to multi-class, addressing the image patching locality problem by injecting a global view of the labels to the network both during training and inference, we also use pre trained weights to tackle the data scarcity and finally, provide a general computationally efficient post processing approach that can be used with other existing seismic segmentation methods. The proposed method is also capable of reducing texture bias, which seems to play an important role in seismic data.

To the best of our knowledge this is the first attempt to solve the problem using this technique and our results can reach state of the art accuracy compared to current solutions used in the field, using only as few as 10 training examples.

Περίληψη

Κατά τη διάρκεια των τελευταίων δέκα ετών, σε συνδυασμό με την αναβίωση του ενδιαφέροντος για αλγόριθμους μηχανικής μάθησης που βασίζονται κυρίως στις επαναστατικές ανακαλύψεις των νευρωνικών δικτύων βαθιάς μάθησης, το πεδίο της εξερεύνησης υδρογονανθράκων έχει λάβει τεράστια ώθηση ως προς την αποδοτικότητα, την παραγωγικότητα και την εξοικονόμηση χρόνου. Το έργο της ερμηνείας σεισμικών δεδομένων, η διαδικασία δηλαδή εντοπισμού αντικειμένων ενδιαφέροντος κάτω απο την επιφάνεια της γης, μπορεί να μελετηθεί αποτελεσματικά απο τον τομέα της σημασιολογικής κατάτμησης (semantic segmentation), ένα πεδίο που μέχρι στιγμής παρουσιάζει πολύ ελπιδοφόρα αποτελέσματα.

Ωστόσο, σε αντίθεση με τις πιο παραδοσιακές τεχνικές σημασιολογικής κατάτμησης, τα σεισμικά δεδομένα παρουσιάζουν ένα σημαντικό πρόβλημα με την ποσότητα των διαθέσιμων ποιοτικών δεδομένων, τα οποία είναι ένα κρίσιμο συστατικό των πιο πρόσφατων αλγορίθμων μάθησης με επίβλεψη (supervised learning) και η έλλειψη αυτή προέρχεται κυρίως από δύο παράγοντες. Ο πρώτος είναι το γεγονός ότι η σωστή σεισμική επισήμανση δεδομένων μπορεί να πραγματοποιηθεί σχεδόν αποκλειστικά από γεωλόγους, ειδικούς στον τομέα τους, γεγονός που καθιστά την διεργασία πιο αργή, καθώς δεν μπορεί να ανατεθεί εύκολα σε μη ειδικούς και ως εκ τούτου, ταυτόχρονα η διαδικασία γίνεται πολύ πιο ακριβή. Ο δεύτερος επίσης σημαντικός παράγοντας είναι ότι λόγω της ετερογένειας της γης, οι ετικέτες που δημιουργούνται από μία τοποθεσία δεν μεταφέρονται πάντα με ακρίβεια σε διαφορετικές τοποθεσίες, ακόμη και όταν αναφέρονται σε παρόμοια αντικείμενα ενδιαφέροντος. Αυτό αντιπροσωπεύει μια πρόσθετη πρόκληση όσον αφορά τα διαθέσιμα δεδομένα και τις αντίστοιχες ετικέτες, σε σύγκριση με τις εφαρμογές για κοινά καθημερινά αντικείμενα.

Ο σκοπός αυτής της διατριβής είναι να αναγνωρίσει αυτήν την έλλειψη δεδομένων, και να προσπαθήσει να μετατοπίσει την απάντηση σε αυτό το πρόβλημα από τη κλασσική λύση μάθησης με επίβλεψη σε ένα νέο πολλά υποσχόμενο υποτομέα της που ονομάζεται εκπαίδευση με λίγα δεδομένα (few-shot learning). Για να αντιμετωπίσουμε αυτό το πρόβλημα στηριζόμαστε σε μια υπάρχουσα υλοποίηση μάθησης με λίγα δεδομένα, την οποία προσαρμόζουμε στις δικές μας απαιτήσεις. Η προτεινόμενη λύση μας περιλαμβάνει τη μετατροπή της λογικής δικτύου από δυαδική ταξινόμηση (binary classification) σε ταξινόμηση πολλών κατηγοριών (multi-class classification), αντιμετώπιση των προβλημάτων που δημιουργούνται με την κατάτμιση των εικόνων μεσω της εισαγωγής ολόκληρων των ετικετών στο δίκτυο κατά τη διάρκεια της εκπαίδευσης και της παραγωγής προβλέψεων, χρησιμοποιούμε επίσης προ-εκπαιδευμένα βάρη για την αντιμετώπιση της έλλειψης δεδομένων και τέλος, παρέχουμε μια γενική και υπολογιστικά αποδοτική προσέγγιση για την μετέπειτα επεξεργασία δεδομένων (post processing) που μπορεί να χρησιμοποιηθεί και με άλλες υπάρχουσες σεισμικές μεθόδους κατάτμησης. Η προτεινόμενη μέθοδος επιπλέον επιτρέπει την ελαχιστοποίηση της μεροληψίας της ¨υφής" των δεδομένων (texture bias) το οποίο δείχνει να είναι σημαντικός παράγοντας στα σεισμικά δεδομένα.

Από όσο γνωρίζουμε αυτή είναι η πρώτη προσπάθεια επίλυσης αυτου του προβλήματος με τη χρήση αυτής της τεχνικής και τα αποτελέσματά μας μπορούν να παράξουν μεγάλη ακρίβεια σε σύγκριση με τις τρέχουσες λύσεις που χρησιμοποιούνται στον τομέα, χρησιμοποιώντας μόνο 10 παραδείγματα εκπαίδευσης.

Acknowledgments

I would like to give my thanks and appreciation to the supervising professor of this dissertation, Dr. Konstantinos Karantzalos for the guidance he provided me and the excellent communication we had during the implementation of this thesis.

In addition, I would like to thank my colleague and good friend Dr. Lukas Mosser for the countless inspiring conversations we had all these past months, and also for the tips and observations he provided that proved to be of great help.

Finally, I would like to thank my family for the endless moral support they showed me during my entire academic journey.

Contents

List of Figures

1 Introduction

1.1 Problem statement

With the significant advances in the field of machine learning, the geophysical community is beginning to realise the potential of embracing such technologies to assist or even replace some of their conventional workflows.

One of the most crucial workflows in hydrocarbon exploration is the interpretation of seismic data, data that depict the underground structure of the Earth, and be able to draw meaningful conclusions from them. Seismic interpretation usually refers to classifying features of interest from the subsurface and it has a big variety of applications.

Hydrocarbon exploration is a 90 year old problem, and in short is the search for deposits of hydrocarbons, particularly petroleum and natural gas in the Earth. In the more recent years, in order for the field to respond to the increasing complexity of conditions and the fact that the world economy is starting to explore different energy approaches, such radical new techniques seem to play a key role in overcoming the new challenges.

However, the interpretation process requires a lot of time and effort from the field experts and for this reason, helping in speeding up such processes or even automating them entirely, would end up having a huge impact because it can unburden them from repeating and tedious tasks, which will create more time that is better allocated elsewhere while at the same time providing a significant reduction in cost of operations.

Seismic data are 3D volumes of underground structures, made during exploration and development of petroleum reservoirs and resemble a lot ultrasound images. A very common task when dealing with seismic data interpretation, is facies classification. Facies are seismic units composed of groups of reflections, whose parameters differ from those of adjacent facies units. Seismic data and facies will be described in more detail in Chapter 3.

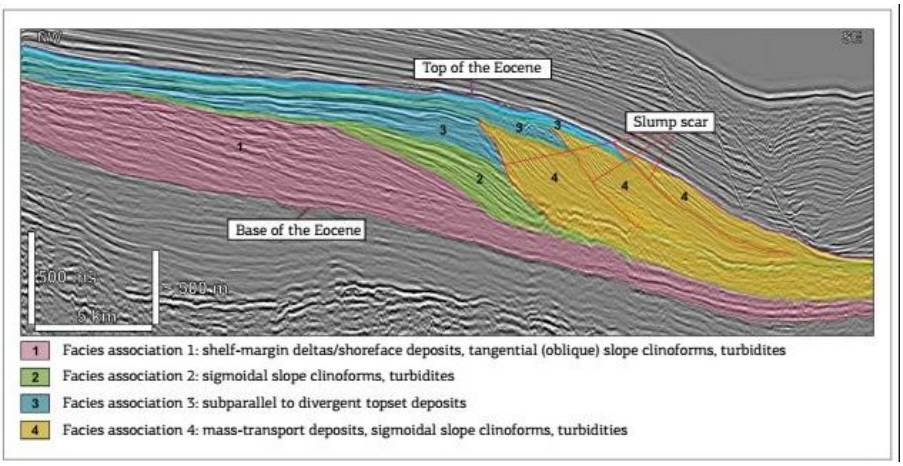


Figure 1: Seismic facies interpretation of a prograding shelf in the Santos Basin of Brazil [1]

To tackle this problem, the current best performing technique consists of training deep neural networks, and more specifically architectures that perform semantic segmentation. Semantic segmentation is the task of classifying every pixel of the image into a specific class.

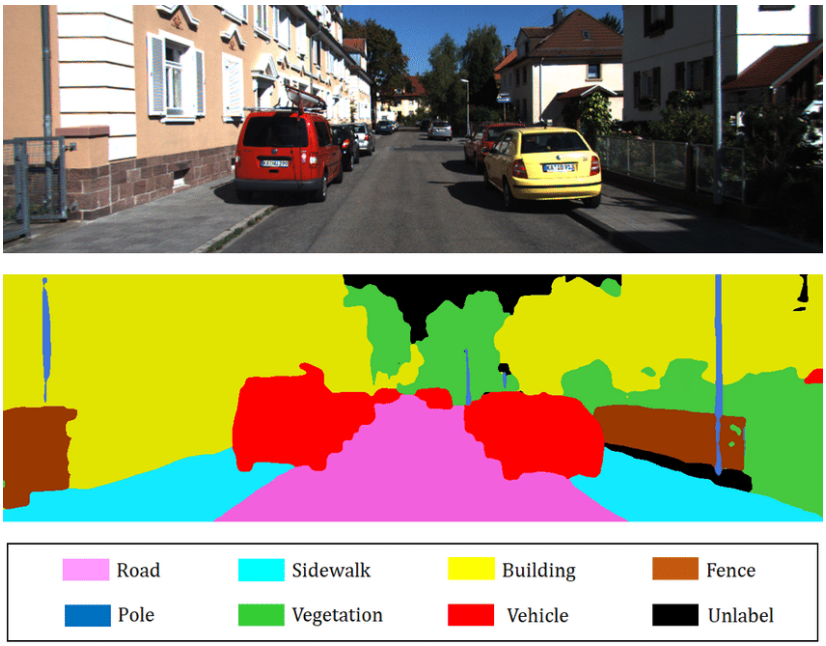


Figure 2: Example of semantic segmentation [2]

1.2 Motivation

From the point of view of machine learning, I believe that we have to stay in touch with new state of the art algorithms and architectures and try to see whether they can be transferred from one domain to another in order to improve the quality of results or address any problems that usually occur.

Seismic data and seismic interpretation in general, is a field that can be addressed by deploying machine learning algorithms specialized to the machine vision domain. While there are many things in common with the traditional techniques, there are also many differences and that opens up a lot of potential for further experimenting and fine-tuning and if possible, update the industry's current solutions with faster and more accurate tools and techniques.

1.3 Contributions

This thesis proposes the examination of the semantic segmentation on seismic data task, from the scope of few-shot learning. Acknowledging the fact that seismic labeled data acquisition is a major obstacle to overcome, we realise traditional supervised methods that rely heavily on a plethora of labeled data do not return optimal results in our scenario.

It is also a fact that seismic labels change a lot throughout the range of the slices of the cube, both in texture and in shape. Combining that with the data scarcity problem, most of the current implementations are having a hard time generalizing on labels that are annotated as being the same but are hundreds of slices apart.

Since few-shot learning is also known to be able to generalize on new classes with a fraction of the training data required, if we make the assumption that "class A" from slice 100 and "class A" from

slice 600 are considered different classes for the network, we can re-adjust the episodic paradigm [21] (the ”recipe” to use FSL networks) to train a few-shot learning architecture that is able to generalize well on the same classes, but in different slice depths.

We present a few-shot learning neural network architecture which consists of a series of modules. These modules are an encoder, a difference of Gaussian pyramid unit, the scale space representation layers and finally a bi-directional Conv-Lstm unit followed by a small decoder. Besides modifying the model architecture we also have adapt to the specific requirements to our specific problem by converting the network logic from binary classification to multi-class, addressing the image patching locality problem by injecting a global view of the labels to the network both during training and inference, we use pre trained weights to tackle the data scarcity and finally, provide a computationally efficient post processing approach that can be used with other existing seismic segmentation methods. At the same time the proposed method is also capable of reducing texture bias, which seems to be an important factor in seismic data. The specifics of each module and procedure are described in more detail in Chapter 4.

Finally in the end some alternative solutions are discussed together with improvements to the current one.

1.4 Thesis Outline

In Chapter 2 we present some basic theoretical background about machine learning and neural networks, specifically focusing on techniques and architectures relevant to semantic segmentation. Also we briefly mention of the current solutions for semantic segmentation on seismic data.

Chapter 3 explains some basic information about the collection process, processing and some basic format of the typical seismic data. Also it explains some common features of interest and gives among others a brief synopsis of seismic facies, the kind of features we use as labels in this thesis.

Chapter 4 describes the few-shot learning model this thesis is built upon, and the improvements and adjustments that had to be made to make this architecture work for our case scenario. Finally we describe our evaluation metrics and performance criteria.

In Chapter 5 the results of our work are presented and compared to current industry standard solutions and we discuss both the visual and quantitative results.

Finally in Chapter 6 a quick synopsis of our work is presented along with some suggestions for future work and improvements.

2 Machine learning for seismic data

This chapter briefly describes some basic background literature for machine learning algorithms, and how it is applied on seismic data interpretation.

2.1 Machine learning tasks

There are 2 major categories when training a machine learning algorithm, based on the way it learns from the provided data.

2.1.1 Supervised Learning

Supervised learning is the task of learning a function, that maps input and output data based on examples of input-output pairs. The set of such data is called the training set and the process of learning a function is called training. The goal is to approximate the mapping function so well that when you have new unseen input data, you can predict the output variables for that data in a reasonable way. Supervised learning can be further categorized based on their output:

Classification: The classification task refers to the use of machine learning algorithms that separate data from the problem domain into categories. A typical classification task is categorizing emails into ”spam” and ”not spam”.

Regression: Regression is the task of approximating a continuous output variable, or more simply learning to predict numerical values. Such examples are algorithms that predict prices (of cars, stocks) coordinates etc.

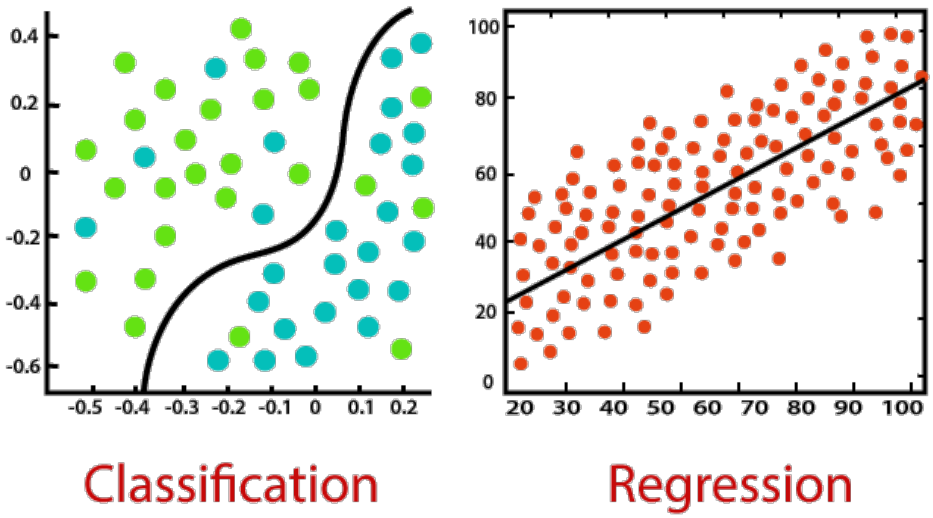


Figure 3: Classification vs regression [3]

2.1.2 Unsupervised Learning

Unsupervised learning is a machine learning technique in which you do not need labeled data and for this reason you do not need to ”supervise” the model. Instead, you provide only the input data, and

allow the algorithm to try and make sense by extracting features and patterns on its own.

There are also some more sub-categories like semi-supervised learning, which in essence are a combination of the above two techniques.

2.2 Neural network

Neural networks are a series of machine learning algorithms that aspire to mimic the way the human brain operates. A neural network is a collection of interconnected units or nodes called neurons. Each neuron is a mathematical operation, that receives an input, multiplies it by it's weights (plus a number called bias) and then passes the sum through an activation function to the other neurons.

$$Y_k = \phi(\sum_{j=0}^{m} \theta_j X_j + \theta_0) \tag{1}$$

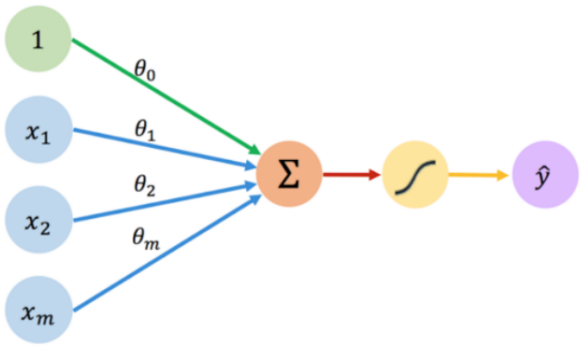


Figure 4: Artificial Neuron [4]

The activation function that is applied to every neuron, is the reason neural networks can approximate any continuous function, provided that a non-linear activation function is used and there is at least one hidden layer. This has been proven by the universal approximation theorem and it adds a theoretical justification as to why neural networks are such good universal approximators [22].

