Alma Mater Studiorum · University of Bologna

School of Science
Department of Physics and Astronomy
Master Degree in Physics

Volumetric Deep Learning Techniques in Oil & Gas Exploration

Supervisor:
Prof. Armando Bazzani

Submitted by:
Giordano Calvanese

Co-supervisors:
Prof. Renato Campanini
Dr. Matteo Roffilli
Dr. Alfonso Marini

Academic Year 2018/2019

Abstract

This work consisted in the study and application of volumetric Deep Learning (DL) approach to seismic data provided by Eni S.p.A., with an industrial utility perspective. After a series of fruitful meetings with the Upstream & Technical Services team, we clearly defined the final objective of this approach: the automatic search for geological structures such as turbidite channel-bases, as potential regions of interest for the Oil & Gas industry. Therefore, we defined a workflow based on the training of volumetric DL models over seismic horizons containing channel bases providing "windrose" input patches, i.e. a planar approximation of a three-dimensional volume.

All components and sources of criticality were systematically analyzed. For this purpose we studied: the effect of preprocessing, the contribution of the dataset augmentation, the sensitivity for the channel-base manual segmentation, the effect of the spatial expansion of the input patches. Evaluating both qualitatively and quantitatively through K-fold cross-validation.
This work showed: how an appropriate preprocessing of the original data substantially helps DL models, how the dataset augmentation is fundamental for good model generalization given the poor representativity of the accessible examples compared to all possible configurations, how this DL approach is susceptible to the channel-base segmentation imposing to invest sufficient effort in the generation of reliable labels, how the size of input patches must be large enough to allow models to perceive around each voxel the structure concavity and the texture of any sediment infill.

We conclude that the volumetric DL approach developed in this work has proved to be very promising.

Sommario

Questo lavoro è consistito nello studio e applicazione di un approccio Deep Learning (DL) volumetrico a dati sismici di Eni S.p.A., con un ottica di utilità industriale.
Dopo una serie di fruttuosi incontri con il team di Upstream & Technical Services, si è definito in maniera chiara l'obbietivo finale di questo approccio: la ricerca automatica di strutture geologiche quali basi di canali turbiditici, in quanto potenziali zone di interesse per l'industria Oil & Gas. Si è pertanto definito un workflow basato sull'addestramento di modelli DL volumetici su orizzonti sismici contenenti basi di canale attraverso patch di input a "windrose", ossia una approssimazione planare di un volume tridimensionale.

Si sono analizzate in modo sistematico tutte le componenti e le fonti di criticità. A tale scopo si è studiato: l'effetto del preprocessing, il contributo della dataset augmentation, la sensibilità rispetto alla segmentazione manuale della base di canale, l'effetto dell'espansione spaziale delle patch di input. Valutando in modo sia qualitativo che quantitativo tramite K-fold cross-validation.
Questo lavoro ha mostrato: come un appropriato preprocessing del dato originale aiuti in modo sostanziale i modelli DL, come la dataset augmentation sia fondamentale per una buona generalizzazione dei modelli data la scarsa rappresentatività degli esempi accessibili rispetto alle configurazioni possibili, come questo approccio DL sia suscettibile alla segmentazione della base di canale imponendo di dedicare sufficiente impegno nella generazione di label attendibili, come la dimensione delle patch di input debba essere abbastanza estesa da permettere ai modelli di percepire nell'intorno di ogni voxel la concavità delle strutture e la tessitura dell'eventuale infill di sedimenti.

Concludiamo che l'approccio DL volumetrico sviluppato in questo lavoro si è dimostrato essere molto promettente.

Acknowledgements

I would like to thank in the first place those who have given me the opportunity to work to this very stimulating thesis: Prof. Armando Bazzani, Prof. Renato Campanini, Dr. Matteo Roffilli and Dr. Alfonso Iunio Marini.
A particular mention goes to Matteo Roffilli who, with the experience provided and the computational and financial resources of Bioretics srl[1], allowed me to conduct the large-scale experiments in this thesis.

A special thank you goes to my friends and girlfriend: you have made this arduous journey more intense and funny, rejoicing in my victories and giving me the strength to overcome the moments of difficulty.
Thank you very much to my family who has always supported and encouraged me to pursue my ambitions.

[1]`https://www.bioretics.com`

Only when you know the question will you know what the answer means.

Deep Thought
The Hitchhiker's Guide to the Galaxy

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Data mining and analytics have played an important role in knowledge discovery and decision-making in the industry process over the past several decades.

In recent years, many unions or countries have announced a new round of development plans in manufacturing. For example, the European Union proposed 20-20-20 goals to achieve a sustainable future, which means 20% increase in energy efficiency, 20% reduction of CO_2 emissions, and 20% renewables by this year 2020. The US government has proposed a new industrial internet framework for developing the next generation of manufacturing. Similarly, China has announced a new manufacturing plan more recently, which is known as "China Manufacturing 2025", the aim of which is also to make the manufacturing process more intelligent. Those goals can only be realized by incorporating more intelligence into the industrial manufacturing process.

To effectively carry out data mining and analytics in the industry process, machine learning algorithms have always played an important role [27].

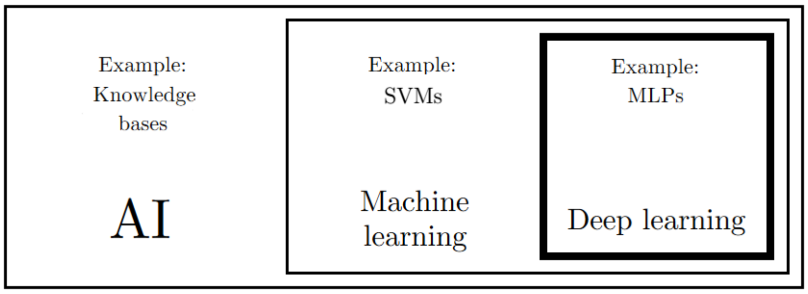


Figure 1.1: AI Eulero-Venn scheme.

1.1 Deep Learning

In this work, we will pay attention to a subset of Machine Learning (ML) called Deep Learning (DL).

DL makes use of Artificial Neural Networks (ANNs), models vaguely inspired by human visual cortex neurons, to create algorithms that can learn how to solve problems from data. From a theoretical point of view, the basis of ANNs can be traced back to the early 1940s with McCulloch's work on neural models. The first hypothesis on the training of these machines was made by Hebb in the late 40s, leading, in the late 50s, to the first true "learning" classifier: Rosenblatt's perceptron [18].

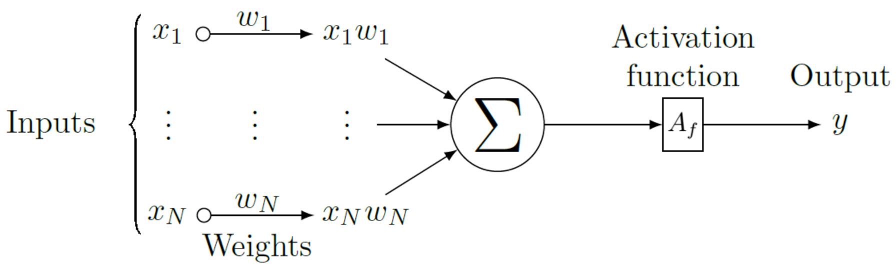


Figure 1.2: Rosenblatt perceptron functioning scheme. A set of real-value inputs x_i are previously multiplied by a weight w_i, like a synaptic weight, and then summed up creating a linear combination of inputs. This linear combination is passed through an activation function A_f that provides the perceptron real-value output y. A single perceptron or a network composed of them belongs to ANNs.

The idea behind the perceptron was simple but effective and it is schematically shown in Figure 1.2 as a biological neuron that receives input from synapses and conditionally propagates an output along the axon. The perceptron has a number of inputs and produces a response as a linear combination of these. The weights of the linear combination determine the relative importance of the input, and by properly adjusting them a simple classifier can be obtained. The right set of weights can be searched for using a feedback system that iteratively compares the correct values with the predicted ones and it adjusts the weights accordingly to their differences.

Perception was a very attractive solution for classification problems, but in 1989 Minsky proved that a single perceptron[1] could not deal with non-linearly separable problems, like the XOR problem, even with activation functions[2] non-linear. This was particularly dramatic because not even by introducing multi-layer architectures it was possible to deal with them, at least with linear activation functions. Stacking multiple layers of

[1]Or equivalently a layer of them, for multiclass classifications.
[2]The activation functions must be monotonous to allow trainability.

perceptrons with linear activation functions is, in fact, equivalent to having only one with appropriate weights. Therefore, the only possible way to solve non-linearly separable problems was to simultaneously introduce multi-layer architectures with non-linear activation functions i.e. Multi-Layer Perceptrons (MPLs). There was simply not enough understanding of the topic to meet this challenge.

The theoretical problems with the perceptron's ability to solve XOR-type problems led to a generalized loss of interest in ANNs by the scientific community in the event known as “First AI winter”. This state lasted until the mid-50s, when the Werbos backpropagation algorithm [24], based on the application of the chain-derived rule, allowed efficient and feasible training for MPLs as well. During the following 15 years, ANNs saw a renewed enthusiasm thanks to the promises of the old models left behind: they were now with a working training method and preliminary results that only increased expectations. Unfortunately, the participation of the scientific community was cold again. Despite the optimistic scenarios supported by technological advances in the field, it became clear that it was only through expensive and mostly ad-hoc hardware that it would be possible to launch the necessary computations that, although relatively simple, were so numerous that they could not be handled by the computers of the time. This drowsed out the remaining interest in ANNs in what is called “Second AI winter”.