A neural network in its simplest form, consists of an input layer a hidden layer and an output layer connected in a sequential way. This is also called a shallow neural network. Information is fed into the input layer which transfers it to the hidden layer. The hidden layer defines the internal function of the network and finally the output layer provides the predictions in a form of a single value or a vector.

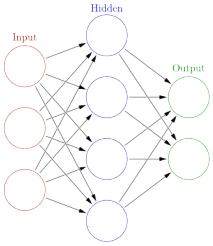


Figure 5: Shallow neural network [5]

The most popular algorithm to train a neural network, is called Backpropagation. The intuition behind it is that an error is calculated between the output prediction and the actual y values and then, this error is distributed back to the network through the network layers updating the parameters (weights and biases), using the chain rule for differentiation. The parameters are updated in such a way that the prediction error is minimized.

2.3 Deep learning for seismic data

In plain words a deep neural network is an artificial neural network as we explained above, with multiple layers between its input and output layers. The hidden layers do not necessary have to be fully connected to one another and also there are many ”layer architectures” that a neural network contains. Based on the task at hand, one must pick the corresponding neural network architecture to achieve the desired results.

As mentioned in the beginning of this chapter, seismic data interpretation is treated as a computer vision problem. The main difference with classic machine vision inputs (three-channeled RGB images) is that seismic data are deriving from sound waves and so they are one-channeled ”images” and they resemble a lot to medical ultrasound images.

For this reason, the most common neural network architectures chosen to tackle this problem are picked from the machine vision domain. Below we can find the most successful and popular ones.

2.3.1 Convolutional neural networks

Convolutional neural networks (ConvNets) are a class of neural networks, that are based on layers that employ a mathematical operation called convolution. They are used mainly for analyzing visual imagery. The architecture of a ConvNet is similar to the connectivity pattern of neurons in the human brain, and more specifically in the visual cortex. Individual neurons respond to stimuli in a restricted region of the visual field, also known as the receptive field. A collection of many such fields are used to cover the entire visual area.

The majority of CNN architectures are consisted of the same following layers.

Convolution Layer

The convolution layer performs a dot product between two matrices, where one of them is a set of learnable parameters (also referred as kernel) and the other is a portion of the receptive field. The kernel slides across the height and width of the input producing a two-dimensional representation also known as the activation map. The sliding movement of the kernel is called a stride and can be illustrated below.

Pooling Layer

Pooling layers are usually placed right after convolutional layers to progressively reduce the spatial size of the activation maps. The reason behind their use is to reduce the amount of parameters and computation in the network and also, for a more empirical explanation, to summarize the presence of the features in the input. There are many pooling layers, the most common of which are max pooling (keeps the maximum value in each patch of feature map) and average pooling (calculates the average in each patch).

Activation Layer

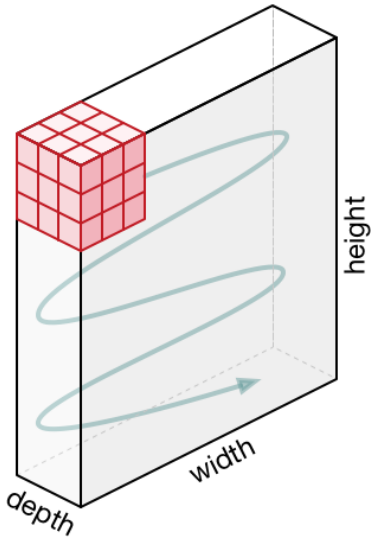
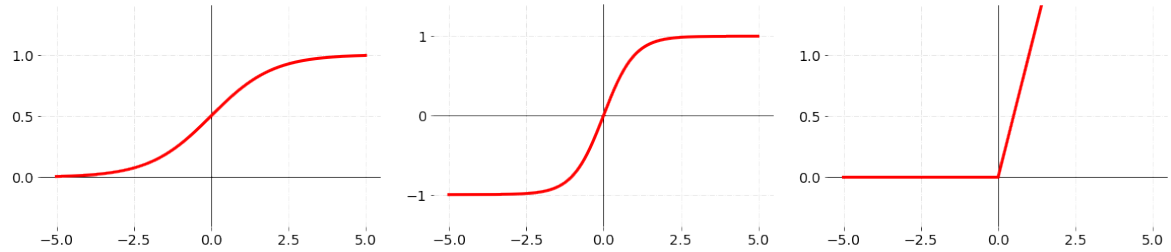


Figure 6: Kernel movement [6]

As mentioned above, activation layers are applied on the activation maps to allow them to learn complex non-linear relationships between the input and output data. The most common activation function nowadays is ReLu since its the least computationally expensive while also providing great results. We can see some non-linear activation functions in the figure below:



(a) Sigmoid activation function (b) Tanh activation function (c) ReLu activation function

Figure 7: Most common activation functions

Batch Normalization Layer

This layer applies the batch normalization technique, which is a method used to make neural networks faster and more stable through normalization of the input layer by centering and re-scaling. It has also proven to substantially reduce the training time of the network.

Dropout Layer

Dropout layers are introduced as a form of regularization on the network and they help prevent it from overfitting by becoming less sensitive to specific weights of neurons. Dropout works by randomly sub-sampling the outputs of a layer which has the effect of thinning the network.

LSTM Layer

The Long-Short Term Memory Layer, LSTM for short, is a layer architecture that is able to learn long term dependencies on data. They were initially introduced in 1997 [23] but over the years received a lot of refining. In contrast to feedforward networks, this architecture has feedback connec-

tions. A simple LSTM unit consists of a cell, an input gate, an output gate and a forget gate. The cell remembers values over some arbitrary time intervals while the other three gates regulate the flow of information that goes in and out of the cell. Nowadays lstm are not used as standalone networks because they got outperformed by more recent architectures both in performance and in training difficulty, but they are used as layers in parts of the architecture, to help in learning dependencies between other units of the network.

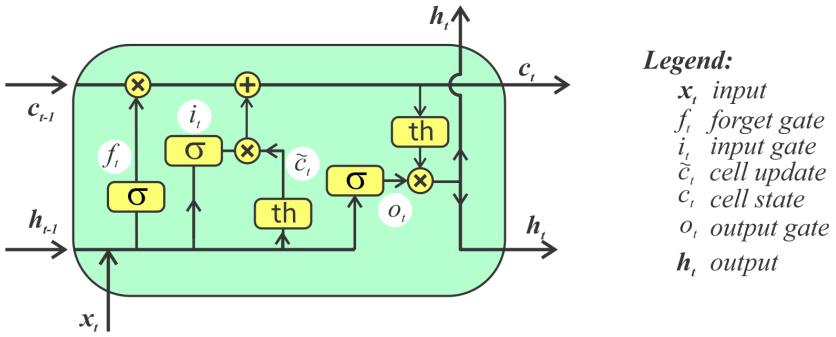


Figure 8: LSTM unit [7]

2.3.2 Residual Networks (ResNet)

Residual networks were introduced by the massively popular paper ”Deep Residual Learning for Image Recognition” in 2015 [9].

ResNet was invented to tackle the ”vanishing gradient” problem. Neural networks train via the backpropagation process, which relies on gradient descent, which aims to lower the loss function by finding weights that minimize it. But if the networks are too deep (there are too many layers stacked one after the other), repeated multiplication keeps making the gradient smaller and smaller until it vanishes causing performance to degrage, since weights are not updating its values.

To solve this problem, they introduced a simple and yet extremely elegant solution named ”skip connections”. Skip connections enable gradients to flow better during the backward step from later layers to initial filters.

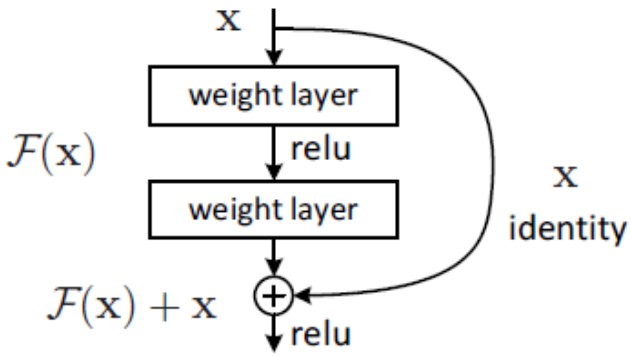


Figure 9: ResNet block [8]

This technique allows combining layers described in the subsection above, to create powerful feature extractors.

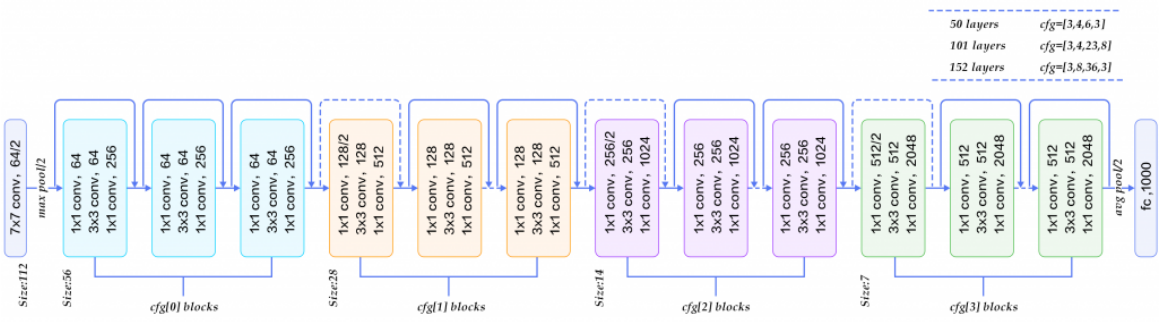


Figure 10: ResNet architecture [9]

Nowadays such networks are not used on their own, but rather act as ”backbone” networks. Backbone networks are established CNN architectures, that are usually initialized beforehand by training in a similar dataset, and act as the network's feature extractors.

2.3.3 Encoder-Decoder Architectures

In such networks, the encoder is responsible for extracting features from the image and the decoder ”analyzes” these features and tries to predict the output. These networks are typically consisted of 3 parts.

- The contracting path (encoder), which down samples the dimensions of the input.
- The hidden vector part, which in a sense ”processes” the feature map produced by the encoder.
- The expansive path (decoder) which up samples the dimensions of the image and gives the predictions of size equal to the input.

Downsampling layers

Downsampling is usually performed using pooling layers or dilated convolutions. Dilated convolution is a clever trick to reduce dimensionality using convolutional layers without pooling, by expanding the region of the kernel. This way the kernel covers a larger area of vision and produces ”fewer” results in its output, the same way a pooling layer would produce less results in the output than in the input.

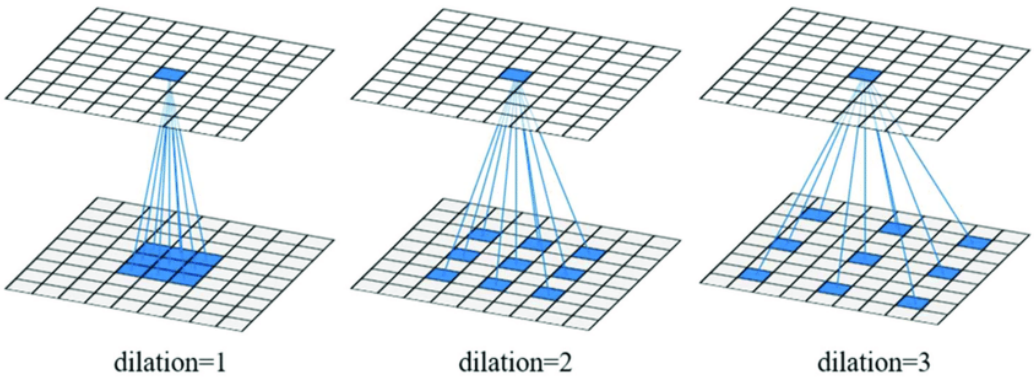


Figure 11: Dilated convolution [10]

Upsampling layers

Upsampling in the decoder is performed either by Upsampling Layers or by Transpose Convolution layers.

The upsampling layers are simple with no weights, and they just double the dimension of the input. This is usually accomplished using either bilinear interpolation or nearest neighbour interpolation.

The transpose convolution is a little more sophisticated. It is an inverse convolutional layer so it has it's own weights, and it will both upsample its input and also learn how to fill in details during the model training process.

The most popular and successful encoder-decoder architecture is U-Net [11]. It was released in 2015 and has won the Grand Challenge for Computer-Automated Detection of Caries in Bitewing Radiography at ISBI 2015, and it has won the Cell Tracking Challenge at ISBI 2015 on the two most challenging transmitted light microscopy categories (Phase contrast and DIC microscopy) by a large margin.

The encoder is a traditional stack of convolutional and downsampling layers and the decoder a stack of convolutional and upsampling layers. It uses all the techniques mentioned above but it comes with a great addition; it also includes concatenation layers to carry information from the encoder directly to the decoder, which helps a lot into restructuring the original information.

To this day there are many more semantic segmentation techniques like the PSP, DeepLabV3 etc but in the seismic industry the U-Net is considered gold standard.

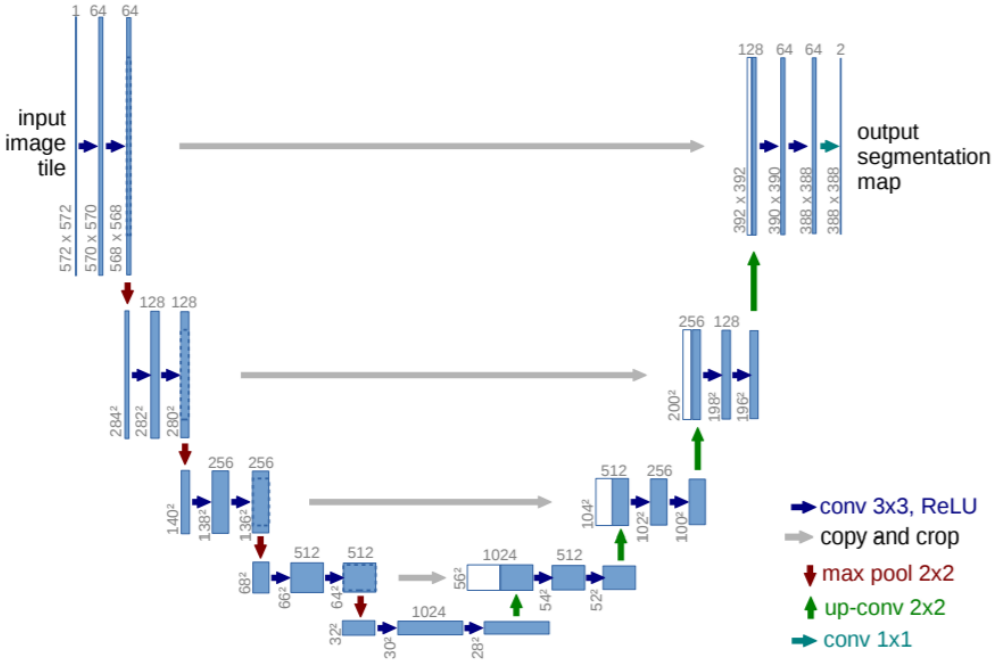


Figure 12: U-Net architecture [11]

3 Seismic data interpretation

Seismic interpretation is the extraction of geologic information about the subsurface of the earth. This process contains the steps of acquiring underground data, process them to transform them in a more refined form and finally interpret the results to draw meaningful conclusions. A very common objective of seismic interpretation is hydrocarbon exploration.

3.1 Seismic Survey

The most common method of generating the data used to extract this information, is called seismic survey. The process is based on the technique of determining the time interval that elapses between the initial time of a seismic wave produced from a selected point and the arrival of the reflected impulses at one or more different locations. The production of the seismic waves is performed with many ways, including air guns, falling weights (thumpers) or even underground dynamite explosions. At the point of arrival, the amplitude and timing of waves are recorded by detectors to produce a seismogram, which is a record of the produced ground vibrations. An important detail is that reflections from different depths return at different times and so the gathered data contain both depth information and also position. This process can be executed both on land and off shore via survey ships and hydrophones.

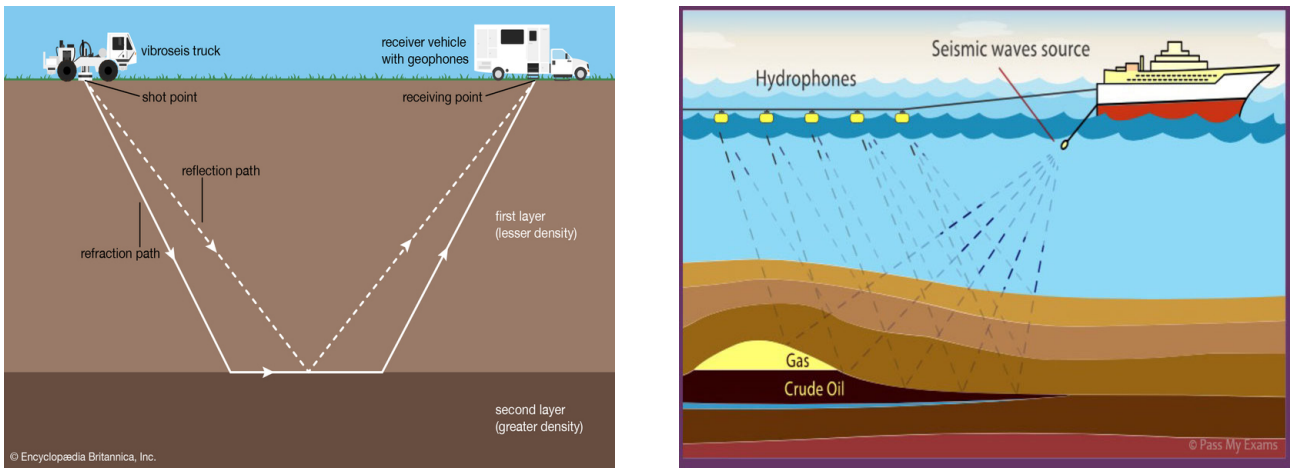


Figure 13: Land [12] and off-shore survey [13]

The measured seismic waves are then processed into 3d volumes, highlighting interfaces and different properties between rocks including also fluid contents in the rocks' pore spaces.

However, the data acquired by such methods only provide an indirect measurement of the underground geology and is often noisy and most of the time require additional processing to reach a more easy to interpret form. The final visual results resemble a lot ultrasound images used in the medical domain but on a much larger scale. For example, a typical seismic survey can easily cover $25km \times 25km$ in surface and $10km$ in depth, which translates in many millions number of pixels.

3.2 Seismic data processing

The data collected by the receivers are in a raw format, which is referred to by geophysicists as pre-stack format. The pre-stack usually is a class of data called "gather", a collection of seismic traces that share some common geometric characteristics. The term usually involves a common image point

(CIP) or common mid-point (CMP) gather and are used to examine the important geometric attributes such as amplitude, frequency content, phase noise etc.

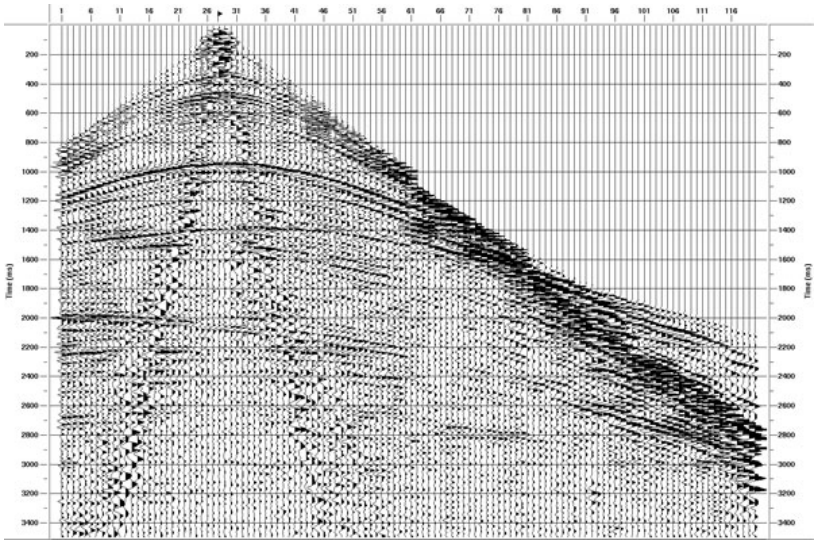


Figure 14: Seismic gather [14]

There is a variety of techniques available to process gathers and convert them to seismic images. Two of the most common ones are called migration and normal moveout correction (NMO) with stacking.

The migration process presumes that the CMP gather can be migrated, by handling the reflection hyperbola as a diffraction hyperbola. This way, and assuming we do not have information about the velocity, we migrate with many trial velocities and evaluate the results. Finally when the velocity in the migration is equal to the medium velocity, the diffraction hyperbola collapses to its peak, which is at the zero-offset trace.

Normal moveout correction (NMO) is a process through which we can correct the measured traveltime of a reflected sound wave t at a given offset x, to obtain the traveltime at normal incidence t_0 by using the equation:

$$t_0^2 = t^2 - \frac{x^2}{u_{NMO}^2} \tag{2}$$

where u_{NMO} is the NMO velocity. There are many variations of this equation but this one is its simplest form and when it is applied to a CMP section, it turns the hyperbola created by the reflection into a straight horizontal line. It is based on the assumption that in a CMP gather, reflection traveltimes as a functions of offset follow hyperbolic trajectories and it is able to remove the moveout effect on traveltimes. Then, traces in each CMP gather are summoned to form a stacked trace at every midpoint location. As a result of this moveout correction traces are extended in a time varying way causing their frequency to shift toward the low end of the spectrum.

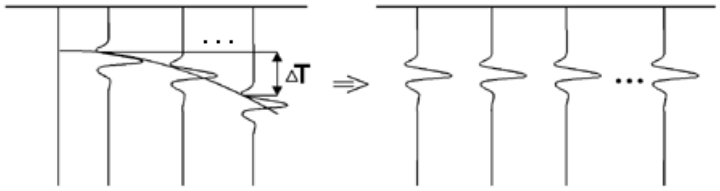


Figure 15: NMO and stacking [15]

The figure below [24] illustrates the two techniques side by side. Image (a) shows a CMP gather from a single reflector in a constant-velocity medium. Image (b) shows the velocity spectrum obtained by migrating using a number of constant velocities and displaying the zero-offset for each and finally image (c), the velocity spectrum derived by NMO correction and stacking.

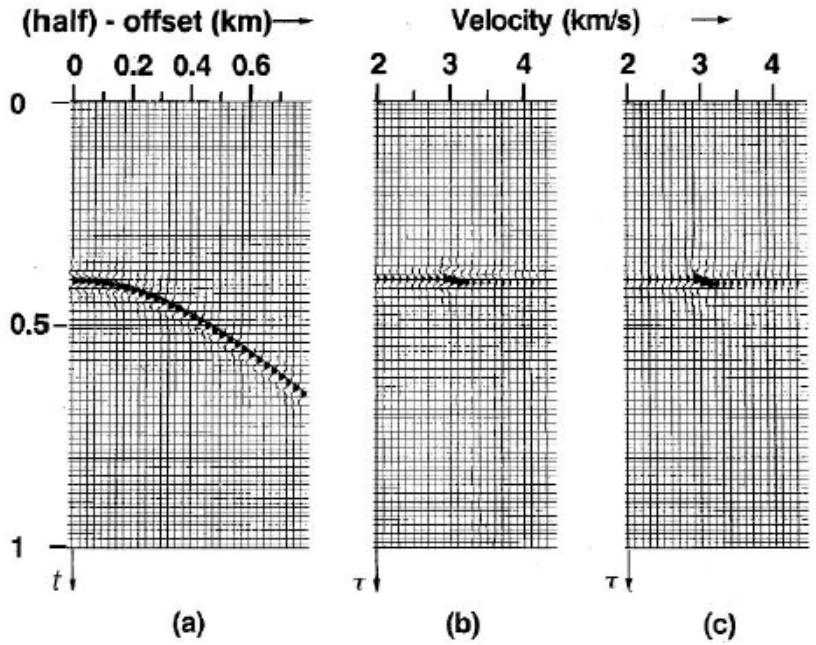


Figure 16: Gather processing techniques

3.3 Seismic data interpretation

After finishing the processing procedure, we have in possession the data in the seismic images format and the next step in the process is the interpretation of the data. This means an interpreter has to construct the most accurate earth model or reservoir description possible. This can be done by drawing with either pencil on paper or a cursor to screen and based on his or her geological understanding, the most likely interpretation from the many valid interpretations the data suggests.

There are many geological features of interest that make up the subsurface, but in this chapter we will mention only the ones that contribute to the visual interpretation of the data rather than the quantitative one. Below are some of the most common geological features that are used in the seismic data interpretation domain.

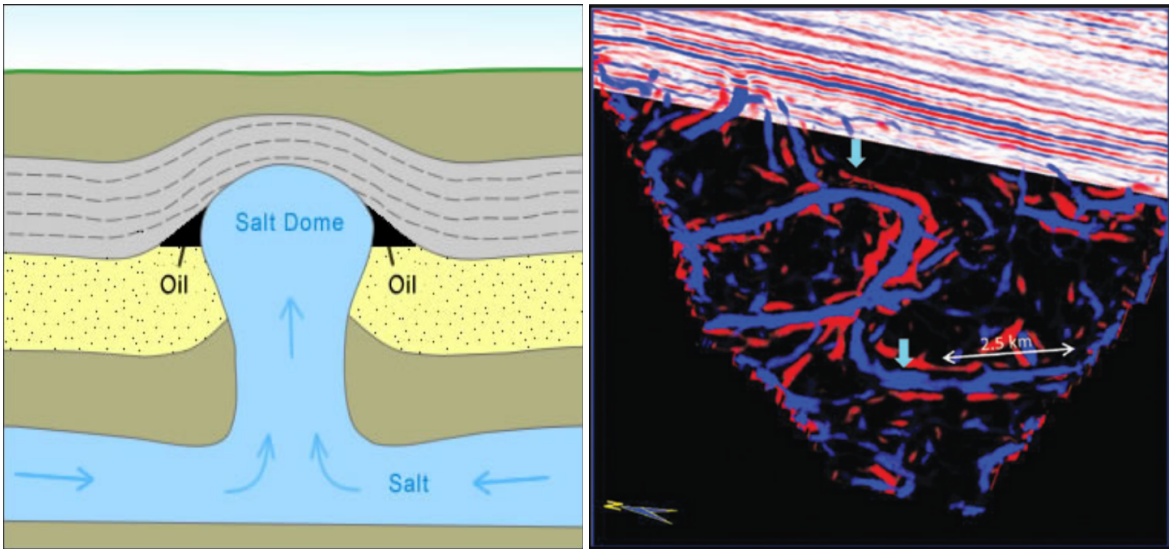
3.3.1 Geobodies

Seismic geobodies are seismic attributes that use mathematical relationships based on the geometry and physical properties of the subsurface and they create many different features of interest.

A good example of geobodies are salt domes and salt diapirs, which are massive structures of salt within the Earth and play a crucial role in hydrocarbon exploration. However, it can be very tricky to interpret salt, as it is associated with weak reflections and is hard to differentiate from other sediments.

Another example of geobodies is a seismic channel, a linear and concave-based depression, through which water and sediment can flow and can be deposited in unique bodies. They can have a big variety in morphologies like straight, meandering or braided.

Both salt bodies and channels are great indicators of hydrocarbon traps.



(a) Salt dome [25] (b) Seismic channels (in blue color) [26]

Figure 17: Seismic geobodies

3.3.2 Horizons

A horizon feature normally refers to a bedding surface, a type of subsurface made up of piles of layers of sediments or volcanic rocks placed one on top of another. These layers are separated from each other by changes in their stratigraphic surface, either lithostratigraphic or chronostratigraphic. From a geology-centric view, there is always uncertainty when trying to pinpoint the exact location of horizons and one can never be entirely sure about the true relation between the seismic horizons and the corresponding geological surfaces and it is considered one of the unsolved problems in exploration geophysics.

3.3.3 Discontinuities

Seismic discontinuities are surfaces where the measured velocity of seismic waves change abruptly. These abrupt changes are indicators of features of interests, the most common of which are seismic faults.

A fault is a fracture in a volume of rock, where there is a significant displacement as a result of mass movement. Usually they are created within the Earth's crust from the action of plate tectonic

forces and the largest of them form types of boundaries between the plates such as subduction zones or transform faults. They can be created either fast by earthquakes or slowly, by aseismic creep. A fault plane is the area that represents the fracture surface of a fault and a fault trace (or fault line) is a place where the fault is visible and can be mapped on the surface.

In the figure below we can see the mapping of 5 horizons and 2 faults.

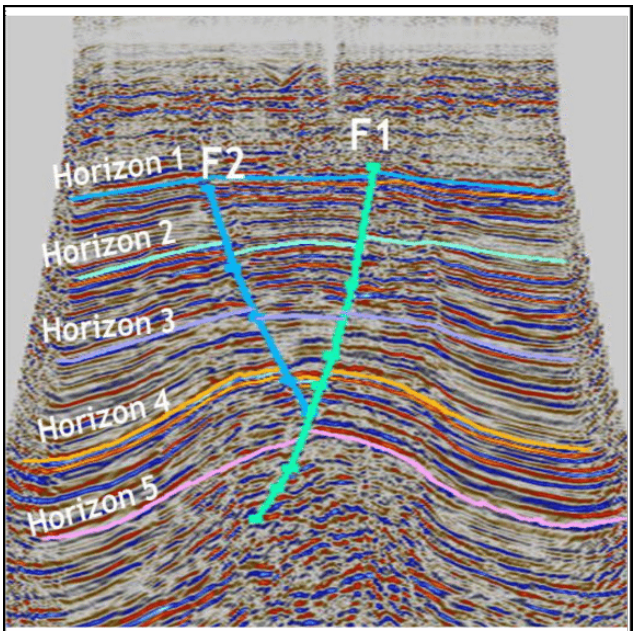


Figure 18: Horizons with faults [16]

3.3.4 Facies

Seismic facies is a technical term for regions inside the Earth with rocks of similar composition or characteristics that are placed in a common environment. They are interfaces outlined by echoes in seismic images and mark boundaries between regions that contain sediments by different types of geological processes. The body of facies can have any observable attribute of rocks such as their overall appearance, composition of formation and their changes over a geographic area. The similar characteristics between facies can be chemical, physical or biological features that distinguishes them from their adjacent rocks. Examples of different sedimentary environments and facies can be:

- Mixes of sand, silt or mud placed in fan-shaped delta at the mouth of a river.
- Coarse sandy sediments in a meandering river channel.
- Fine-grained sediments in a shallow lakebed.

Seismic facies is the geological features we are going to use as inputs in the dataset of this thesis and we will describe them further in the following chapters.

3.3.5 Indicative datasets

This sections illustrates a few seismic datasets and their seismic interpretation.

Parihaka Dataset

This dataset originates from the ”Seismic Facies identification challenge”, an online AI competition [19]. They provide a seismic cube from a public-domain seismic survey called Parihaka, which was made available by the New Zealand government. The cube is consisted of $1006 \times 782 \times 590$ pixels and each pixel is classified into one of 6 different facies by expert geologists.

This is the first dataset we use to test our method.

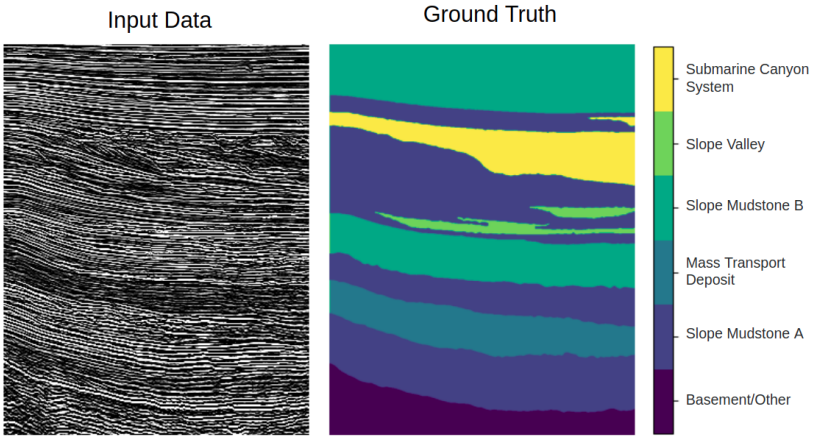


Figure 19: Parihaka dataset example

F3 Netherlands dataset

This dataset originates from a paper [27], whose claimed purpose is to help with the lack of free available labeled seismic data, by providing the well known F3-Netherlands dataset with labels to serve as benchmark for new machine learning methods. In their version, instead of providing the dataset and labels as cubes they simply give the generated inlines and crosslines as images.

Their interpretation resulted in identifying 9 horizons, which they name H1, H2, ..., H9 sorted in descending order of geological age and the labeled images are created by taking the intersection between the seismic lines and the horizon surfaces. Since they do not use a specific name for each label and refer to them by the number of horizon that is above them, we use the same notation in our thesis as well. This results in 9 labels $[1-9]$, plus the label 0 which corresponds to zero in input.

This is the second dataset we use to test our method.

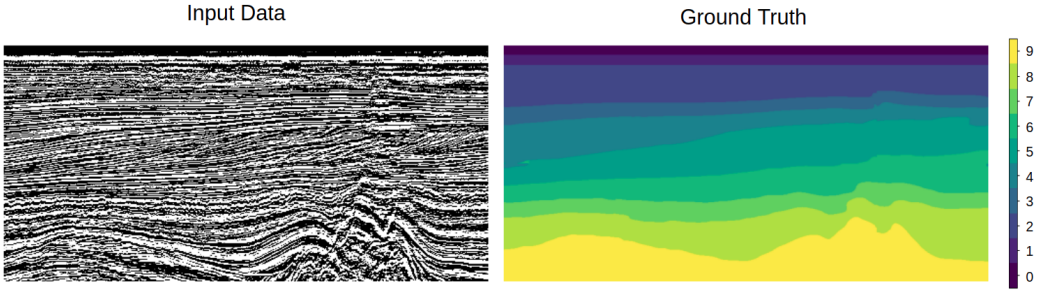


Figure 20: F3 Netherlands dataset example

Penobscot dataset

This dataset also originates from a paper [28] and originates from the public 3d seismic survey called Penobscot 3D. As with the F3 Netherlands dataset they also provide the labeled seismic data to ”foster machine learning development” and they interpret seven horizons H1, H2, ..., H7 numbered

from the highest depth to the lowest. They divide the seismic cube into eight intervals between the horizons who share some common geological features.

We did not use this dataset for our results and the figure below originates from their paper.