Even small networks require an extremely high number of tunable parameters. The only possible solution to manage the training process in acceptable times was a heavy form of parallelization. It was around 2009/2011 that the current definition of DL came into existence. When Andrew Ng[3] began using Nvidia Graphical Processing Units (GPUs) to train his algorithms. GPUs are perfect for parallelizing a vast number of repetitive operations, such as geometric transformations or 3D renderings, and with a little bit of secondary work even those involved by the backpropagation algorithm, taking a fraction of the time it would have taken on the CPU. Moving from CPU to GPU computation was the crucial step in dramatically scaling-up ANNs thanks to the wide availability at a reasonable price of hardware that was prevalently marketed for video games. This combined with the ease of programming provided by new GPU tools such as Nvidia CUDA, led to an unprecedented spread of DL research.

DL, especially recent developments, has changed the way we look at problems and challenges, opening up possibilities that were unimaginable until 10 years ago. DL has seen a massive application in almost all fields of science and industry in a very short period of time, with massive investments of resources and improvements emerging very quickly that span the academic and industrial fields and produce applications ranging from event identification in particle physics to autonomous driving. The reason for this is to be found in the DL's ability to obtain results with relative ease when compared

[3]Andrew Yan-Tak Ng is a Chinese-American computer scientist and statistician, focusing on machine learning and AI. Also, a business executive and investor in Silicon Valley, co-founded and led Google Brain and was a former Vice President and Chief Scientist at Baidu.

with traditional ML methods: where the success of a given strategy is largely influenced by the scientist's experience of manually engineering features for the model. DL makes in many cases this work obsolete thanks to its ability to produce an internal and linearly separable representation of the features space that largely surpasses the handmade ones in efficiency. On the other hand, this implies that these high dimensional representations and their decisional criteria are too complex to be seen and interpreted by a human being. This means that in many cases the DL acts as a "black box" whose internal functioning is not fully understood.

1.1.1 Learning Algorithms

A machine learning algorithm is an algorithm that is able to learn from data. But what do we mean by learning? Mitchell (1997) [14] provides a succinct definition:

"A computer program is said to learn from experience ***E*** *with respect to some class of tasks* ***T*** *and performance measure* ***P****, if its performance at tasks in* ***T****, as measured by* ***P****, improves with experience* ***E****."*

- **Task, T**

  In this relatively formal definition of the word "task", the process of learning itself is not the task. Learning is our means of attaining the ability to perform the task. Machine learning tasks are usually described in terms of how the machine learning system should process an *example*. An example is a collection of N_F features that have been quantitatively measured from some object or event that we want the machine learning system to process. We typically represent an example as a vector $\boldsymbol{x} \in \mathbb{R}^{N_F}$ where each entry x_i of the vector is another feature.

  One of the most common task is *classification*, in which the computer program is asked to specify which of N_C categories some input $\boldsymbol{x}$ belongs to. To solve this task, the learning algorithm is usually asked to produce a function $f : \mathbb{R}^{N_F} \to \{1, 2, \dots, N_\mathrm{C}\}$.
  Where $y = f(\boldsymbol{x})$, the model, through f, assigns an input described by vector $\boldsymbol{x}$ to a category identified by integer code y.

- **Experience, E**

  Machine learning algorithms can be broadly categorized as *unsupervised* or *supervised* by what kind of experience they are allowed to have during the learning process. Learning algorithms are allowed to experience an entire *dataset*. A dataset $\mathbb{D}$ is a collection of N examples[4].

[4]Unlabeled examples $\{\boldsymbol{x}_j \mid j = 1, 2, \dots M\}$ in the case of unsupervised learning and labeled examples $\{(\boldsymbol{x}_j, y_j) \mid j = 1, 2, \dots M\}$ in the case of supervised learning.

Unsupervised learning algorithms experience a dataset containing only features, then learn useful properties of the structure of this dataset.
Supervised learning algorithms experience a dataset containing features and labels.

Roughly speaking, unsupervised learning involves observing several examples of a random vector $\boldsymbol{x}$ and attempting to implicitly or explicitly learn the probability distribution $P(\boldsymbol{x})$, or some interesting properties of that distribution; while supervised learning involves observing several examples of a random vector $\boldsymbol{x}$ and an associated value y, then learning to predict y from $\boldsymbol{x}$, usually by estimating $P(y \mid \boldsymbol{x})$[5].

Unsupervised learning and supervised learning are not formally defined terms. The lines between them are often blurred. Many machine learning technologies can be used to perform both tasks.
For example, the chain rule of probability states that for a vector $\boldsymbol{x} \in \mathbb{R}^{N_F}$, the joint distribution can be decomposed as:

$$P(\boldsymbol{x}) = \prod_{i=1}^{N_\mathrm{F}} P(x_i \mid x_1, x_2, \ldots, x_{i-1}) \tag{1.1}$$

This decomposition means that we can solve the ostensibly unsupervised problem of modelling $P(\boldsymbol{x})$ by splitting it into n supervised learning problems.
Alternatively, we can solve the supervised learning problem of learning $P(y \mid \boldsymbol{x})$ by using traditional unsupervised learning technologies to learn the joint distribution $P(\boldsymbol{x}, y)$, then inferring the conditional probabilities.

$$P(y \mid \boldsymbol{x}) = \frac{P(\boldsymbol{x}, y)}{\sum_{y'} P(\boldsymbol{x}, y')} \tag{1.2}$$

- **Performance, P**

  To evaluate the abilities of a machine learning algorithm, we must design a quantitative measure of its performance. Usually this performance measure **P** is specific to the task **T** being carried out by the system.

  For tasks such as classification we often measure the *accuracy* of the model. Accuracy is just the proportion of examples for which the model produces the correct output. We can also obtain equivalent information by measuring the *error rate*, the proportion of examples for which the model produces an incorrect output.

[5]Conditional probabilities that the correct label is y given the example $\boldsymbol{x}$.

1.1.2 Learning as Mathematical Optimization

Most deep learning algorithms involve *optimization* of some sort. Optimization refers to the task of either minimizing or maximizing some function $J(\boldsymbol{w})$ by altering $\boldsymbol{w}$. We usually state most optimization problems in terms of minimizing[6] $J(\boldsymbol{w})$. The function we want to minimize is called the *loss function*. We denote the values that minimize it, with a superscript *.

$$\boldsymbol{w}^* = \arg\min_{\boldsymbol{w}} [J(\boldsymbol{w})] \tag{1.3}$$

1.1.2.1 Stochastic Gradient Descent

If we want to minimize a real-valued function defined on a multidimensional space $J : \mathbb{R}^N \to \mathbb{R}$, perhaps the simplest method we can use is *gradient descent*. Gradient descent is a first order[7] iterative method which make use of Taylor's expansion:

$$\begin{aligned} \boldsymbol{w}_{t+1} &= \boldsymbol{w}_t - \epsilon \nabla_{\boldsymbol{w}} J_{|\boldsymbol{w}_t} \\ &= \boldsymbol{w}_t - \epsilon \mathbf{g}_t \\ &= \boldsymbol{w}_t + \Delta \boldsymbol{w}_t \end{aligned} \tag{1.4}$$

where ϵ is the *learning rate*, a positive scalar determining the size of the step $\Delta\boldsymbol{w}_t$.

A recurring problem in DL is that large datasets $\mathbb{D}$ are necessary for good results[8], but at the same time large sets are more computationally expensive. In fact DL loss function obviously depends on the dataset $J(\boldsymbol{w};\mathbb{D})$, and often it decomposes as a sum over dataset examples of some per-example loss function J_e.

$$J(\boldsymbol{w};\mathbb{D}) = \sum_{j=1}^{M} J_e\left(\boldsymbol{w};(\boldsymbol{x}_j, y_j)\right) \tag{1.5}$$

Therefore the computational cost of calculating $\mathbf{g}$ is $O(M)$. As the dataset size grows to billions of examples, the time to take a single gradient step become prohibitively long. *Stochastic gradient descent* overcame this issue using this approximation:

$$J(\boldsymbol{w};\mathbb{D}) \simeq J(\boldsymbol{w};\mathbb{B}) \tag{1.6}$$

[6]Maximization may be accomplished via a minimization algorithm by minimizing $-J(\boldsymbol{w})$.

[7]Second order methods can be stated as well, of course involving the Hessian matrix $H(\boldsymbol{w})$. For example Newton's method: $\boldsymbol{w}_{t+1} = \boldsymbol{w}_t - \epsilon H^{-1}(\boldsymbol{w_t})\mathbf{g}_t$. However the computational cost of calculating H^{-1} makes this method impractical in DL

[8]We denote $\hat{P}_{\text{data}}$ as the dataset generating probability distribution function estimated from $\mathbb{D}$ itself and P_{data} as the true one. As M approaches infinity we have: $\hat{P}_{\text{data}} \xrightarrow[M\to\infty]{} P_{\text{data}}$.

In virtue of[9]

$$J(\boldsymbol{w}; \mathbb{D}) = \mathbb{E}_{\mathbb{B} \sim \hat{P}_{\text{data}}} \left[J(\boldsymbol{w}; \mathbb{B}) \right] \tag{1.7}$$

where $\mathbb{B}$ is a randomly sampled *minibatch* of examples $\mathbb{B} = \{(\boldsymbol{x}_{j(B)}, y_{j(B)}) \in \mathbb{D} \mid B = 1, 2, \ldots, M_{\text{B}}\}$ of fixed size $M_{\text{B}} \ll M$.

More sophisticated methods can be defined, for example *ADADELTA* (2012 [26]). This is a per-dimension learning rate method for gradient descent. ADADELTA dynamically adapts over time using only first order information. The method requires no manual tuning of learning rate.

$$\Delta \boldsymbol{w}_{t+1} = -\frac{RMS\left[\Delta \boldsymbol{w}_{t-1}\right]}{RMS\left[\mathbf{g}_t\right]} \mathbf{g}_t \tag{1.8}$$

where

$$\begin{aligned} RMS\left[\mathrm{x}_t\right] &= \sqrt{E_R(\mathrm{x}_t^2) + \varepsilon} & \varepsilon > 0 \\ E_R(\mathrm{x}_{t+1}) &= \rho E_R(\mathrm{x}_t) + (1-\rho)\mathrm{x}_t & \rho \in (0,1) \end{aligned} \tag{1.9}$$

E_R is a pseudo running average over a time window. Window size is implicitly specified by ρ, asymptotically infinite-sized as ρ approaches 1. RMS is a pseudo root mean square whit a constant positive scalar ε for better conditioning of the denominator in Equation 1.8.