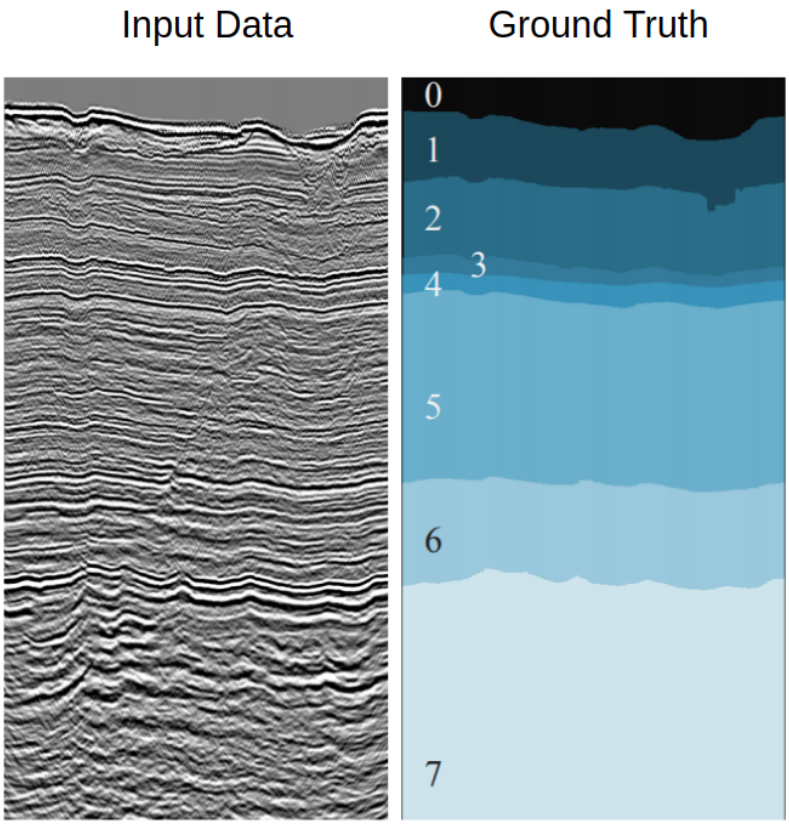


Figure 21: Penobscot dataset example

TGS Salt Dataset

There are many more different datasets containing seismic objects of interest besides facies. One famous example is the dataset from the online kaggle competition "TGS Salt Identification Challenge" that depicts salt deposits. It consists of images that are 101×101 pixels and each pixel is classified as either salt or sediment. This dataset is used as benchmark in many seismic salt detection oriented papers.

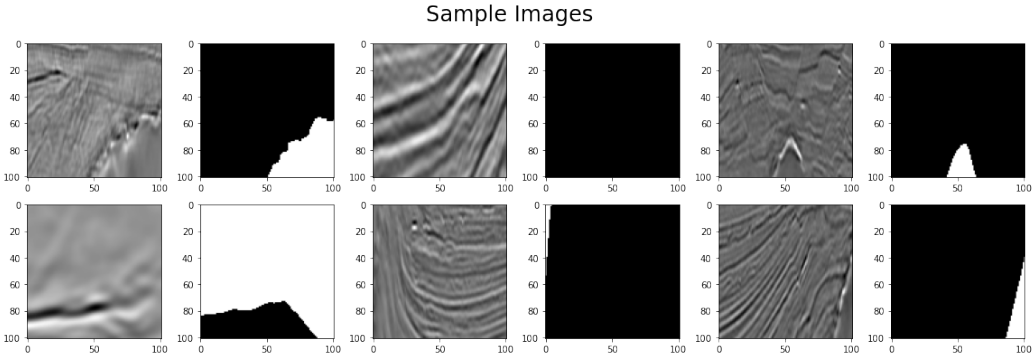


Figure 22: TGS salt dataset example

4 Methodology

In Chapter 2 there was provided some literature review on machine learning and more specifically on algorithms suited for seismic data. In Chapter 3, there was a presentation on the seismic data in more detail, from the way they are generated, processed and the kind of data relevant to this thesis.

In this Chapter, there will be a presentation of the proposed solution, why it was chosen and how it works in more detail. Finally the results of our solution will be displayed in order to showcase our contribution in a more clear way.

4.1 Seismic interpretation problems

Good seismic datasets, are pretty hard to manufacture. There is great cost in acquiring the raw data from underground, a lot of processing to transform them in their final ready-for-training form and finally, there is great effort in creating labels for such data. The problem in acquiring quality labeled data comes down to three factors:

- Labeling can be done only by geologists, experts in their field. That means it is very expensive and time consuming.
- Much of the information contained in a geological region, is not easily (if at all) transferable to new regions.
- When training on large images, it becomes a necessity to use image patches in order to perform semantic segmentation. This creates a problem of topological loss of information.

When creating a dataset to tackle traditional machine vision problems, for example ImageNet, it contains classes of everyday objects like people and animals that in theory anybody can classify. For this reason one can outsource such tasks to large groups of non explicitly trained people, which reduces the cost and time to manufacture significantly, when compared to labeling seismic facies that is usually done by small groups of experts.

In addition, even when such labels are indeed created, due to the nature of the data, identified classes from one geographical region, are not easily transferable to other regions. In more classical problems for example, if you create a classifier that detects a specific type of animal, it should be able to detect instances of this animal in variety of new poses or different background etc. This problem is highlighted also in our results, where we get good accuracy of a class in the slices near the training dataset, but the further away we move from these training slices we notice a significant drop in accuracy even though we are examining the same class. Finally, seismic facies classification does not require only contextual and shape information, but also requires some topological orientation. For example classifying class ”B” would require class ”A” being on top of it and class ”C” being below it.

4.2 Few shot learning for Seismic data

Considering the above reasons, we propose two things:

- Approach this problem using the few-shot segmentation technique, to help in generalization leveraging only a few data.

- Apply a unique neural network architecture that assists in solving both the transferability problem and the localization problem caused by image patching.

While humans are able to recognize new classes of objects without requiring having seen instances of it, most machine learning algorithms need thousands of examples in order to achieve similar performance. Few-shot learning is a technique aspiring to do just that, train a neural network that is able to generalize well using only very few instances. In this case, we are using a subsection of it called few shot segmentation. Since most current architectures require to have a strong backbone network as feature extractor, and it is considered very hard to create one using such few data, the most common approach is to create a classifier on bigger datasets and then apply them to their architecture. For this reason few shot learning is characterized by many as a meta-learning problem.

Few-shot learning considers of three datasets, the train dataset $D_{train} = \{(X_i, Y_i)\}_{i=1}^{N_{train}}$, the support dataset $D_{support} = \{(X_i, Y_i)\}_{i=1}^{N_{support}}$ and a test dataset $D_{test} = \{(X_i)\}_{i=1}^{N_{train}}$, where $X_i \in \mathbb{R}$ is the image and $Y_i \in \{0,1\}$ is its corresponding pixel level mask for binary classification. The classes are shared among the support and test set but are disjoint with the training set $\{C_{train}\} \cap \{C_{support}\} = \emptyset$ and the goal is to train the network on D_{train} and then to be able to generalize and provide good results to a class on the D_{test}.

The few-shot semantic segmentation task also assumes we the standard notation of N-way and K-shot classification, meaning we try to classify N classes using K examples of each. This paradigm is followed both during training and also during inference and is known as the episodic paradigm [21]. To make use of this technique each training episode is generated by:

- Sampling a support training set from D_{train} for k classes and their corresponding binary masks.
- Sampling a query set also from D_{train} for the same k classes and its corresponding binary masks.

For example, if we are using 5-way 1-shot segmentation and we sample the classes C_4, C_5, C_2, C_1, C_3 as query set and 1 sample of each class, then the classes from support set must be the same and in the exact same order but with different instances of each class (the sampling of class instances is also random, so in come cases the same class instance could be selected both in query and support set).

The input of the network is composed of $[S_{image}, S_{mask}, Q_{image}]$ and the produced segmentation results are compared with Q_{mask} to calculate the training set metrics.

During testing phase each episode is generated by:

- Sampling a support set from $D_{support}$ for k classes and their corresponding binary masks.
- Sampling a query set from D_{test} for k classes and their corresponding binary masks set.

with the same principles explained in the training example in mind, and the segmentation results of the network are used for the evaluation of test set metrics.

This means if we want to generate predictions for a new class image, we must already have at least k-shot labeled samples of the given class

4.3 Original network architecture

To address all the above problems, the architecture chosen was based on the paper ”On the Texture Bias for Few-Shot CNN Segmentation” by Reza Azad et al. [17]. They refer to it as DoG-LSTM and its architecture is illustrated in high level in the figure below.

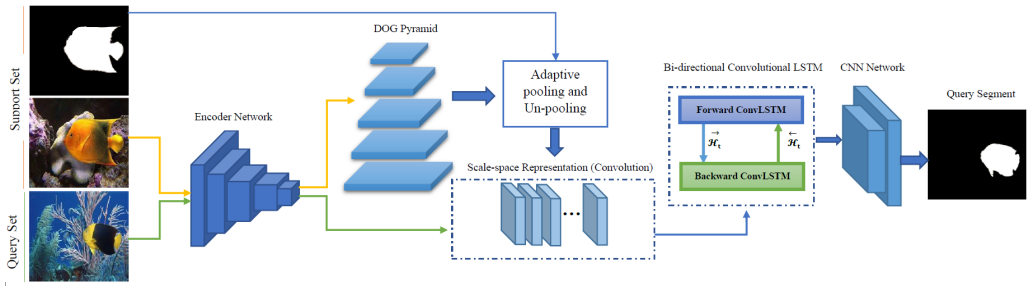


Figure 23: DoG-LSTM architecture [17]

To train and infer with this model two sets of items are required, a query set and a support set. The query set is the input whose mask we want to predict and the support set is a set of input and its mask, depicting an object of the same class as the query set but of a different instance. So the input data of the model are three images $Q_{image}, S_{image}, S_{mask}$ and the output is the query mask Q_{mask}, the predictions of the network for the query image.

The network first applies the difference of gaussians technique on S_{image} to attenuate high-frequency components on the feature map. At the same time the S_{mask} is down-sampled to match the spatial resolution of the feature maps of S_{image}. Then, after being combined together and go through pooling and unpooling layers they are merged with the feature map produced by the S_{image}. Their product serves as input to the Bi-directional Conv LSTM unit which merges the information from all these different sources and finally, it goes through the decoder part where it is upsampled and produces the final Q_{mask}.

4.3.1 Encoder Network

The encoder network unit is responsible for extracting image feature maps, both from the *Qimage* and the S_{image}. The input of the encoder is the same as the input image dimensions $W \times H \times C$ where W and H are the width and height and C is the channel dimension, which in this case is equal to 3 one for every R,G,B channel of the given photo.

The architecture used to generate this feature space in this paper is either a variation of ResNet architecture [9] or VGG 16 [29] architecture. It is a very common strategy to use classic and proven network architectures that are able to provide great feature maps either on their entirety, or extracting small parts of it. The main difference between these two is that Resnet uses skip connections to concatenate the Convolution-Maxpool layers, a term explained in Chapter 2.

Another very common technique is the use of pre-trained weights on the encoder layers, that are already trained on another dataset. Typically this other dataset is huge, consisting of many classes and also a great variation of these classes and for this reason it is able to create meaningful representations without requiring much further training. While there is still the possibility to train the encoder from scratch, this method is very useful to help the network converge faster and also create more robust encoders when dealing with a situation of having a small amount of data available.

4.3.2 DoG Pyramid unit

Recent studies on CNN's show that despite initial beliefs, they tend to have bias to the texture of the object they are trying classify and this paper suggests this can be one of the reasons that they achieve high performance only when large labeled datasets are provided. By removing texture bias, we give

incentive to the network to base predictions more on the shape of the objects, which is closer to the way the human visual cortex works.

A more intuitive way to think about this problem is by comparing it with trying to recognize objects in different distances. When dealing with objects that are far away, while we may be able to identify them we can only see some rough details but in contrast, when objects are getting closer we can see more and more fine-grained details. However, the ideal scale space level to identify important features for every object is unknown, and so by blurring the image with different Gaussian distribution values, each produced result represents a different scale-space level and we can emulate features on different 'distance'.

To reduce this texture bias, this architecture proposes the integration of a set of difference of gaussians (DoGs) into the learned feature map, to attenuate high frequency local components like texture. This idea is inspired by the paper "Distinctive Image Features from Scale-Invariant Keypoints" by David G. Lowe [30].

To achieve this, the network first uses a CNN as an encoder to produce a feature map of the input images. This results in $F_s \in \mathbb{R}^{W\times H\times M}$ for S_{image} and $F_q \in \mathbb{R}^{W\times H\times M}$ for Q_{image} where W, H and M represent width, height and feature dimensionality on the latent space. Then, to encode the high frequency information a DoG is applied on every channel $m \in M$ of the feature map from the F_s which is formulated as:

$$G_s = \Gamma_{\sigma_1,\sigma_2}(F_s) = (F_s^m * \frac{1}{2\pi\sigma_2^2} exp^{-\frac{x^2+y^2}{2\sigma_2}}) - (F_s^m * \frac{1}{2\pi\sigma_1^2} exp^{-\frac{x^2+y^2}{2\sigma_1}}), \forall m \in M \tag{3}$$

where σ_1 and σ_2 ($\sigma_2 > \sigma_1$) are the variance of the Gaussian filters, x and y represent the spatial position in the encoded feature map and $*$ is the convolution operator. This results in multiple feature maps produced for every single image and thus creating distinctive invariant features for each object. This way we generate L levels of representation for every support sample which can be denoted as $G_s^l \in \mathbb{R}^{W\times H\times M}, \forall l \in L$. According to both [17] and [30] the optimal setting suggests that 5 scale levels give the optimal results, so in this implementation we set L=4 to add to the original one.

The new encoded feature maps at each scale G_s^l are averaged over the known foreground regions in the support mask $Y_s(c)$ and so, in each level we can estimate

$$f_s^l = \frac{1}{\mid \tilde{Y}_s(c) \mid} \sum_{i=1}^{WH} G_s^l \tilde{Y}_s(c) \tag{4}$$

where $\tilde{Y}_s(c) \in \{0,1\}^{H\times W}$ is the down sampled mask that matches the resolution of the feature maps G_s^l and $|\tilde{Y}_s(c)| = \sum_i \tilde{Y}_{s,i}(c)$ is the number of foreground locations in $\tilde{Y}_s(c)$. After that every representation is unpooled to match the spatial resolution of the query feature map F_q and then are convolved to it.

Finally a scale space representation (SSR) is defined, which serves as input to the Bi-directional convLSTM unit and it can be formulated as a convolution between the class representative feature maps and the feature map derived from the Q_{image}:

$$SSR = \{BN(\psi_s^l * F_q)\}, \forall l \in L \tag{5}$$

where ψ_s^l are the upsampled f_s^l prototypes and BN is a batch normalization layer.

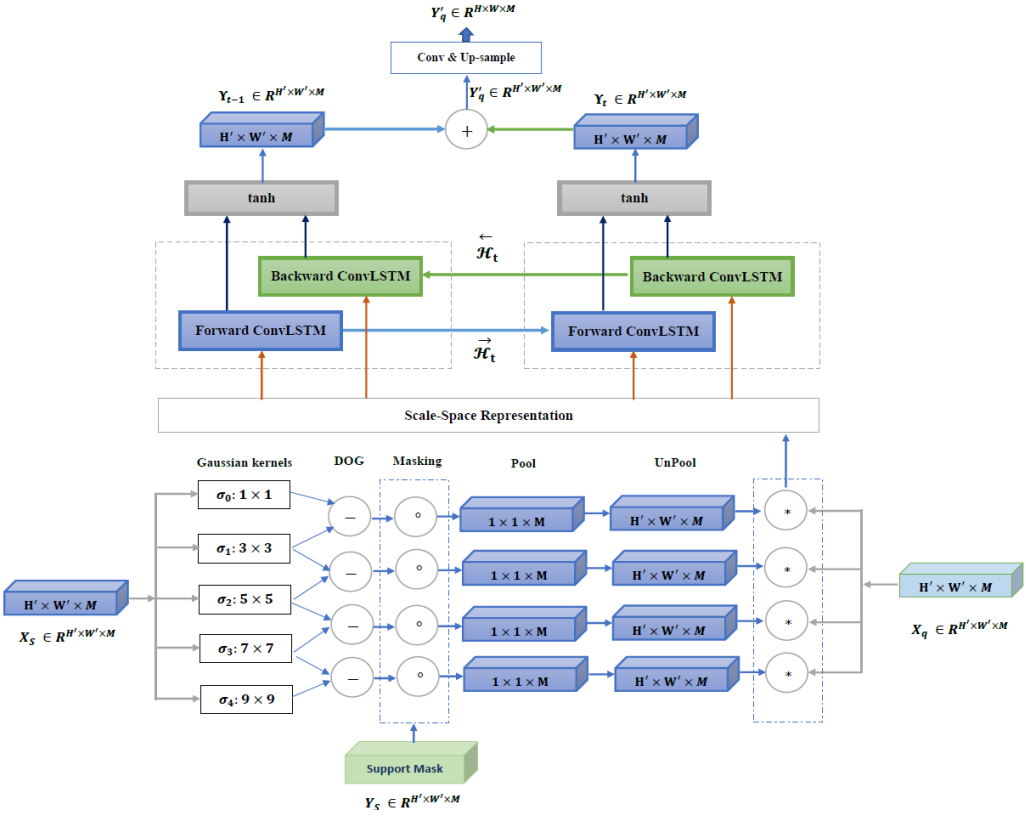


Figure 24: SSR encoding block [17]

4.3.3 Bi-directional convLSTM unit

The input of the bi-directional convLSTM unit is the output of the SSR unit where the query feature map is fused with the multi-scale class representations from the support features ψ_s^l. The paper [17] suggests that while logical or average operations may be the straightforward solution to obtain unique representations, they fail to utilize inner relationship data and for this reason they reformulate the problem as a sequential task.

The convLSTM (Long-Short Term Memory) unit, was introduced in 2015 [18] to address the limitations of LSTMs in various tasks, such as semantic segmentation. The main drawback in handling spatiotemporal data, was the usage of full connections in input-to-state and state-to-state transitions where no spatial information was encoded. Conv-LSTMs introduce a convolution operator in the state-to-state and input-to-state transitions which helps in preserving spatial information in input images. All inputs $X_1,...,X_t$, cell outputs $C_1,...,Ct$, hidden states $H_1,...,H_t$ and gates i_t, f_t, o_t are 3d tensors whose last two dimensions are spatial dimensions (rows and columns). The gate functions are shown [18] below, where * denotes convolution and ∘ denotes the Hadamard product:

$$i_t = \sigma(W_{xi} * X_t + W_{hi} * H_{t-1} + W_{ci} \circ Ct - 1 + b_i) \tag{6}$$

$$f_t = \sigma(W_{xf} * X_t + W_{hf} * H_{t-1} + W_{cf} \circ Ct - 1 + b_f) \tag{7}$$

$$o_t = \sigma(W_{xo} * X_t + W_{ho} * H_{t-1} + W_{co} \circ Ct - 1 + b_o) \tag{8}$$

where X_t and H_t denote the input, in our case the SSR unit, and hidden state at time t, W_x, W_h and W_c the set of learnable parameters and b is the bias term. The generated proposal for the cell state is:

$$C_t = f_t \circ C_{t-1} + i_t \circ tanh(W_{xc} * X_t + W_{hc} * H_{t-1} + b_c) \tag{9}$$

Finally, the new hidden state is estimated as:

$$H_t = o_t \circ tanh(C_t) \tag{10}$$

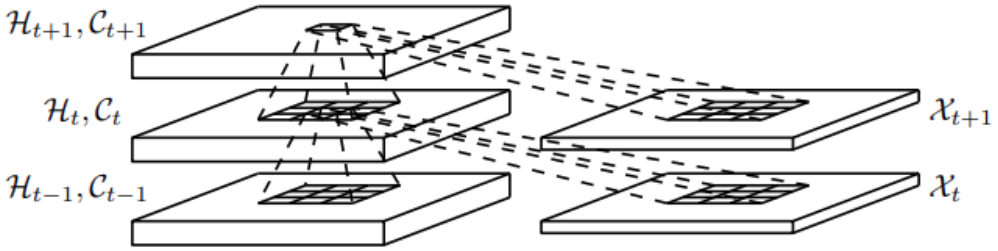


Figure 25: Inner structure of ConvLSTM unit [18]

An interesting addition to the classic LSTM unit, which remembers and preserves information from the "past", is the bidirectional LSTM. This unit runs the input data in two ways (from start to end and then from end to start) by using two separate LSTM units and this way, it is able to also preserve information from the "future". Empirically they show good results as they seem to understand context in data better and are preferred when all the timesteps of the input data are available beforehand. The downside is that this technique increases the parameters of the initial unit. The output prediction for a query image is defined as:

$$\hat{Y}^q = tanh(W_y^{\overrightarrow{H}} * \overrightarrow{H} + W_y^{\overleftarrow{H}} * \overleftarrow{H} + b) \tag{11}$$

where $\overrightarrow{H}$ and $\overleftarrow{H}$ are the hidden states of the forward and backward convLSTM units and b is the bias term.

4.4 Our approach

To apply the above architecture to our problem domain, we have to make quite a few adaptations. First and foremost, we modify the few-shot learning paradigm in such a way that support classes are the same $\{C_{train}\} = \{C_{support}\} = \{C_{test}\}$ in all three datasets $D_{train}, D_{support}, D_{test}$, but in different geological locations (or different "slices") in the seismic cube.

This happens because as we explained in the Introduction, due to the nature of seismic data we make the assumption that labels of the same type but from different regions of the seismic cube are considered to be different labels. For example "class A" from slice 100 is considered to be different from "class A" in slice 650.

To generate a training episode:

- For the query set we sample data from D_{train} for k classes and their corresponding masks.
- For the support set we also sample data from D_{train} for k classes and their corresponding masks.

And to generate a test episode:

- For the query set we sample data from D_{test} for k classes and their corresponding masks.
- For the support we set pick data from $D_{support}$ but in our case, the support image and corresponding masks are slices from the training set that are the closest to the ones in D_{test} for the given data slice

To give a quick example of the above, assuming we trained on slices 100, 150 and 200 (the D_{train}), when we want to generate predictions for slice 101, the Q_{image}, Q_{mask} are the data from slice 101 and the S_{image}, S_{mask} are the data from slice 100, for slice 160 the S_{image}, S_{mask} are the data from slice 150, for slice 190 the S_{image}, S_{mask} are the data from slice 200 etc.

Having the same classes both in train and in test set may sound like a trivial task, but as explained in previous chapters this is a big obstacle to overcome when working with seismic data. For instance a very common task is to generate predictions for a given seismic cube with as little labeled data as possible, since generating such labeled data is expensive, time consuming and usually non-transferable. Non-transferable means that obtaining labeled data from class A in regions of seismic cube A does not translate to producing useful features for the same class A in other regions of seismic cube A let alone using them for a different seismic cube B.

The figure below illustrates what we mentioned above. It originates from a semantic segmentation facies identification challenge (not few-shot) and we can see both train and the two test sets are part of the same seismic cube.

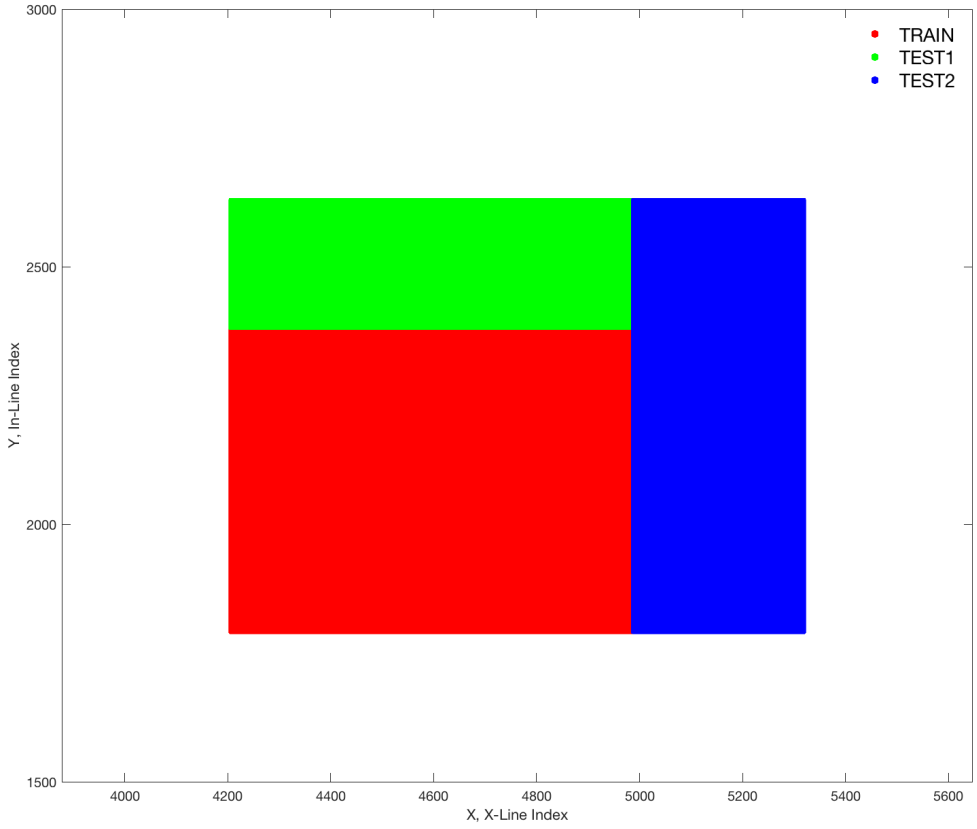
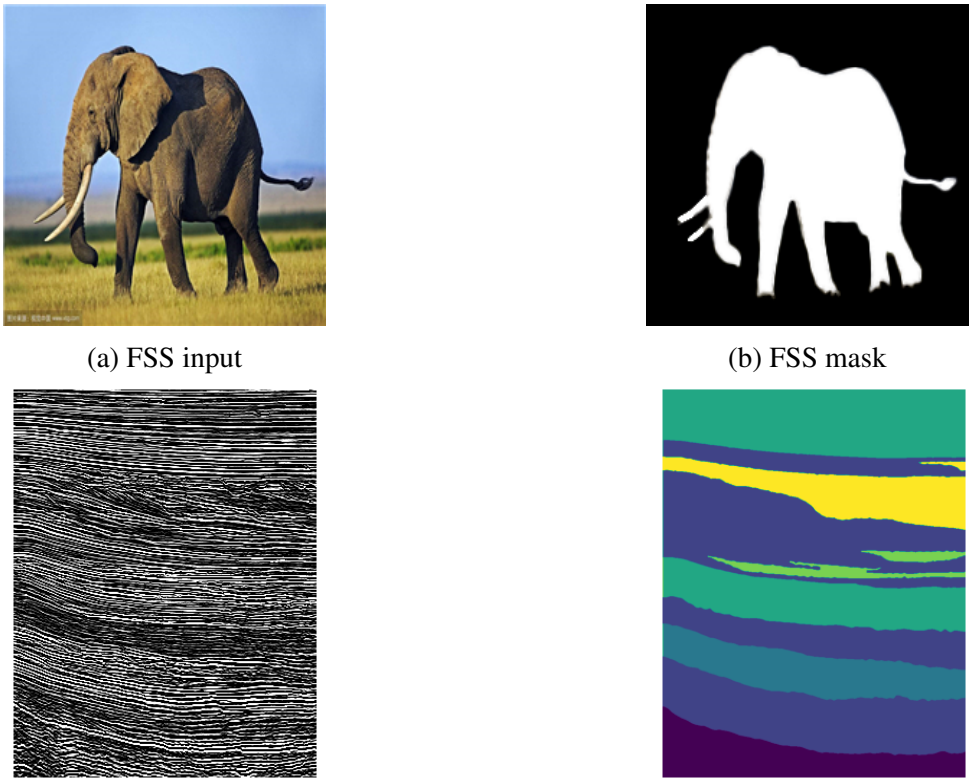


Figure 26: Parihaka test regions [19]

The difficulty of this task is also further demonstrated in the ”results” chapter where we compare our predictions with the latest industry standard solution.

The second big difference between the two implementations, is the structure of the labels. In FSS-1000 the masks of the images are binary, meaning the mask for every given class instance is 1 where the item is depicted in the image and 0 where it is not, while in the case the masks of the images are multi class meaning that in every mask for every image, multiple objects are depicted.



(a) FSS input

(b) FSS mask

(c) Parihaka slice input

(d) Parihaka slice mask

Figure 27: Binary vs multiclass difference

The architecture that we use indicates that for every Q_{image} we can provide only one Q_{mask} so with respect to that, we converted our implementation from multi class to one-vs-all. This means that during training for every multiclass image mask, we create many different binary masks.

During inference, in order to combine the binary predictions back together to create the final multiclass prediction again, for every image input we run inference as many times as there are number of classes, each time using as S_{mask} a different class and selecting the max predictions.

Another significant difference we have, is in the amount of data available for training. In the work of [17] they train and evaluate on the FSS-1000 dataset [31], which is a famous dataset for few-shot segmentation. It contains 1000 classes of common every day objects and each one of them is consisted of 10 such class instances, which is a very respectable amount of data to be able to train a robust neural network. In our case, since we have to examine every seismic cube separately, we end up with datasets consisting only of classes that the specific seismic cube contains. In addition, we also take it one step further and use only a very small subset of the trained seismic cube, 10 slices to be more specific and so instead of 10000 images that the FSS dataset contains we end up having in our disposal only Classes $\times$ Class instances $\times$ Patches of samples of training images.

This brings us to our next major difference the use of image patching. The FSS-100 dataset consists of images that have dimension 224x224x3 while a typical seismic cube slice is much bigger, e.g. from the parihaka dataset that we used [19], the slice size is 1006x782x1. Image sizes this big, cannot

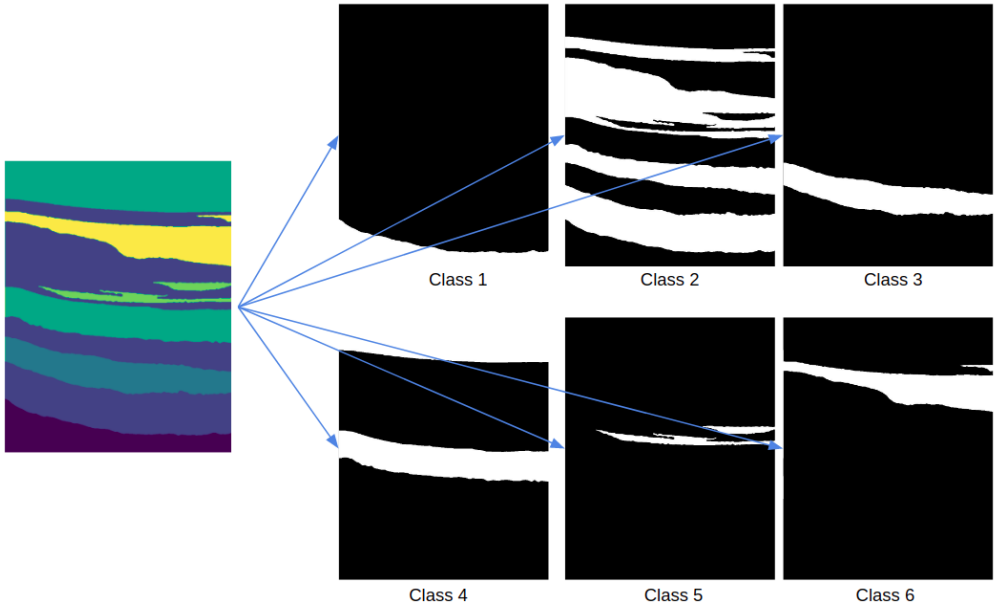


Figure 28: Multiclass to binary

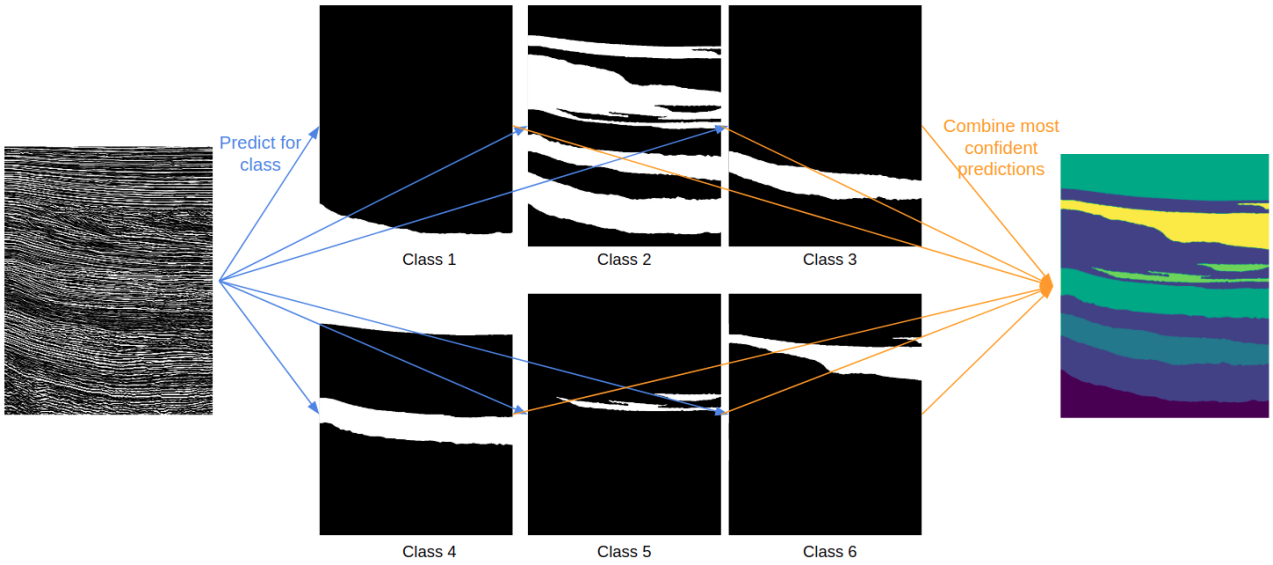


Figure 29: One-vs-all

be immediately fed into a neural network due to current hardware limitations. To overcome this, we use a very common technique called image patching, which breaks down an image into smaller parts of arbitrary size. This creates a major disadvantage for our network because in seismic data interpretation, the location of each class in the image plays a critical role in its correct identification, especially for classes that do not have their own distinct shape. For example in the figure above, class 4 and class 3 look a lot alike and their main distinction, besides their input values, is also their topological relation. So when the neural network sees only a part of them during a smaller image patch, it was no way to determine their original ”depth” in the image.

For most traditional semantic segmentation architectures this is a pretty big problem, to which there is still not a good answer that we are aware of. However, in our case by using this specific

few-shot segmentation neural network we are able to address this problem applying a very elegant solution. For every Q_{image} and Q_{mask} that are a patch of the original image, instead of passing through the S_{image} and S_{mask} of some other randomly sampled class patch instance, which is the first obvious answer, we supply a resized version of the whole image and its corresponding class. This was a major breakthrough in our implementation and even from the first time we applied it the network went from being unable to converge to basically providing state-of-the-art predictions.