1.1.2.2 Loss Function and Activation Function

The function that we want to minimize is the loss function. Fortunately, the loss functions for neural networks are more or less the same as those for other *parametric models*[10]. DL model, in classification tasks, defines a conditionally probability distribution[11] $P_{\text{model}}\left(\boldsymbol{y} \mid \boldsymbol{x}; \boldsymbol{w}\right)$ and are trained using *maximum likelihood*.

$$\boldsymbol{w}^* = \arg\min_{\boldsymbol{w}} \left[-P_{\text{model}}(\mathbb{D}; \boldsymbol{w})\right] \tag{1.10}$$

[9] $\mathbb{E}_{\mathrm{x} \sim P(\mathrm{x})}\left[\mathrm{x}\right]$ is the expectation value of a random variable x drawn from a probability distribution $P(\mathrm{x})$.

[10] DL models are parametric respect to weights $\boldsymbol{w}$.

[11] Now we denote labels in vectorial form using the one-hot encoding. In other words labels are expressed as versors of a N_{C}-dimensional space.

In the hypothesis that examples in $\mathbb{D}$ are independently drawn from the true but unknown data generating distribution P_{data}, we have:

$$\begin{aligned} \boldsymbol{w}^* &= \underset{\boldsymbol{w}}{\arg\min} \left[-\prod_{j=1}^{M} P_{\text{model}} \left(\boldsymbol{y}_j \mid \boldsymbol{x_j}; \boldsymbol{w} \right) \right] \\ &= \underset{\boldsymbol{w}}{\arg\min} \left[-\sum_{j=1}^{M} \ln\big(P_{\text{model}} \left(\boldsymbol{y}_j \mid \boldsymbol{x}_j; \boldsymbol{w} \right) \big) \right] \\ &= \underset{\boldsymbol{w}}{\arg\min} \left[-\mathbb{E}_{(\boldsymbol{x},\boldsymbol{y})\sim\hat{P}_{\text{data}}} \ln\big(P_{\text{model}} \left(\boldsymbol{y} \mid \boldsymbol{x}; \boldsymbol{w} \right) \big) \right] \end{aligned} \tag{1.11}$$

Training using maximum likelihood means that the loss function is simply the negative log-likelihood, equivalently described as the *cross-entropy* between the dataset and the model probability distribution.

$$J(\boldsymbol{w}) = -\mathbb{E}_{(\boldsymbol{x},\boldsymbol{y})\sim\hat{P}_{\text{data}}} \ln\big(P_{\text{model}} \left(\boldsymbol{y} \mid \boldsymbol{x}; \boldsymbol{w} \right) \big) \tag{1.12}$$

The choice of loss function is tightly coupled with the choice of the activation function A_f of *output unit*. The choice of how to represent them determines the form of the cross-entropy function. Any kind of activation function that may be used for output units can also be used for hidden unit as well and vice versa; but any time we wish to represent a probability distribution over a N_{C} discrete classes, we may use the *softmax* function. Formally the softmax function is given by[12]

$$\text{softmax}(\boldsymbol{z})_i = \frac{\exp(z_i)}{\sum_{k=1}^{N_{\text{c}}} \exp(z_k)} \tag{1.13}$$

Therefore if

$$\begin{aligned} P_{\text{model}}(\boldsymbol{y}_i = \hat{\boldsymbol{c}} \mid \boldsymbol{x}_i; \boldsymbol{w}) &= P_{\text{model}}\big(\boldsymbol{y}_i = \hat{\boldsymbol{c}}; \boldsymbol{z}(\boldsymbol{x}_i, \boldsymbol{w})\big) \\ &= \text{softmax}\big(\boldsymbol{z}(\boldsymbol{x}_i, \boldsymbol{w})\big)_c \end{aligned} \tag{1.14}$$

then minimizing the cross-entropy implies the minimization of:

$$\begin{aligned} -\ln\big(\text{softmax}(\boldsymbol{z})_c\big) &= -z_c + \ln\left(\sum_{k=1}^{N_{\text{c}}} \exp(z_k) \right) \\ &\simeq -z_c + \max_k\{z_k\} \end{aligned} \tag{1.15}$$

Minimizing Equation 1.15 encourages the first term z_c, relative to the true class, to be pushed up, while the second term encourages all of $\boldsymbol{z}$ to be pushed down.

[12] We refer to z as the inner summation of inputs inside a perceptron, see Figure 1.2.

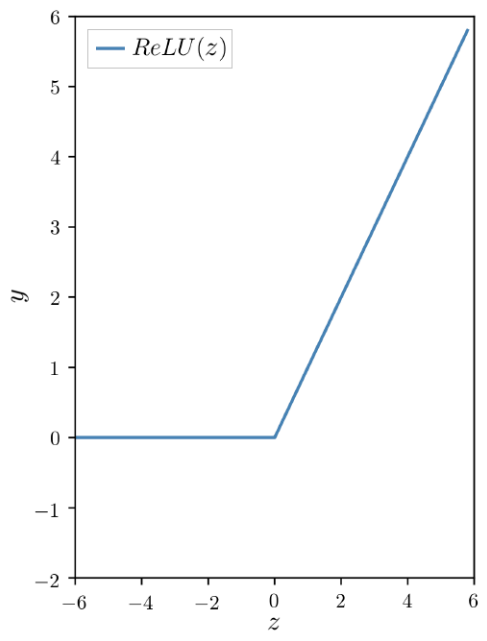


Figure 1.3: Graphical representation of ReLU function.

The design of *hidden unit* is an extremely active area of research and not yet have many definite guiding theoretical principles. *Rectified linear units* are an excellent default choice of hidden units. Them use the activation function $A_f(z) = \mathrm{ReLU}(z)$

$$\mathrm{ReLU}(z) = \max\{0, z\} \tag{1.16}$$

These units are easy to optimize because they are so similar to liner units. The only difference is that a rectified linear unit outputs zero across half of its domain. This makes the derivatives through a rectified linear unit remain large whenever the unit is active. Its simplicity is useful in the learning phase because the information taken by the gradient is more effective than it would be with activation function that introduce second-order effects. One may point out that the non-differentiability of *ReLU* in $z = 0$ could be an invalidate characteristic when used with a gradient-based learning algorithm. In practice, however, gradient descent still performs well enough because we are usually not interested in arriving to a local minima of the loss function; but merely to a significantly low value of it. Because we do not expect training to actually reach critical points[13], it is acceptable for the minima of the loss function to correspond to points with undefined gradient and therefore not accessible through gradient descent.

[13] Critical points of a function $J(\boldsymbol{w})$ are points $\boldsymbol{w}^*$ where $\nabla J_{|w^*} = \mathbf{0}$.

1.1.2.3 Capacity, Underfitting and Overfitting

The central challenge in machine learning is that our algorithm must perform well on new, previously unseen inputs not just those on which our model was trained. The ability to perform well on previously unobserved inputs is called *generalization*. Typically, when training a machine learning model, we have access to a *training set*; we can compute some error measure on the training set, called the training error; and training process reduces this training error.

So far, what we have described is simply an optimization problem. What separates machine learning from optimization is that we want the generalization error, also called the *test error*, to be low as well.
We typically estimate the test error of a machine learning model by measuring its performance on a *test set* $\mathbb{T}_e$ of examples that were collected separately from the training set $\mathbb{T}_r$[14].

$$\mathbb{D} = \mathbb{T}_r \cup \mathbb{T}_e \tag{1.17}$$

In order to theoretical justify the generalization ability of the model, it has to be that the training and test data are generated by a shared probability distribution called the dataset generating probability distribution P_{data}. We also make a set of assumptions known collectively as the "i.i.d. assumptions". These assumptions are that the examples in each dataset are *independent* from each other, and that the training set and test set are *identically distributed*. These assumptions enable us to mathematically study the relationship between training error and test error. One immediate connection we can observe between training error and test error is that the expected training error of a randomly selected model is equal to the expected test error of that model. Of course, when we use a machine learning algorithm, we do not fix the parameters ahead of time, then sample both datasets. We sample the training set, then use it to choose the parameters to reduce training error, then we evaluate test error. Under this process, the expected test error is greater than or equal to the expected value of training error.

The factors determining how well a machine learning algorithm will perform are its ability to:

- Make the training error small.
- Make the gap between training and test error small.

These two factors correspond to the two central challenges in machine learning: *underfitting* and *overfitting*.

[14]A common practice is to, actually, monitoring the training phase whit a third set called *validation set* $\mathbb{T}_v$, which is obtained from $\mathbb{T}_r = \mathbb{T}_x \cup \mathbb{T}_v$. Effective training set is then $\mathbb{T}_x$ because validation set is not presented to the model; it's instead used to tuning *hyperparameters*.

Underfitting occurs when the model is not able to obtain a sufficiently low error value on the training set. Overfitting occurs when the gap between the training error and test error is too large.

We can control whether a model is more likely to overfit or underfit by altering its *capacity*. Informally, a model's capacity is its ability to fit a wide variety of functions which is indeed strongly related to the number of its trainable parameter. Models with low capacity may struggle to fit the training set. Models with high capacity can overfit by memorizing properties of the training set that do not serve them well on the test set. There are many ways to change a model's capacity. Capacity is not determined only by the choice of model. The model specifies which family of functions the learning algorithm can choose from when varying the parameters in order to reduce a training objective. This is called the *representational capacity* of the model. In many cases, finding the best function within this family is a difficult optimization problem. In practice, the learning algorithm does not actually find the best function, but merely one that significantly reduces the training error. These additional limitations, such as the imperfection of the optimization algorithm, mean that the learning algorithm's *effective capacity* may be less than the representational capacity of the model family.