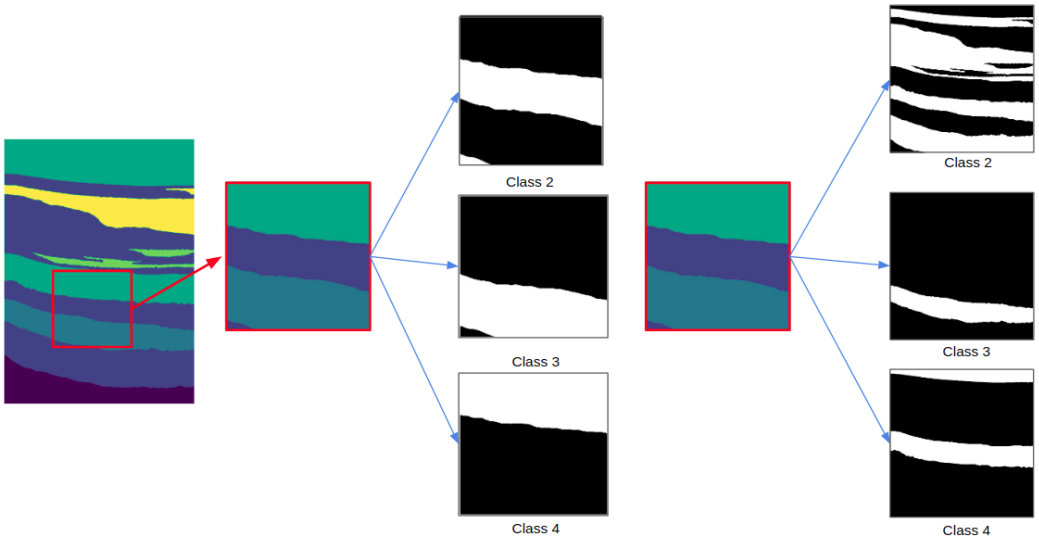


Figure 30: Image patching problem and solution

Finally another modification we had to make was on the encoder module. Due to the fact that 10 slices are not considered enough to train an efficient feature extractor, we decided to go with the solution of transfer learning, meaning we loaded pre-trained weights on the layers of the encoder. However the traditional trained weights that are available, do not originate from networks that accept the same data inputs as ours. The ones we ended up using were trained on ImageNet, which is trained on 3 channeled images as we mentioned earlier. To bypass this limitation we ended up duplicating our Q_{image} three times and thus artificially converting our image input from $(W \times H \times 1)$ to $(W \times H \times 3)$ and furthermore, we also had to scale our data to the original mean and standard deviation values of ImageNet.

As far as the encoder architecture goes, we experimented with many versions and ended up using a simple skip connection architecture consisting of a few convolution and two max pooling layers, concatenated in the end. This architecture is considered one of the many building blocks of the overall ResNet architecture.

4.5 Performance Criteria

4.5.1 Metrics

The task of seismic facies interpretation is treated as a machine vision and more specifically as a semantic segmentation one. For this reason the metrics used to evaluate quality of predictions will be the traditional ones used in this field which are:

- Intersection-Over-Union (Iou or Jaccard Index)

- F1 score (or Dice Coefficient)
- Overal pixel accuracy

Intersection-over-Union

The Intersection-Over-Union is one of the most popular metrics in semantic segmentation both for its simplicity and its effectiveness and is also very easy to understand because it is very intuitive. It is also referred as the Jaccard index and it measures similarity between finite sample sets.

IoU is the area of overlap between the predicted segmentation of the model and the ground truth segmentation divided by the area of the union of the two, as it is portrayed by the figure below:

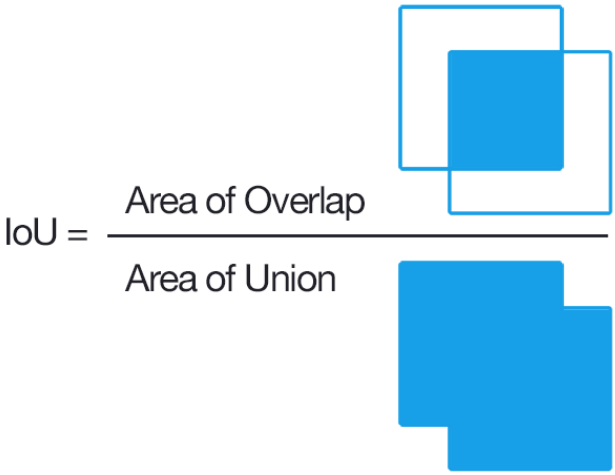


Figure 31: Intersection-Over-union [20]

and its equation is:

$$IoU = \frac{|A \cap B|}{|A \cup B|} \tag{12}$$

This metric ranges from 0-1 with 0 indicating there is no overlap and 1 that the two objects perfectly overlap with each other. For binary or multi-class segmentation, the IoU of the image is calculated by computing the IoU of each class and average them.

This is the main metric we are going to focus on or results

F1 score

The F1 score, also known as Sørensen–Dice coefficient, is a statistic also used to calculate the similarity of two samples. It is measured by multiplying the overlap of the ground truth and corresponding predictions by 2, divided by their total number of pixels. Its equation is:

$$F1 = \frac{2 * |A \cap B|}{|A| + |B|} \tag{13}$$

and it can also be illustrated like:

The f1 score is very similar to the IoU and they are positively correlated, which means if one model scores better than another using the first metric then it will be the same for the other metric as well. Finally this metric also ranges from 0-1 with 0 being the lowest and 1 being the highest.

Overal Accuarcy

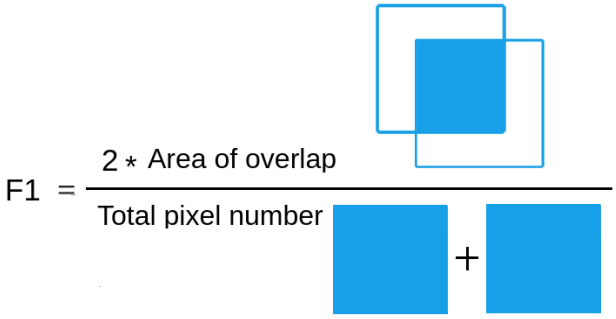


Figure 32: F1 score

Finally, one last metric that is available for semantic segmentation is the overall pixel accuracy. It is very intuitive and the easiest to understand as it is simple the percent of pixels in our predictions that are classified correctly. Traditionally this metric is not used to draw useful insights as it is very misleading, especially when having strong class imbalance between the data and the class we are most interested in is the minority.

4.5.2 Defining validation set

As mentioned earlier, moving away from slices that the model was trained on creates a significant drop in accuracy. For this reason, we wanted to measure the accuracy of the model both near slices that it was exposed to during training and also measure its robustness on slices that are away from the training ones. To achieve that, we avoided spreading the training slices all over the line range of the seismic cube and we organized the training and validation data as follows:

- Although the dataset we work on is fully labeled, we used only 10 slices for training to emulate the more realistic data shortage scenario, and try to solve the problem using the few-shot learning technique.

- In order to try and include as much area of the cube as possible, we picked two labeled slices every 100 slices, both of them relatively close to each other in order to create both short and long unlabeled cube intervals.

- The two slices that are relatively close to each other have 10 unlabeled slices in between. This is the first range of validation data that is created and we refer to it as the Validation range 1. This helps us check the generalization of the model close to the training set.

- Between the second slice and the next one, there are 80 unlabeled slices in between which creates a much bigger unlabeled slice interval which we refer to as the Validation range 2. This is a very good way to check the robustness of the model on slices that are very far away from the training set.

The below figure illustrates the validation ranges:

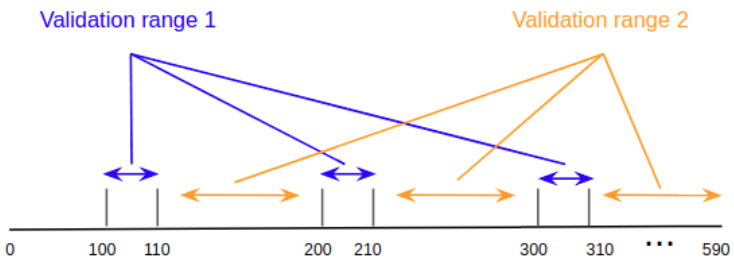


Figure 33: Validation slice ranges

5 Experimental results and validation

In this chapter we present our results and analyze them both visually and quantitatively. We used two different seismic cubes to create two different datasets.

We compare our results with a U-Net architecture neural network which is the industry standard at the time of writing this thesis.

We are going to examine a few slices from the 3 different slice ranges training slices range, validation range 1 and validation range 2 for both datasets. For every slice we mention the exact slice number, the range that it is part of and the support slice we used as S_{mask} for our implementation. The rationale behind choosing every S_{mask} is that if the slice is part of the training range then we use itself, while for the validation ranges we provide the closest training set mask available.

5.1 Parihaka dataset

The first dataset originates from the ”Seismic Facies identification challenge”, a brief explanation of which was given in Chapter 3.

While we have hundreds of slices in our disposal, as we mentioned in previous chapters we want to emulate the few-shot learning scenario, and for this reason we used only 10 of them for training. Also, to see the drop-off in accuracy as we move further away from ”training set slices” we did not separate them evenly in the whole range of available slices.

So, the training slices chosen for the Parihaka dataset are $\{100, 110, 200, 210, 300, 310, 400, 410, 500, 510\}$. Here is a preview of 5 of the slices used for training where we can see how their labels shape and differentiate over the range of the data.

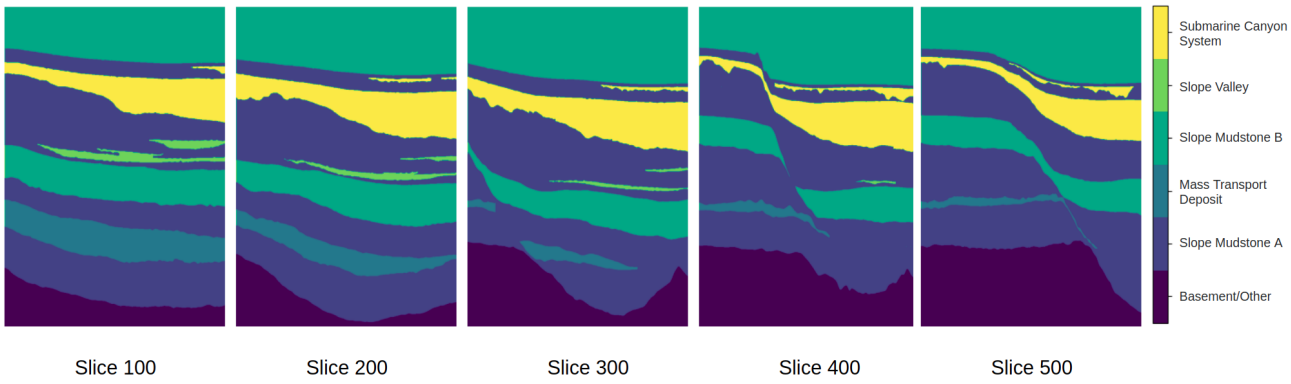
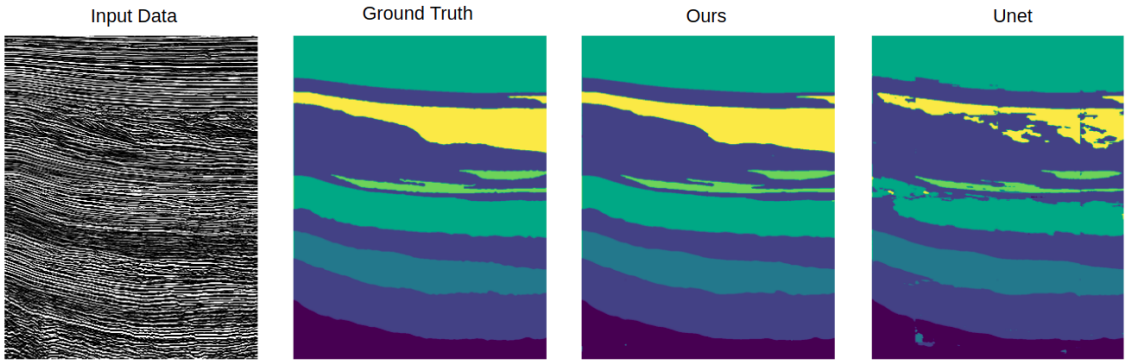


Figure 34: Parihaka dataset

As we can see the 6 labels change shape drastically which by itself is a very difficult problem to solve, which is only made worse by the compulsory use of image patching, since every slice dimension is 1006×782 and cannot be supplied as a whole as input in a neural network. The size of image patches is 256×256.

Below we can see the comparison of our results and the results of the U-Net model. In the results from both our method and from U-Net we did not apply any post processing.

Slice 100 - Training dataset

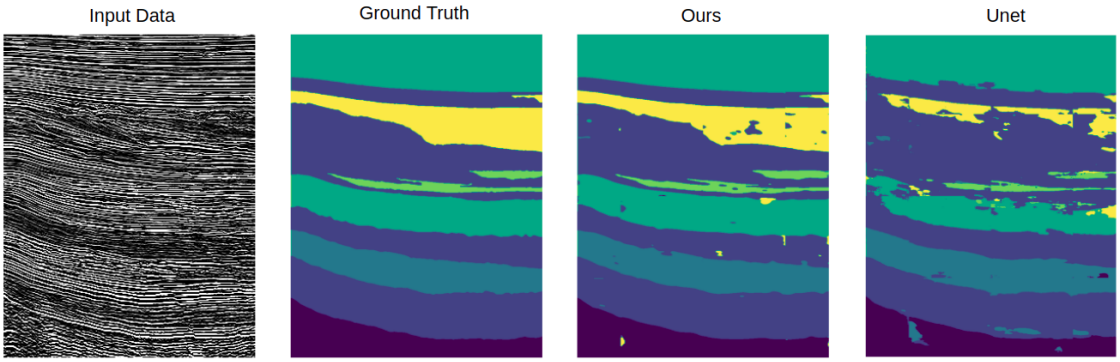


Slice number: 100 - support slice number used: 100

m_iou: 0.982
f1: 0.991
precision: 0.991
recall: 0.991
accuracy: 0.995

m_iou: 0.851
f1: 0.914
precision: 0.883
recall: 0.966
accuracy: 0.933

Slice 105 - Validation range 1

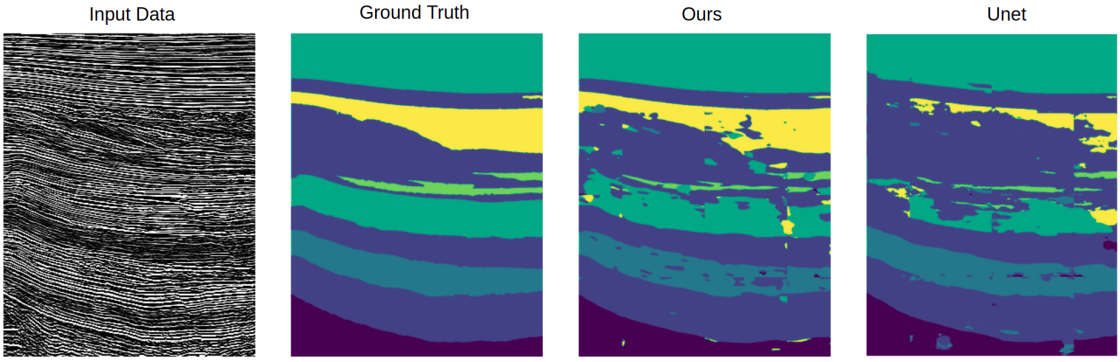


Slice number: 105 - support slice number used: 100

m_iou: 0.937
f1: 0.966
precision: 0.968
recall: 0.965
accuracy: 0.979

m_iou: 0.731
f1: 0.827
precision: 0.796
recall: 0.901
accuracy: 0.885

Slice 120 - Validation range 2

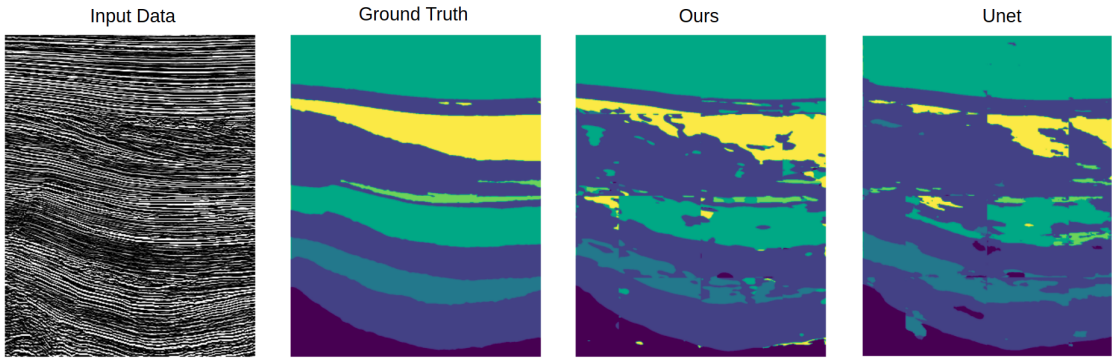


Slice number: 120 - support slice number used: 110

m_iou: 0.794
f1: 0.873
precision: 0.847
recall: 0.920
accuracy: 0.921

m_iou: 0.646
f1: 0.758
precision: 0.725
recall: 0.849
accuracy: 0.839

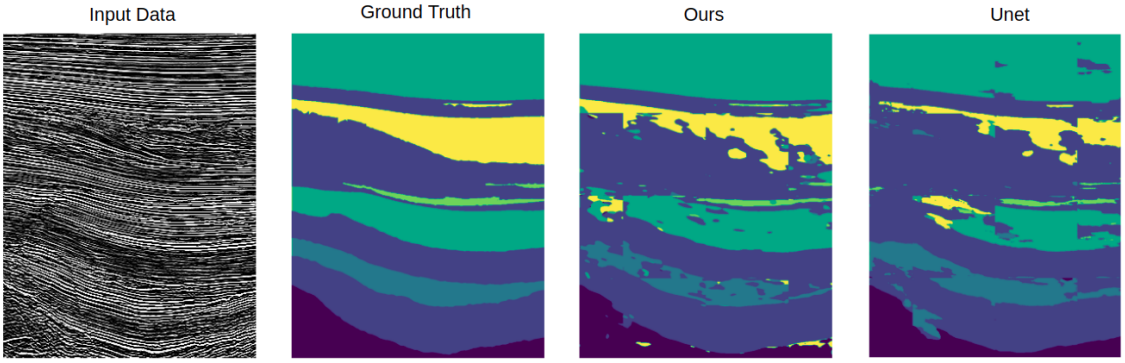
Slice 160 - Validation range 2



Slice number: 160 - support slice number used: 200

Ours	Unet
m_iou: 0.697	m_iou: 0.588
f1: 0.799	f1: 0.711
precision: 0.760	precision: 0.687
recall: 0.864	recall: 0.778
accuracy: 0.882	accuracy: 0.815

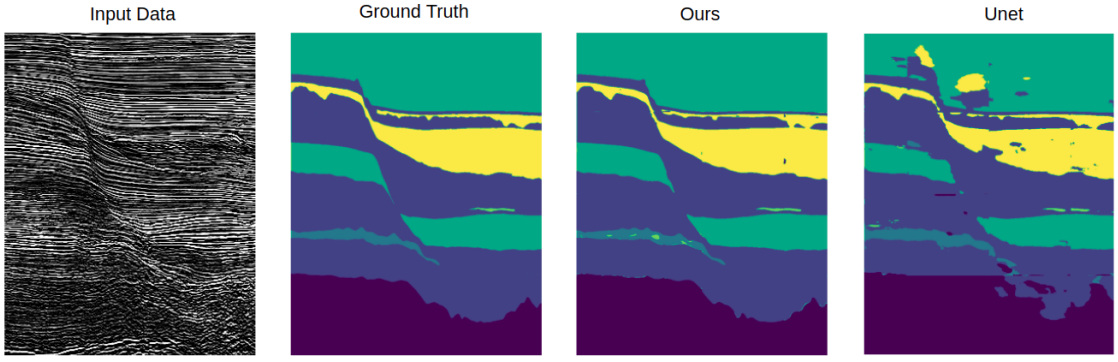
Slice 180 - Validation range 2



Slice number: 180 - support slice number used: 200

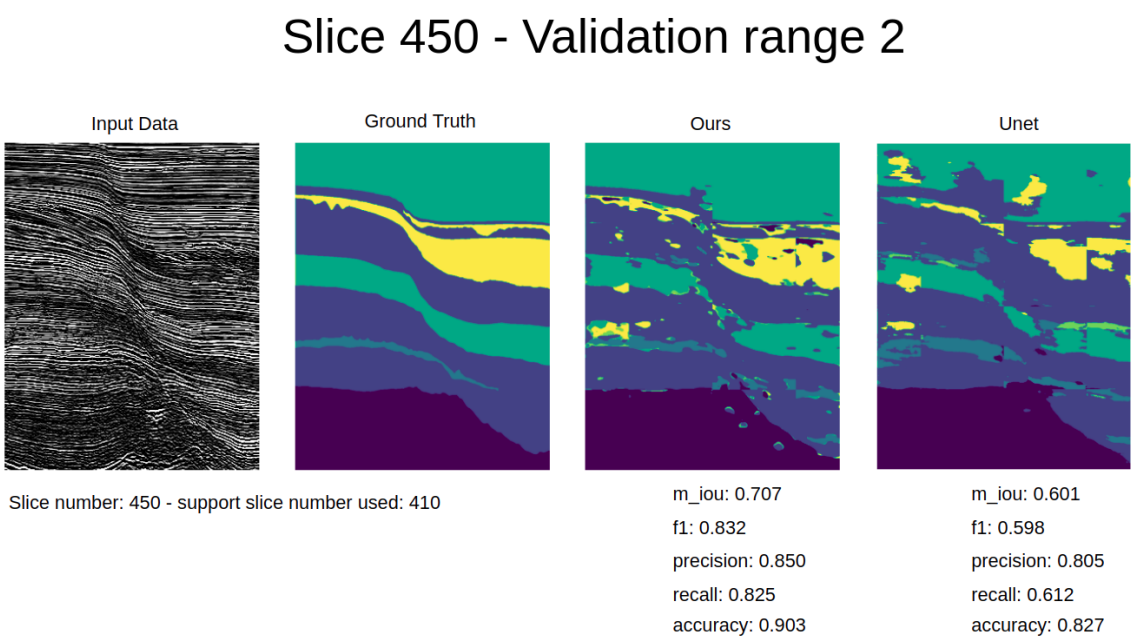
Ours	Unet
m_iou: 0.733	m_iou: 0.646
f1: 0.832	f1: 0.764
precision: 0.801	precision: 0.731
recall: 0.874	recall: 0.835
accuracy: 0.904	accuracy: 0.847

Slice 400 - Training dataset



Slice number: 400 - support slice number used: 400

Ours	Unet
m_iou: 0.814	m_iou: 0.748
f1: 0.824	f1: 0.844
precision: 0.823	precision: 0.906
recall: 0.991	recall: 0.805
accuracy: 0.992	accuracy: 0.919



Finally, we plotted a mean IoU over line number graph, to compare the mean Intersection over Union of the two models on a whole range of slices. In the figure below we can see the results for the range (100-200).

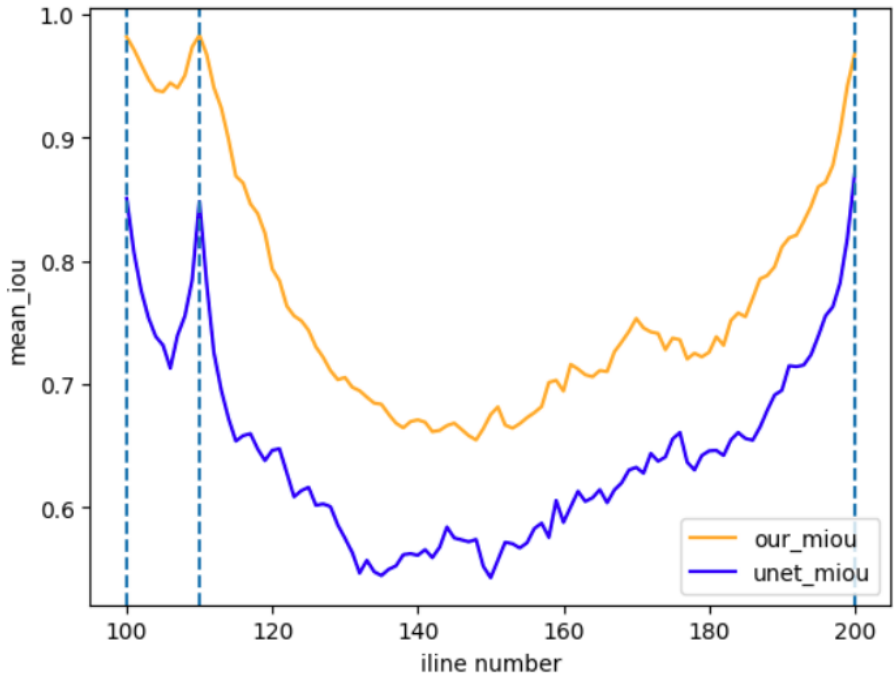


Figure 35: Parihaka mean IoU over slice range

We can observe that our method not only surpasses the U-Net architecture in measured IoU over every individual slice, but also the drop in accuracy in the validation ranges between the training slices (especially in validation range 1) is not as severe.

5.2 F3 Netherlands dataset

To further test our method, we applied our neural network on a second dataset that is named ”F3-Netherlands” which we also mentioned in Chapter 3. As with our first dataset, below we can see a preview of 6 slices and notice how the labels are shaped as we move along the inline axis.

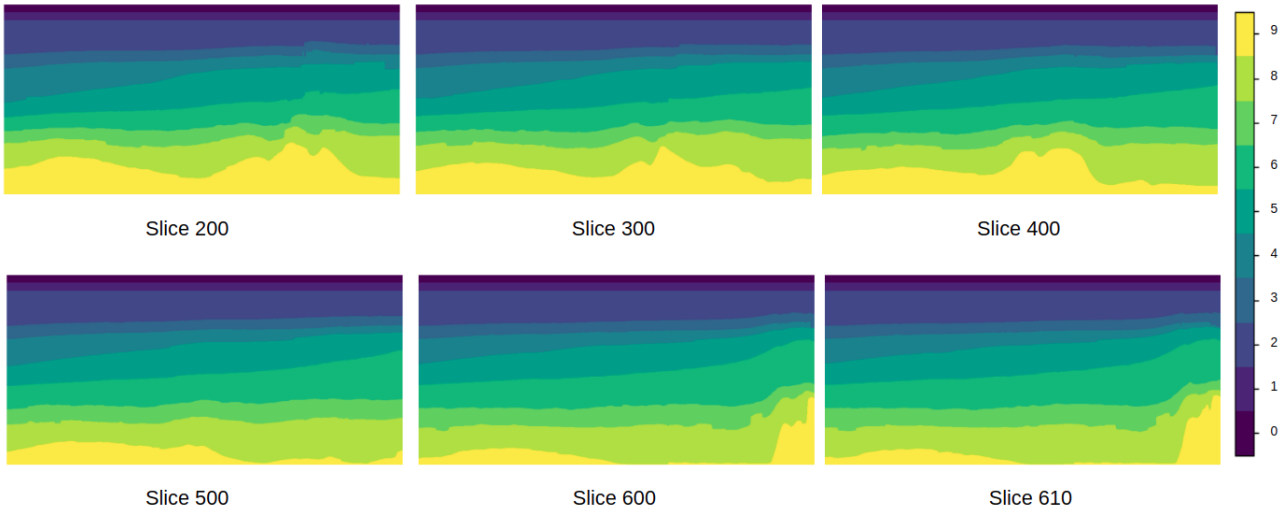


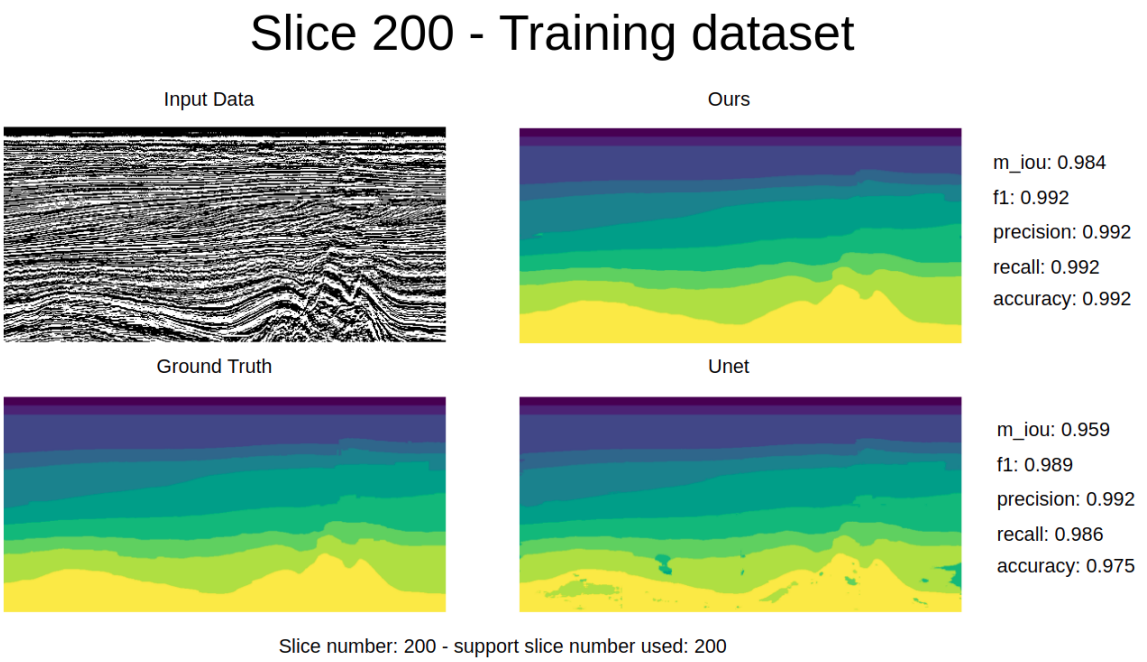
Figure 36: F3-Netherlands dataset

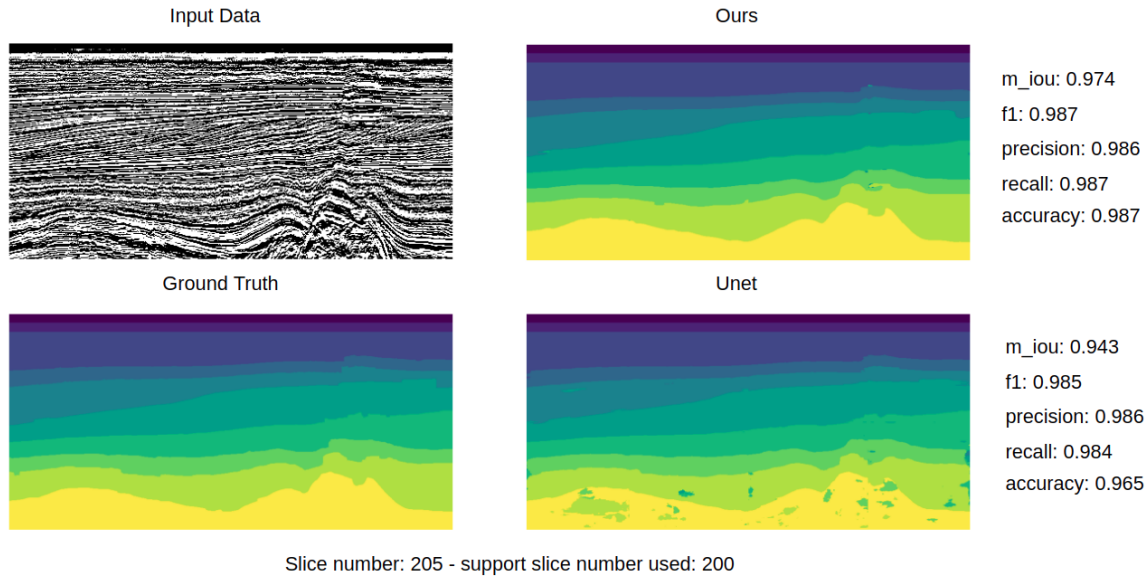
As we can see, this dataset has less distinctly shaped labels and many of them look a lot alike throughout the entirety of the inline range.

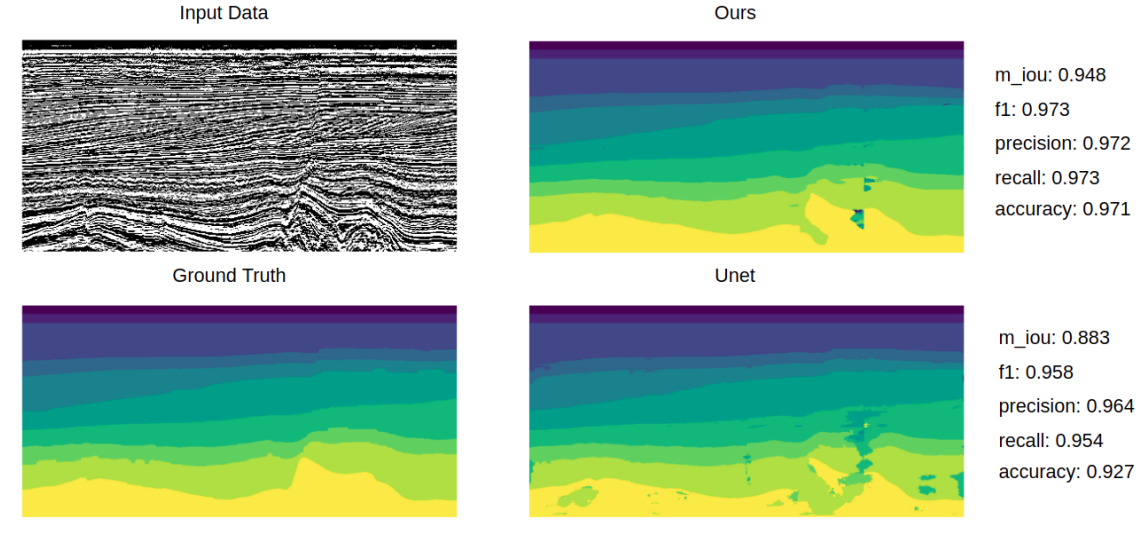
This creates a challenge to our method which uses the one-vs-all technique and requires for every Q_{image} a corresponding set of S_{image}, S_{label}. For example we were able to generate accurate predictions for label ”2” while using as $S_{label} =$ ”7” and vice versa, because they both look a lot alike and in an image patch where one label is present without the other the network gets confused. This problem can be easily tackled though, with a simple step; since we know the nature of the labels, which is that all of them are stacked on top of one another in order then we can safely make the assumption that the ”first” few labels are never present in the bottom of the image and vice versa.