Perhaps the most important results in statistical learning theory shows that the discrepancy between training error and test error is bounded from above by a quantity that grows as the model capacity grows but shrinks as the number of training examples increases (Vapnik and Chervonenkis, 1971 [23]).

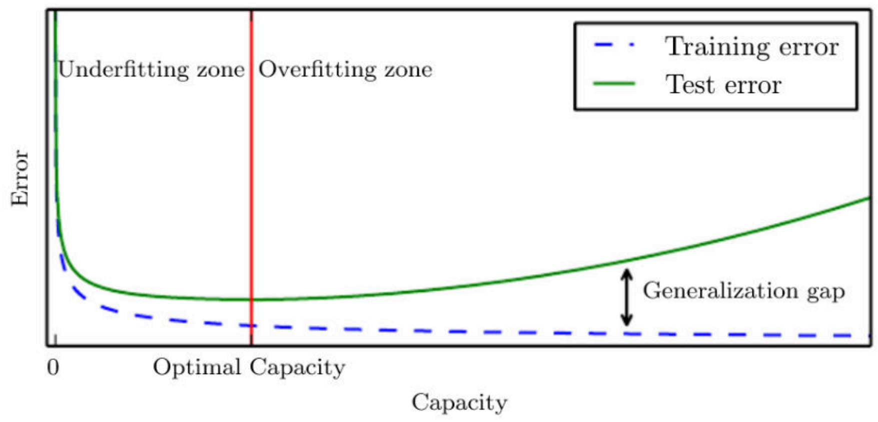


Figure 1.4: Typical relationship between error and capacity [6]. Typically, training error decreases until it asymptotes to the minimum possible error value as model capacity increases. Typically test error has a U-shaped curve as a function of model capacity.

These bounds provide intellectual justification that machine learning algorithms can work, but they are rarely used in practice because it can be quite difficult to determine the effective capacity of deep learning algorithms. We must simply remember that while simpler functions are more likely to generalize, we must still choose a sufficiently complex hypothesis to achieve low training error.

One may suggest in order to enhance capacity and parallel have a better chance to have separable classes, to increase the number of example's features N_F. However, this is a double-edged sword commonly known as the *curse of dimensionality*. In fact, we have:

$$\lim_{N_F \to \infty} \left|\left|\hat{P}_{\text{data}} - P_{\text{data}}\right|\right| = \infty \tag{1.18}$$

This is due to the fact that the number of possible configurations of every example $\boldsymbol{x}$ grows exponentially as the number of features increase, and therefore the examples are diluted in this high-dimensional feature space. An increase in N_F then requires an exponential increase in M to compensate for this dilution and to allow the limit in Equation 1.18 to converge.

1.1.2.4 Regularization Techniques

Regularization is any modification we make to a learning algorithm that is intended to reduce its test error but not its training error. Regularization is one of the central concerns of the field of machine learning, rivaled in its importance only by optimization.

- **Early Stopping**

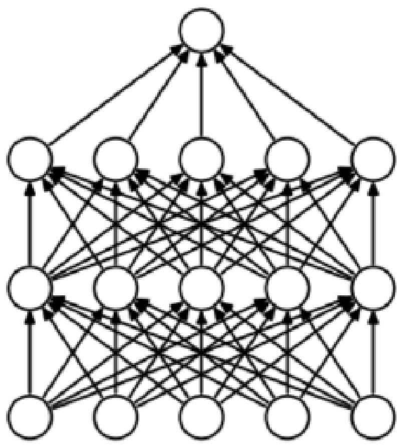
  When training large models with sufficient representational capacity to overfit the task, we often observe that training error decreases steadily over time, but validation set error begins to rise again. See Figure 1.4 for an example of this behavior, which occurs reliably. This means we can obtain a model with better validation set error (and thus, hopefully better test set error) by returning to the parameter setting at the point in time with the lowest validation set error. Every time the error on the validation set improves, we store a copy of the model parameters. When the training algorithm terminates, we return these parameters, rather than the latest parameters.

  This strategy is known as *early stopping*. It is probably the most commonly used form of regularization in deep learning.

- **Dropout**

  Dropout (Srivastava et al., 2014 [21]) provides a computationally inexpensive but powerful method of regularizing a broad family of models. Specifically, dropout trains the ensemble consisting of all subnetworks that can be formed by removing a certain percentage P_{Dropout} of non-output units from an underlying base network, as illustrated in Figure 1.5.

Dropout provides therefore an inexpensive approximation to training and evaluating a wrapped ensemble of exponentially many neural networks, increasing model's generalization capabilities.


(a) Multi Layer Perceptron.

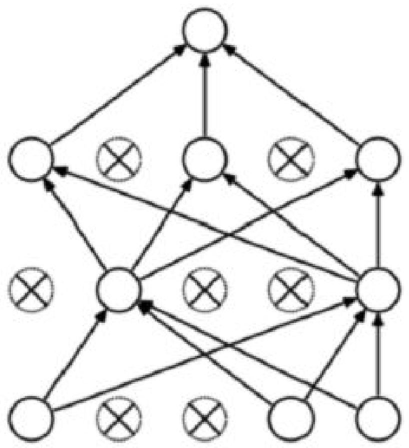

(b) Multi Layer Perceptron after applying dropout.

Figure 1.5: Dropout [21].

- **Dataset Augmentation**

  The best way to make a machine learning model generalize better is to train it on more data. Of course, in practice, the amount of data we have is limited. One way to get around this problem is to create fake data and add it to the training set. For some machine learning tasks, it is reasonably straightforward to create new fake data.

  Dataset augmentation has been a particularly effective technique for a specific classification problem: object recognition. Images are high dimensional and include an enormous range of factors of variation, many of which can be easily simulated. Operations like translating the training images a few pixels in each direction can often greatly improve generalization. Many other operations, such as rotating the image or scaling the image, have also proved quite effective

1.1.3 Convolutional Networks

Convolutional networks, also known as Convolutional Neural Networks (CNNs), are a specialized kind of neural network for processing data that has a known grid-like topology. Examples include time-series data, which can be thought of as a 1D grid taking samples at regular time intervals, and image data, which can be thought of as a 2D grid of pixels. Convolutional networks have been tremendously successful in practical applications.

"Convolutional neural network" indicates that the network employs a mathematical operation called *convolution*[15]. Convolution is a specialized kind of linear operation. Convolutional networks are simply neural networks that use convolution in place of general matrix multiplication in at least one of their layers.

1.1.3.1 Convolution

Convolution operation, in two dimensions (i_1, i_2), is defined as:

$$\begin{aligned} \boldsymbol{S} &= \boldsymbol{x} * \boldsymbol{w} \\ S(m,n) &= \sum_{i_1}\sum_{i_2} x(i_1,i_2)w(m-i_1,n-i_2) \end{aligned} \tag{1.19}$$

In convolutional network terminology, the first argument $\boldsymbol{x}$ to the convolution is often referred to as the input, and the second argument $\boldsymbol{w}$ as the kernel. The output $\boldsymbol{S}$ is sometimes referred to as the feature map.

It can be shown that convolution is commutative. The commutative property of convolution arises because we have flipped the kernel relative to the input. The only reason to flip the kernel is to obtain the commutative property. Even if the commutative property is useful for writing proofs, it is not usually an important property of a neural network implementation. Instead, many neural network libraries implement a related function called the *correlation*, which is the same as convolution but without flipping the kernel.

$$\begin{aligned} \boldsymbol{S} &= \boldsymbol{x} \star \boldsymbol{w} \\ S(m,n) &= \sum_{i_1}\sum_{i_2} x(i_1,i_2)w(m+i_1,n+i_2) \end{aligned} \tag{1.20}$$

1.1.3.2 Parameter Sharing

Convolution leverages two important ideas that can help improve a machine learning system: *parameter sharing*[16], *sparse interactions.*

[15] Actually it's not convolution, it's instead correlation. Many machine learning libraries implement correlation but call it convolution.

[16] In the case of convolutional networks this property is tightly related to the concept of *equivariant representations* or specifically equivariance to translation. To say a function is equivariant means that if the input changes, the output changes in the same way. Similarly with images, convolution creates a 2-D map of where certain features appear in the input. If we move the object in the input, its representation will move the same amount in the output.

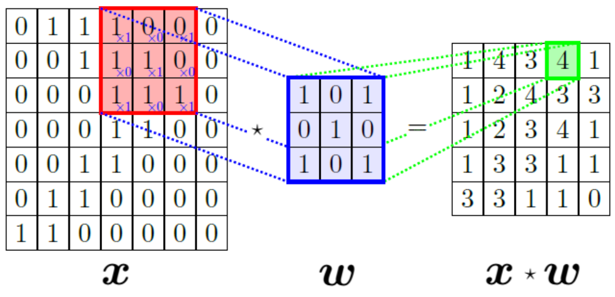


Figure 1.6: 2D correlation operation. We notice that, without padding of input, feature map is smaller than input and shrinking is related to the kernel size.

Parameter sharing refers to using the same parameter for more than one function in a model. Discrete convolution can be viewed as multiplication by a matrix, but the matrix has several entries constrained to be equal to other entries. In addition to these constraints that several elements be equal to each other, convolution usually corresponds to a very sparse matrix[17].

1.1.3.3 Pooling

A pooling function replaces the output of the net at a certain location with a summary statistic of the nearby outputs. For example, the *max pooling* operation reports the maximum output within a rectangular neighborhood.

In all cases, pooling helps to make the representation approximately invariant to small translations of the input. Invariance to translation means that if we translate the input by a small amount, the values of most of the pooled outputs do not change. Invariance to local translation can be a useful property if we care more about whether some feature is present than exactly where it is.