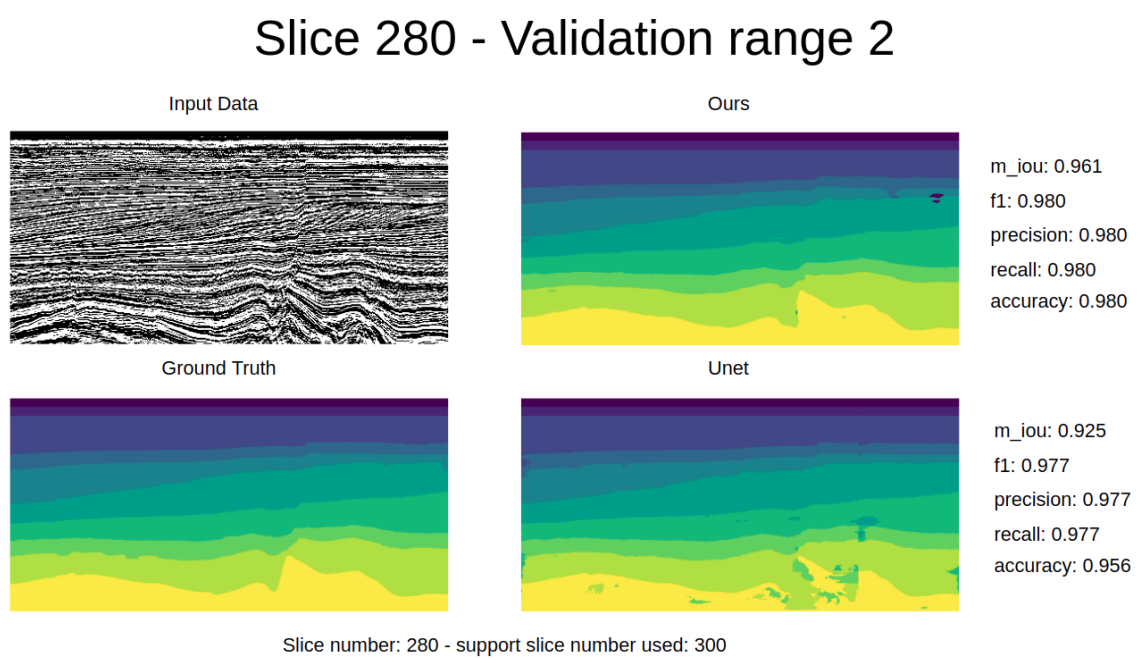
With this assumption in mind, and since with our one-vs-all method we have to predict every patch of every image for all the different unique labels, we simply decide in the top half of the image to predict only for layers 1-6 and for the bottom half only for layers 4-9. So after all, we can claim that our technique also provides an easy post-processing tool to refine the final results, something that is not available with the U-Net architecture.

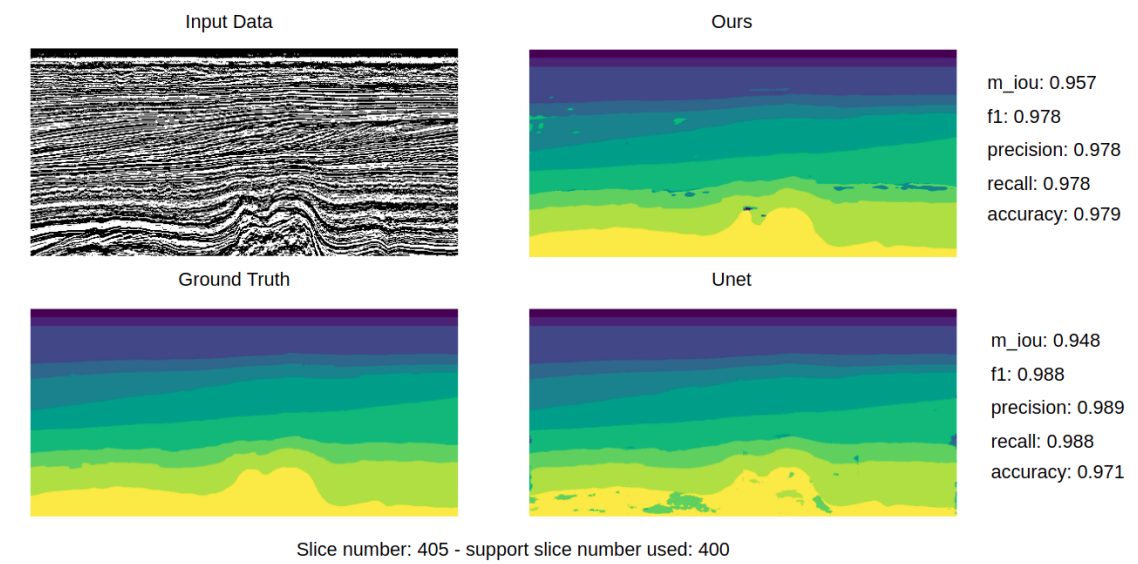
We compare again our final results with the U-Net architecture, both visually and quantitatively in the figures below.

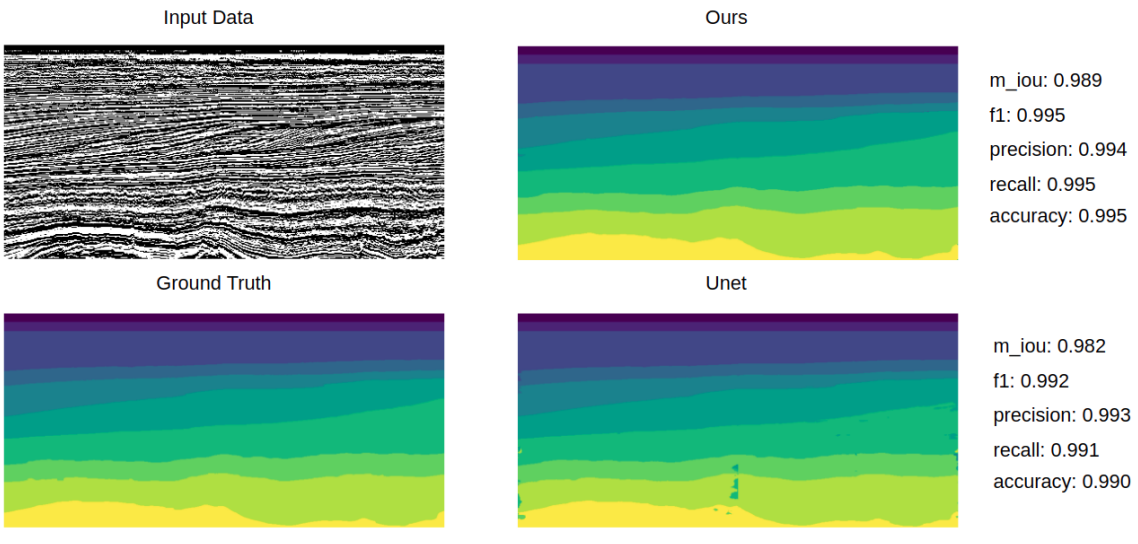

Slice 200 - Training dataset
Input Data
Ours
m_iou: 0.984
f1: 0.992
precision: 0.992
recall: 0.992
accuracy: 0.992
Ground Truth
Unet
m_iou: 0.959
f1: 0.989
precision: 0.992
recall: 0.986
accuracy: 0.975
Slice number: 200 - support slice number used: 200

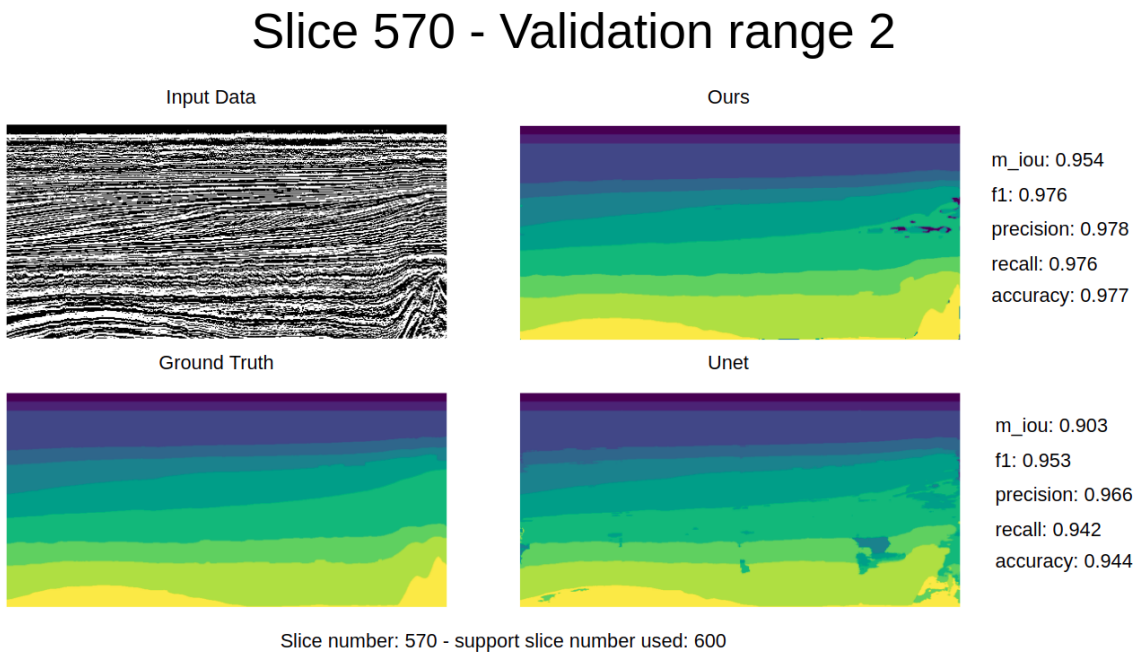

Slice 205 - Validation range 1
Input Data
Ours
m_iou: 0.974
f1: 0.987
precision: 0.986
recall: 0.987
accuracy: 0.987
Ground Truth
Unet
m_iou: 0.943
f1: 0.985
precision: 0.986
recall: 0.984
accuracy: 0.965
Slice number: 205 - support slice number used: 200


Slice 260 - Validation range 2
Input Data
Ours
m_iou: 0.948
f1: 0.973
precision: 0.972
recall: 0.973
accuracy: 0.971
Ground Truth
Unet
m_iou: 0.883
f1: 0.958
precision: 0.964
recall: 0.954
accuracy: 0.927
Slice number: 260 - support slice number used: 210


Slice 280 - Validation range 2
Input Data
Ours
m_iou: 0.961
f1: 0.980
precision: 0.980
recall: 0.980
accuracy: 0.980
Ground Truth
Unet
m_iou: 0.925
f1: 0.977
precision: 0.977
recall: 0.977
accuracy: 0.956
Slice number: 280 - support slice number used: 300


Slice 405 - Validation range 1
Input Data
Ours
m_iou: 0.957
f1: 0.978
precision: 0.978
recall: 0.978
accuracy: 0.979
Ground Truth
Unet
m_iou: 0.948
f1: 0.988
precision: 0.989
recall: 0.988
accuracy: 0.971
Slice number: 405 - support slice number used: 400


Slice 500 - Training dataset
Input Data
Ours
m_iou: 0.989
f1: 0.995
precision: 0.994
recall: 0.995
accuracy: 0.995
Ground Truth
Unet
m_iou: 0.982
f1: 0.992
precision: 0.993
recall: 0.991
accuracy: 0.990
Slice number: 500 - support slice number used: 500



Finally, we plot the mean IoU over the range of 100 inline slices to compare the degradation of performance as we move away from the training data.

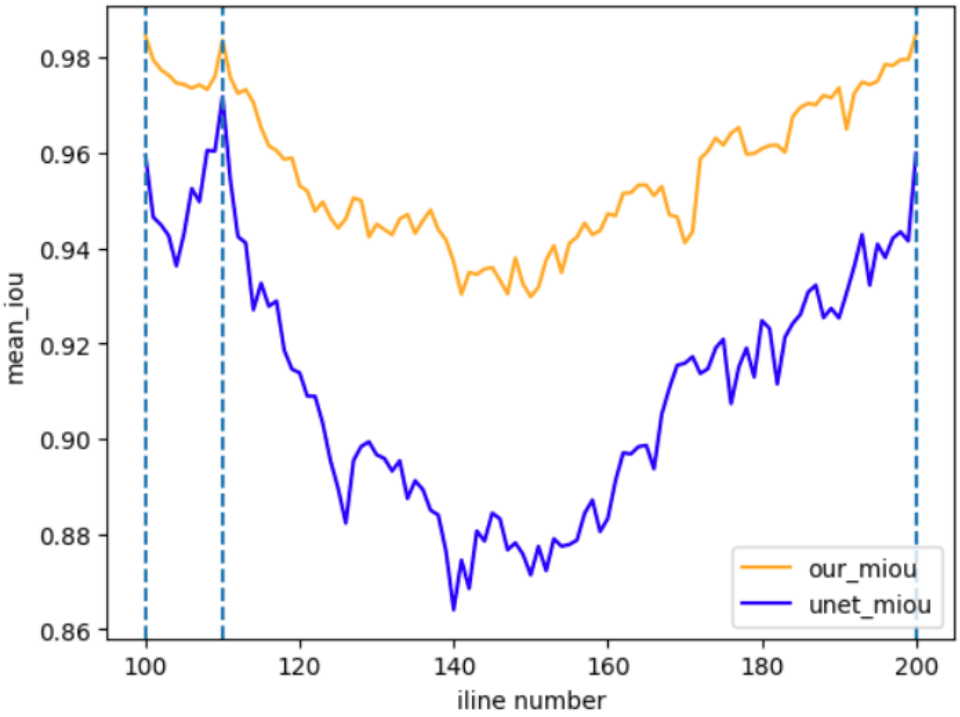


Figure 37: F3-Netherlands mean IoU over slice range

In this plot too we can see that our method not only surpasses the U-Net IoU in every slice, the accuracy drop is way smaller in both validation ranges.

6 Conclusions

This diploma thesis introduces a few-shot semantic segmentation approach to tackle the problem of correctly annotating seismic data with semantic labels that correspond to seismic facies and geobodies in the subsurface.

We relied on an existing architecture and modified it to work with our multi-class problem by converting the network and the data from binary segmentation to one-vs-all segmentation. To do that we changed the way we pre-process the data and evaluate the results and also created a new data generator.

One of the biggest breakthroughs from this implementation, was the idea to replace patch labels with global ones during training and inference. This played one of the most vital roles in the success of our results since it addressed an already existing major problem with image patching and is a very elegant way to ”show” to the network what the final result should look like in its entirety. Before coming up with this idea, the network was severely under performing and in some cases had trouble converging at all during training.

The post processing technique is a simple and fast way to improve our final results. While we were able use it only in the F3-Netherlands dataset it proved to increase the accuracy significantly since it helped the network with the problem of having classes who look alike a lot.

Finally another contributor to our results was the fact that the original architecture we chose already addresses the texture bias that exists in seismic data, and reduces it with the help of the Difference of Gaussians module.

To our knowledge this is the first attempt at using few-shot learning network architectures for this type of datasets and our results indicate that such an approach can produce results that can be considered state of the art and compete with the current preferred industry solutions.

There are various future directions to improve to our current work. One straightforward idea is to implement the same architecture using different k value for k-shot learning, meaning training and inferring with more than 1 labeled image for each class. Another idea is to try and fine-tune both the encoder and also the entire network parameters, and also maybe use an encoder that is pre-trained on other seismic data and not on ImageNet. Another path to follow is to try and convert the network from one-vs-all to multiclass and not one-vs-all because the only downside to our implementation is that the inference time is longer because we predict every image against every label and pick the most probable results. Finally it would be great to see more different few-shot architectures tackling this problem, compare the results.

References

[1] seg wiki, "Seismic facies classification."

[2] J. Jeong, T. S. Yoon, and J. B. Park, "Towards a meaningful 3d map using a 3d lidar and a camera," *Sensors 2018*, 2018.

[3] C. Thomas, "Regression vs. classification in machine learning."

[4] A. Amini, "Introduction to deep learning."

[5] D. Monsters, "7 types of artificial neural networks for natural language processing."

[6] C. Thomas, "An introduction to convolutional neural networks."

[7] B. Hrnjicas, "In depth lstm implementation using cntk on .net platform."

[8] S. Sahoo, "Residual blocks — building blocks of resnet."

[9] K. H. an Xiangyu Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[10] X. Cui, K. Zaheng, L. Gao, and B. Zhang, "Multiscale spatial-spectral convolutional network with image-based framework for hyperspectral imagery classification," *Remote sensing 11*, 2016.

[11] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," *MICCAI 2015*, 2015.

[12] T. E. of Encyclopaedia Britannica, "Seismic survey."

[13] F. C. Oil, "Finding crude oil."

[14] SubSurfWiki, "Gather."

[15] Öz Yilmaz, "Stacking data in cmp gathers."

[16] A. H. A. Latiff, S. Osman, and S. F. Jamaludin, "Seismic data analysis to the converted wave acquisition: A case study in offshore malaysia.," *IOP Conference Series Earth and Environmental Science*, 2016.

[17] R. Azad, A. R. Fayjie, C. Kauffman, I. B. Ayed, M. Pedersoli, and J. Dolz, "On the texture bias for few-shot cnn segmentation," *WACV'21*, 2021.

[18] X. Shi, Z. Chen, H. Wang, D.-Y. Yeung, W. kin Wong, and W. chun Woo, "Convolutional lstm network: A machine learning approach for precipitation nowcasting," *Computer Vision and Pattern Recognition*, 2015.

[19] B. Hrnjicas, "seam-ai-parihaka-seismic-facies-identification-challenge."

[20] W. Commons, "Jaccard index."

[21] O. Vinyals, C. Blundell, T. Lillicrap, K. Kavukcuoglu, and D. Wierstra, "Matching networks for one shot learning.," *MICCAI 2015*, 2016.

[22] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural Networks*, vol. 2, no. 5, pp. 359 – 366, 1989.
DOI: 10.1016/0893-6080(89)90020-8.

[23] K. H. an Xiangyu Zhang, S. Ren, and J. Sun, "Long short-term memory," *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[24] D. Oldenburg and F. Jones, "Migration velocity analysis."

[25] Wikipedia, "Salt dome."

[26] Wikipedia, "Attributes for fluvial channel interpretation."

[27] R. M. Silva, L. Baroni, R. S. Ferreira, D. Civitarese, D. Szwarcman, and E. V. Brazil, "Netherlands dataset: A new public dataset for machine learning in seismic interpretation," 2019.

[28] L. Baroni, R. M. Silva, R. S. Ferreira, D. Civitarese, D. Szwarcman, and E. V. Brazil, "Penobscot dataset: Fostering machine learning development for seismic interpretation," *Geophysics*, 2019.

[29] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *Computer Vision and Pattern Recognition*, 2014.

[30] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, 2004.

[31] X. Li, T. Wei, Y. P. Chen, Y.-W. Tai, and C.-K. Tang, "Fss-1000: A 1000-class dataset for few-shot segmentation," *Computer Vision and Pattern Recognition*, 2019.