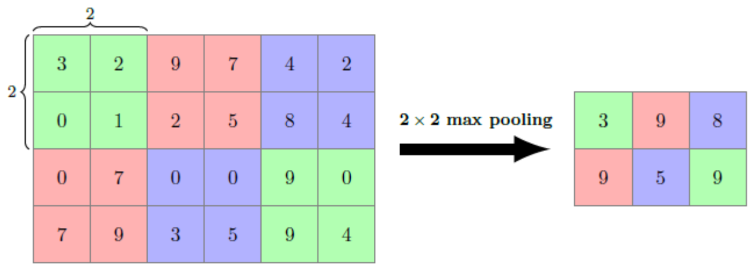


Figure 1.7: 2D max pooling operation.

[17]A sparse matrix is a matrix whose entries are mostly equal to zero. This is because the kernel is usually much smaller than the input image.

1.2 Hydrocarbon Exploration

Exploration and production of hydrocarbons (HC) is a high-risk venture.
These uncertainties originated from geological models and coupled with economic and engineering models involve high-risk decision scenarios, with no absolute guarantee of successfully discovering and developing hydrocarbons.

The future trends in oil resources availability will depend largely on the balance between the outcome of the cost-increasing effects of depletion and the cost-reducing effects of the new technology. Technological advances allowed the exploration in well established basins as well as in new frontier zones such as ultra-deep waters.

Information is vital for decision-making. Therefore, it's necessary to define the value of information associated with important decisions. Information only has value in a decision problem if it results in a change in some action to be taken by a decision maker. The information is seldom perfectly reliable and generally it does not eliminate uncertainty, so the value of information depends on both the amount of uncertainty, or equivalently the prior knowledge available, and payoffs involved in the petroleum exploration and production projects.

Over the last two decades, the advances in computer-aided decision making processes have provided a mechanism to improve the quality of decision making in modern petroleum industry.
However, as Newendorp [22] emphasized, the decision analysis does not eliminate or reduce risk and will not fully replace professional judgment of geoscientists, engineers, and managers.

1.2.1 Components of a Prospect

A *prospect* is a potential trap which geologists believe may contain hydrocarbons. A significant amount of geological, structural and seismic investigation must first be completed to redefine the potential hydrocarbon drill location from a *lead* to a prospect. Four geological factors have to be simultaneously present for a prospect to work and if any of them fail neither oil nor gas will be present.

- **Source Rock**

  In petroleum geology, *source rock* refers to rocks from which hydrocarbons have been generated or are capable of being generated. They form one of the necessary elements of a working petroleum system. They are organic-rich sediments that may have been deposited in a variety of environments including deep water marine, lacustrine and deltaic.

  A river delta is a landform created by deposition of sediment that is carried by a river as the flow leaves its mouth and enters slower-moving or stagnant water. This occurs where a river enters an ocean, sea, estuary, lake, or (more rarely) another river that cannot carry away the supplied sediment.

During *sedimentary diagenesis*[18] the degradation of living matter eventually trapped in sediments begins. The original organic matter could comprise lacustrine and marine algae and plankton and terrestrial higher-order plants. During diagenesis large biopolymers e.g. proteins and carbohydrates in the original organic matter decompose partially or completely. These resulting units can then polycondense to form geopolymers. The formation of geopolymers in this way accounts for the large molecular weights and diverse chemical compositions associated with *kerogen*[19].

Resulting changes in the burial temperatures and pressures lead to further changes in kerogen composition including loss of hydrogen, oxygen, nitrogen, sulfur, and their associated functional groups, and subsequent isomerization and aromatization. Such changes are indicative of the thermal maturity state of kerogen.

During the process of thermal maturation, kerogen breaks down in high-temperature pyrolysis reactions to form lower molecular weight products including bitumen, oil, and gas. The extent of thermal maturation controls the nature of the product, with lower thermal maturities yielding mainly bitumen/oil and higher thermal maturities yielding gas. These generated species are partially expelled from the kerogen-rich source rock and in some cases can charge into a reservoir rock.

- **Migration**

  Migration is the movement of hydrocarbons from their source into reservoir rocks.

  Migration typically occurs from a structurally low area to a higher area because of the relative buoyancy of hydrocarbons in comparison to the surrounding rock. Migration can be local or can occur along distances of hundreds of kilometers in large sedimentary basins, and is critical to the formation of a viable petroleum system. The hydrocarbons are expelled from source rock, moving by density-related mechanisms. Most hydrocarbons could even migrate till the surface as oil seeps, but some will get trapped.

- **Reservoir**

  An oil and gas *reservoir* is a subsurface pool of hydrocarbons contained in porous or fractured rock formations. Oil and gas reservoirs are broadly classified as conventional and unconventional reservoirs. In case of conventional reservoirs, the

[18]After deposition, sediments are compacted as they are buried beneath successive layers of sediment and cemented by minerals that precipitate from solution.

[19]Kerogen is a mixture of organic chemical compounds that make up a portion of organic matter in sedimentary rocks. It is insoluble in normal organic solvents due to the enormous molecular weight of the constituent compounds. The soluble portion is known as bitumen.

naturally occurring hydrocarbons, such as crude oil or natural gas, are capped by overlying rock formations (the seal) with lower permeability. While in unconventional reservoirs the rocks have high total porosity and very low permeability which keeps the hydrocarbons trapped in place not allowing migration, therefore not requiring a cap rock. Reservoirs are found using hydrocarbon exploration methods.

- **Trap**

  The hydrocarbons are buoyant respect to the higher density water usually trapped in the sediments (formation water), hence the need to have a trap configuration, limiting the buoyancy. The hydrocarbon *trap* has to be covered by an impermeable rock known as a seal or cap-rock in order to prevent hydrocarbons escaping to the surface. A trap forms when the buoyancy forces driving the upward migration of hydrocarbons through a permeable rock cannot overcome the capillary forces of a sealing medium. The timing of trap formation relative to that of petroleum generation and migration is crucial to ensuring a reservoir can form. All the trap elements have to be correctly timed in order to co-occur.

1.2.2 Seismic Survey

Visible surface features such as oil seeps, natural gas seeps, pockmarks[20] provide basic evidence of hydrocarbon generation. However, most exploration depends on highly sophisticated technology to detect and determine the extent of these deposits using exploration geophysics. Areas thought to contain hydrocarbons are initially subjected to a gravity survey, magnetic survey, passive seismic or regional seismic reflection surveys to detect large-scale features of the sub-surface geology. Once defined the features of interest, known as leads, these are subjected to more detailed *seismic surveys*. The seismic surveys principle and models make use of the relationships that exist between the propagation of acoustic and/or elastic waves, including, reflections and refractions, according to the kind of material (rock type) and its filling fluid (water, gas or liquid HC, other fluids), and the physical (mechanical) properties of the matter; allowing to obtain clearer images of the underlying geological structure. Seismic data have to be interpreted in order to identify all the prospect elements, and trap geometries.

Finally, when a prospect has been identified and evaluated and passes the oil company's selection criteria, an exploration well could be drilled in an attempt to conclusively determine the presence or absence of hydrocarbons.

1.2.2.1 More Details on Seismic Data Acquisition

Seismic data acquisition involves applying a seismic energy source generating a propagating pulse. This source is dependent on the area where survey has been designed:

[20]Underwater craters caused by escaping gas

onshore, offshore, intermediate very shallow water areas. Vibroseis truck, dynamite shot, or an air gun, generates acoustic, or better elastic waves that travel into the Earth in different modes. The most used mode that is processed to obtain seismic images is the compressional. Waves pass through strata with different seismic responses and earth filtering effects (alterating the initial pulse shape, that is non-stationary along is path), and return back to the surface to be recorded as seismic data by geophones or seismometers.

Seismic data acquisition involves applying a seismic energy source. This source such as a vibroseis truck, dynamite shot, or an air gun, generates acoustic or elastic vibrations that travel into the Earth, pass through strata with different seismic responses and filtering effects, and return to the surface to be recorded as seismic data by geophones or seismometers.

The study area of *marine survey acquisition*, in particular, is considered to be a "deep-water area" with a column of water that reaches 500 m or much more. Accordingly the seismic is acquired with particular techniques, see Figure 1.8.

1.2.2.2 Elements of Seismic Data Processing

Seismic processing consists of several operation steps on the acquired or "raw" seismic data, to suppress noise, enhance signal and migrate seismic events to its appropriate location in space. Seismic processing facilitates better interpretation because subsurface structures and reflection geometries are more apparent. There are three main processes in seismic data processing: deconvolution, Common-MidPoint (CMP) stacking and migration.

- **Deconvolution**

  Deconvolution is a process that tries to extract the reflectivity series of the Earth, under the assumption that a seismic trace is just the reflectivity series of the Earth convolved with distorting filters. This process improves temporal resolution by collapsing the seismic wavelet lenght, but it is non-unique unless further information is available such as well logs, or further assumptions are made. Deconvolution operations can be cascaded, with each individual deconvolution designed to remove a particular type of distortion.

- **CMP Stacking**

  CMP stacking is a robust process that uses the fact that a particular location in the subsurface have been sampled numerous times and at different offsets. This allows a geophysicist to construct a group of traces with a range of offsets that all samples the same subsurface location,known as a Common Midpoint Gather. Another process that is applied to proceed to CMP stack is the Normal MoveOut (NMO), see Figure 1.9. The moveout quantity is dependent from the propagation velocity of the rock to pressure waves. NMO align horizontally all the seismic

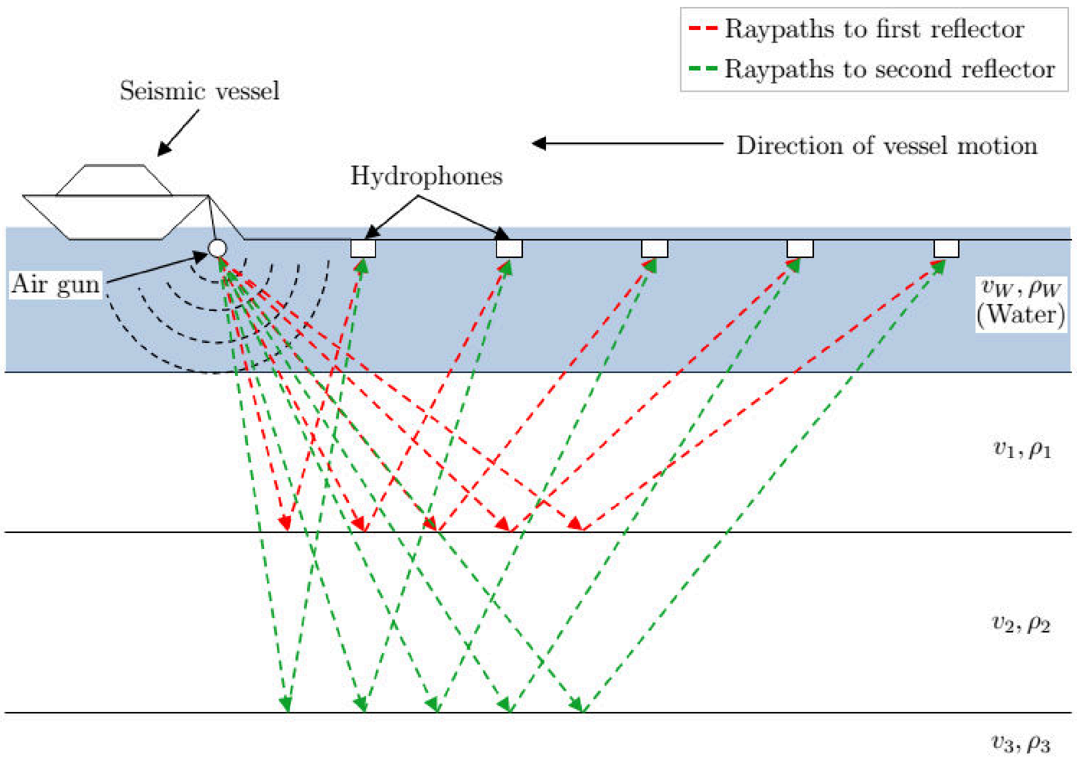
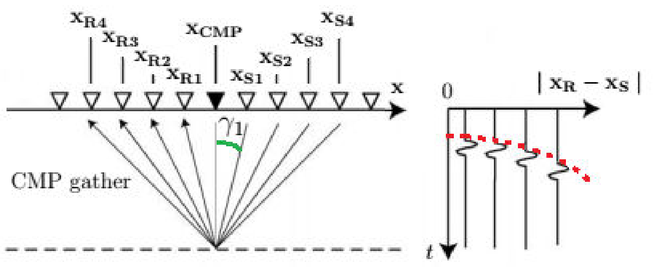
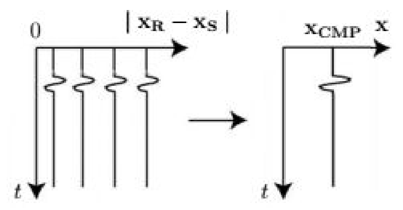


Figure 1.8: Marine seismic acquisition. Traditional marine seismic surveys are conducted using specially-equipped vessels that tow one or more cables containing a series of hydrophones at constant intervals. The cables are known as streamers, with 2D surveys using only 1 streamer and 3D surveys employing up to 12 or more. The streamers are deployed just beneath the surface of the water and are at a set distance away from the vessel. The seismic source, usually an airgun or an array of airguns but other sources are available, is also deployed beneath the water surface and is located between the vessel and the first receiver. Marine seismic surveys generates a significant quantity of data, in fact each streamer can be up to 6 or even 8 km long and the seismic source is typically fired every 15 or 20 seconds.

event, that are curved along hyperbola, according to their propagation law, with respect to the offset. Better stack along constant event times can be therefore performed. The average amplitude is calculated along time samples, resulting in significantly lowering the random noise but also losing all valuable information about the relationship between seismic amplitude and offset (information on elastic properties of the rocks).



(a) CMP gather. The same event in a CMP gather has a hyperbolic time location respect to offset. γ_1 is the particular angle of reflection for raypath 1.



(b) NMO correction. Traces are re-located in time to account offset, and then averaged out to increase signal-to-noise ratio.

Figure 1.9: CMP stacking.

- **Seismic Migration**

  Seismic migration is the process by which seismic events are geometrically relocated in either space or time to the location the event occurred in the subsurface rather than the location that it was recorded at the surface. Creating, thereby, a more accurate image of the subsurface, see Figure 1.10. Migration precision is function of appropriate algorithms and of the knowledge of acoustic and elastic properties, first of all the pressure waves velocity model.

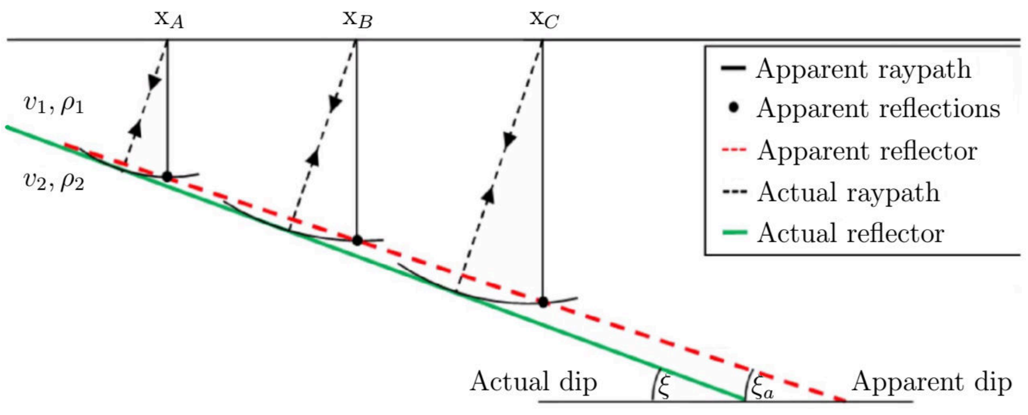


Figure 1.10: Seismic migration.

1.2.3 A View on Sedimentary Processes: Turbidite Systems

Among a wide list of geological depositional mechanism, we mention here the turbidity currents since well spread in the studied area; they are gravity driven turbid mixtures of sediment temporarily suspended in water. The name is derived from their characteristics of being opaque mixtures of sediment and water. They flow down slopes or over a horizontal surface provided that the thickness of the flow is greater up-flow than it is down-flow. The deposit of a turbidity current is a *turbidite*. The volumes of material involved in a single flow event can be anything up to tens of cubic kilometres, which is spread out by the flow and deposited as a layer a few millimetres to tens of meters thick. Turbidity currents, and hence turbidites, can occur in water anywhere there is a supply of sediment and a slope. They are common in deep lakes, and may occur on continental shelves, but are most abundant in deep marine environments, where turbidites are the dominant clastic deposit. As more sediment is deposited from the decelerating flow a deposit accumulates and the flow eventually comes to a halt when the flow has spread out as a thin, even sheet.

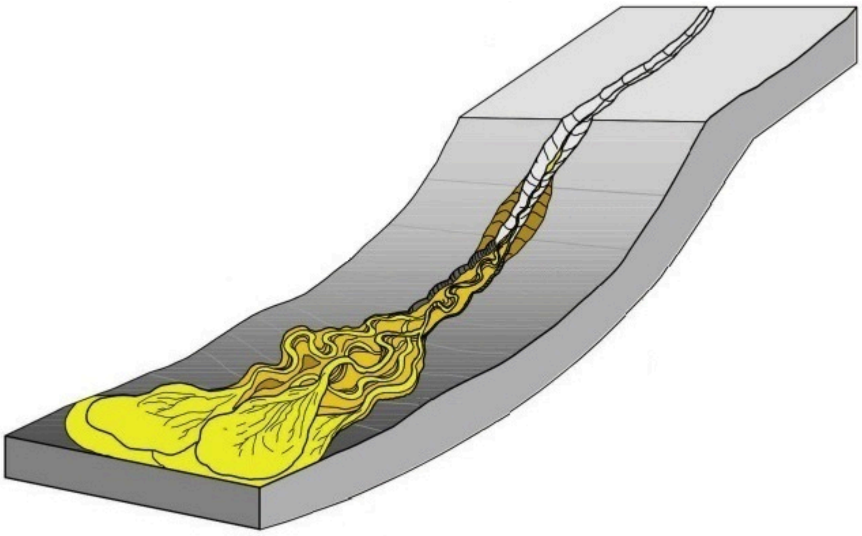


Figure 1.11: Turbidite system.

Lithified[21] accumulations of turbidite deposits may, in time, become hydrocarbon reservoirs and the oil and gas industry makes strenuous efforts to predict the location, overall shape, and internal characteristics of these sediment bodies in order to efficiently develop fields as well as explore for new reserves.

[21]Lithification is the process in which sediments compact under pressure, expel trapped fluids, and gradually become solid rock. Essentially, lithification is a process of porosity destruction through compaction and cementation. Lithification includes all the processes which convert unconsolidated sediments into sedimentary rocks.

Chapter 2

Data, Task and Models

In this work we applied deep learning techniques to the knowledge field of Oil & Gas (O&G) exploration. This thesis is made possible by the collaboration with Eni S.p.A., in particular with the Upstream & Technical Services team. Through a fruitful series of meetings, an industrial objective for Eni to apply deep learning techniques was focused on. This led to the definition of a dataset on which this objective could be applied, which Eni kindly provided us with. Some data and text could be anonymised due to NDA restrictions.

Figure 2.1: Study area: large view. The orange box represent the specific location of study area.

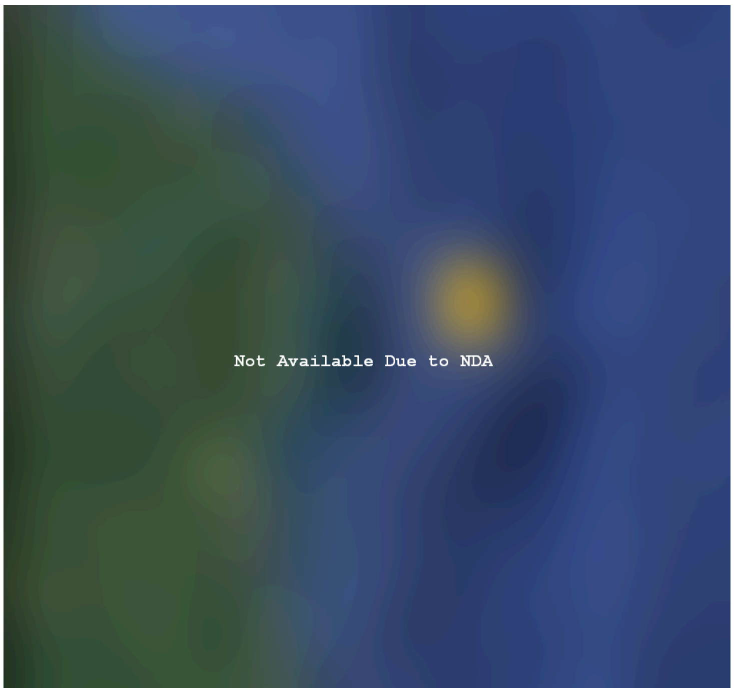


Figure 2.2: Study area: detail. The orange box represent the specific location and dimension of study area. We introduce the coordinate system used through this work: *Cross Line* (XL) and *In Line* (IL). Xx xxxxxxxxx xxxxxx xxxxxx xxxxx xxx xx xxxxxxxxx xxxxxx xxxx.

2.1 Dataset Description

Xxx xxxxx xxxx xx xxxxxxx xx xxx xxxxxxxx xxxx xx xxx xxxxxxxxxx xxxxxxx xxxx xxxx xxx xxxxxx xxxx xxxxxxxx. Xxx xxxxxx xxx xx xx xxx xxxxx xxxx xxx xxxxxxx xx xxx xxxxxx xxxxx xxxx xx xxxxxxxxxxxxx xx xxxxxxx xxxxxx xxxxxxxxxx.

The dataset Eni gave us comprises of:

- Two high quality 3D Pre-Stack Depth Migrated (PSDM)[1] Volumes: one *Near* reprocessed angle stack and one *Far* angle stack[2]. These two volumes are different types of stacks, not including all the offsets, but only selected ones, according to being Near (closer receivers offsets to the source, meaning low incidence angles of reflections) or Far (far receivers offsets to the source, meaning higher incidence angles of reflections). They are sensitive to different seismic properties; for instance we could say that the Far is more sensitive to fluid presence.

[1]PSDM indicates that the seismic migration procedure is done before the CMP stacking one.

[2]The difference between them lies in the range of angle reflections collected in the CMP stacking procedure: $\gamma_{\text{Near}} \in [3°, 18°]$ and $\gamma_{\text{Far}} \in [33°, 48°]$. A result is that the Near volume has a higher spatial resolution than Far volume.

- Three manually interpreted surfaces or *horizons*[3] that have at least one embedded turbidite channel-base: Base I, Base II, Base III.
- Four high quality exploration porosity well logs[4] (W_1, W_2, W_3, W_5) each one composed of: the actual measured porosity log and the relevant frequency filtered version[5].

2.1.1 Seismic Volumes

Spatial extension and sampling density of seismic volumes are summarized in Figure 2.3.

Not Available Due to NDA

Figure 2.3: Seismic volumes overview. Depth axis points downward and has zero value on sea level.

Xx xxx xx xxxxxxxxxx xxxxxx xxxxx xxxxxxxx xxx x xxxxxxx xx xxxxxxxxx x xxxxx x xxxxxxxxx xxxxxxxxx xx xxxxx x. Xx xxx xx xxxxxxxxxx xxxxxx xxxxx xxxxxxxx xxx x xxxxxxx xx xxxxxxx x xxxxx x xxxxxxxxx xxxxxxxxx xx xx x. Xxx xxxxx xxxxxxx xxxxxxxx x xxxxxx xx xxxxxxx xxxxxx x xxxxx xxxxxxxxxx x xxxxxxx xx xxxxxxxxxxxxxx xxxxxxxxxxxx x xx xxxxxx. Xxx xxx xxxxxxx xxxxxxxx xxxxxx xx x xxxxxx xx xxxxxxx xxxxxx x xxx xxxxx xx xxx xxxxx xxxxxx xx xxxxx xxxxxxx.

Therefore volumes dimensions expressed as tensor indexes are:

- **Near**: 1,901 XLs, 606 ILs, 2,001 Depths.
- **Far**: 1,901 XLs, 606 ILs, 1,601 Depths.

We finally point out that in both volumes, *voxels*[6] are non-isotropic and have the same size which is 12.5 m x 25 m x 2.5 m. Voxels are huge hence oil and gas predictions has to be extremely accurate.

[3]An informal term used to denote a surface constituted of a distinctive layer of rock that is represented by a reflection pattern in seismic data. A horizon can be thought of as a geological snapshot of the surface history. These horizons are interpreted using Near seismic volume, see [16].

[4]A measurement versus depth of one or more physical quantities in or around a hydrocarbon exploration well.

[5]Frequency filtration has been done by convolution between measured log and a proper wavelet φ in order to match seismic data spatial resolution.

[6]A voxel represents a value on a regular grid in three-dimensional space. The 3D analogous of 2D pixel.

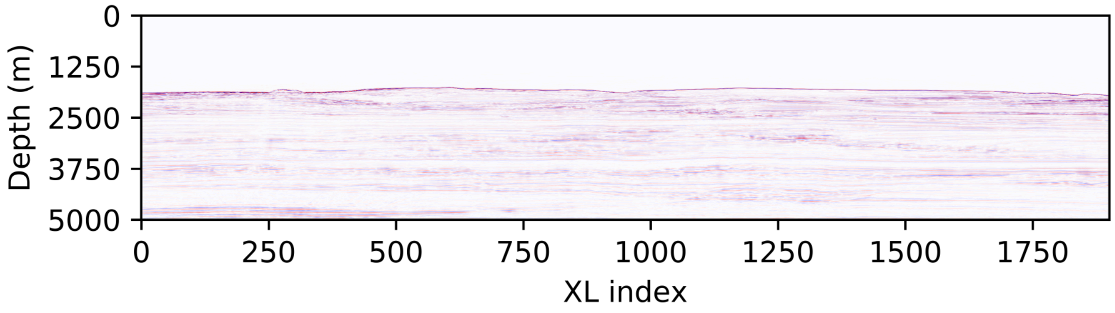


Figure 2.4: IL-slice of Near volume: axes are in reciprocal proportion to the study volume.

In Figure 2.4 we show how looks like an IL-slice of Near volume seen in proportion to the real physical volume. Seismic voxels contains scalar values directly related to the local variation of *acoustic impedance*[7], or equivalently, related to the acoustic reflection coefficient R_C. Seismic volumes are usually graphically rendered in diverging pseudo color which associates blue to negative values, red to positive values and white to zero crossing.

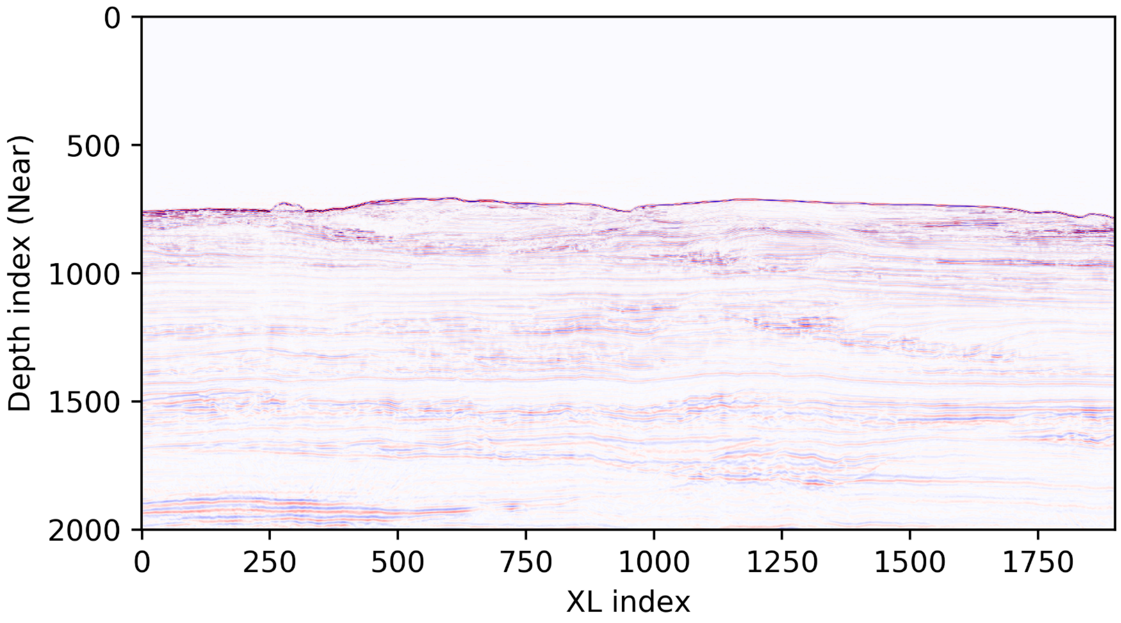


Figure 2.5: IL-slice of Near volume.

[7]Acoustic impedance Z_M is a physical property of matter. It describes how much resistance a sound beam encounters as it passes through a layer. It is defined by the product between density ρ_M and speed of sound waves in medium v_M : $Z_M = \rho_M v_M$.

2.1.2 Seismic Horizons

Seismic horizons play a key role in this work and they are expressed as a cloud of points.

Figure 2.6: Base I horizon. XL and IL axes maintains reciprocal proportion to the study volume. Base I horizon is rendered trough depth in pseudo color. Wells are superimposed to understand their exact locations.

Geological and geomorphological experience allow to recognize erosive patterns that are attributable to turbidite channel-base systems; the exact location and segmentation of them. However, it is not a trivial task and it involves geophysical expertise.

Remaining horizons are showed next. As you might notice, horizons are presented in order of decreasing depth; Base I, in fact, is the deepest and hence the oldest.
We finally point out the notion of *sequence* which is a group of relatively conformable strata that represents a cycle of deposition and is bounded by unconformities.

- **Sequence I**: bounded by Base I and Base II.
- **Sequence II**: bounded by Base II and Base III.
- **Sequence III**: bounded by Base III and a surface of non-geophysical interest called Top Interp.

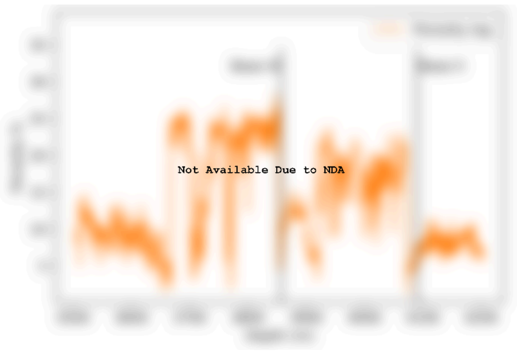
Figure 2.7: Base II horizon.

Figure 2.8: Base III horizon.

2.1.3 Well Logs

Well logs are subsurface property measurement acquired in the borehole. Seismic volumes, in fact, are a non-direct information, just the best result of the O&G industry to solve a poorly-posed problem such as the time-depth inversion of seismic traces.

In this case, the porosity logs are a more direct and physical measurement indicating the local pore volumes in percentage over the total. High porosity, for this kind of study area, is a strong indicator of sandy clastic sediments[8] which indeed suggest the presence of a turbidite channel infill and therefore a candidate reservoir (lead).



(a) W1 measured porosity log.

(b) W1 frequency filtered porosity log.

Figure 2.9: W1 porosity well logs.

Frequency filtering of measured porosity logs is done by convolution whit the same depth-dependent[9] wavelet used in seismic survey processing. The aim of that is to match the density of information content between logs and seismic volumes.

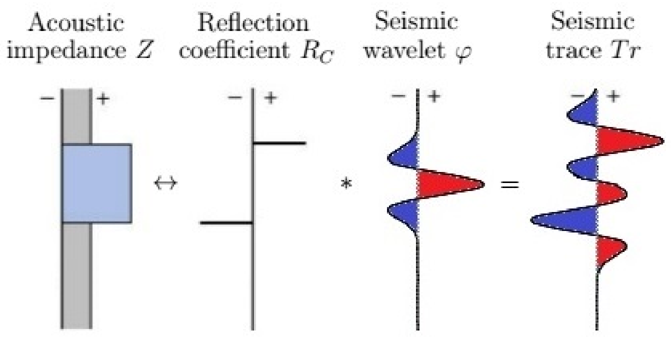
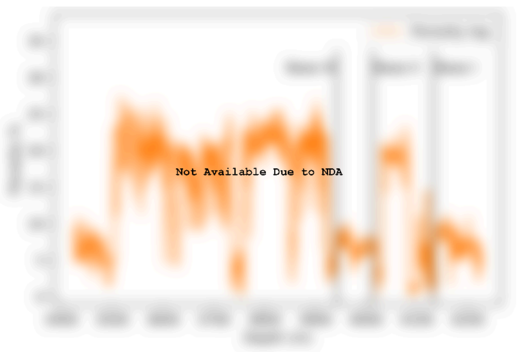


Figure 2.10: Seismic wavelet explanation.

[8]Sediment consisting of broken fragments derived from pre-existing rocks and transported elsewhere and redeposited before forming another rock.

[9]An important fact is that rock acts as a low pass filter therefore wavelet shape is depth-dependent. You may notice it in Figure 2.5.

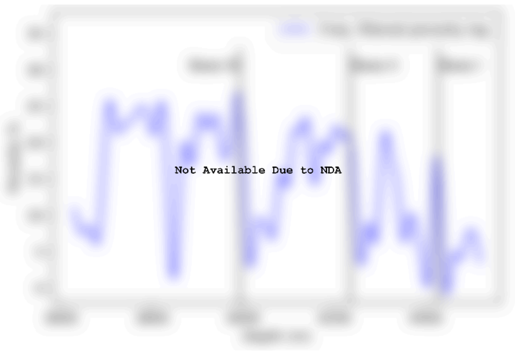


(a) W2 measured porosity log.

(b) W2 frequency filtered porosity log.

Figure 2.11: W2 porosity well logs.

(a) W3 measured porosity log.



(b) W3 frequency filtered porosity log.

Figure 2.12: W3 porosity well logs.

(a) W5 measured porosity log.

(b) W5 frequency filtered porosity log.

Figure 2.13: W5 porosity well logs.

All informations about dataset are summarized in Figure 2.14.

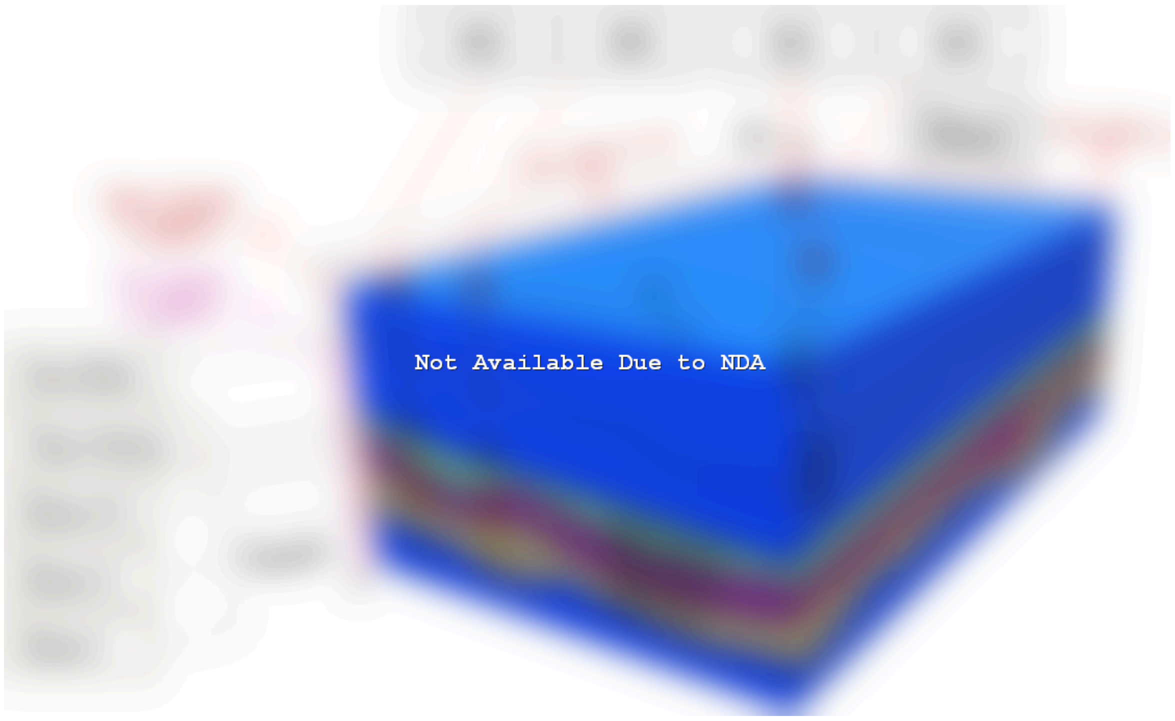


Figure 2.14: Dataset comprehensive scheme.

Data Format and Visualization Tools

A significant part of the work in this thesis consisted in understanding and learning how to manage the formats used in the O&G exploration industry. In particular, learning how to read and produce files in SEG-Y file format. Another important part was to visualize the data and results produced by deep learning algorithms.

SEG-Y File Format

The SEG-Y file format is one of several standards developed by the Society of Exploration Geophysicists (SEG) for storing geophysical data. It is an open standard, and is controlled by the SEG Technical Standards Committee, a no-profit organization. The format was originally developed in 1973 to store single-line reflection seismic (traces) digital data on magnetic tapes. The specification was published in 1975.
However, since its release, there have been significant advancements in geophysical data acquisition, such as 3-dimensional seismic techniques and high speed, high capacity recording. The most recent revision of the SEG-Y format was published in 2017, named

the rev 2.0 specification. It still features certain legacies of the original format (referred as rev 0), such as an optional SEG-Y tape label, the main 3,200 byte textual EBCDIC character encoded tape header and a 400 byte binary header.

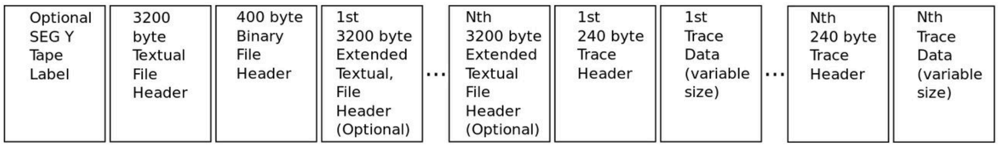

Optional SEG Y Tape Label
3200 byte Textual File Header
400 byte Binary File Header
1st 3200 byte Extended Textual, File Header (Optional)
...
Nth 3200 byte Extended Textual File Header (Optional)
1st 240 byte Trace Header
1st Trace Data (variable size)
...
Nth 240 byte Trace Header
Nth Trace Data (variable size)

Figure 2.15: SEG-Y file structure. The file is organized by traces and these are typically arranged as IL-slice, i.e. $\{\text{Trace}(il = 0, xl = 0), \text{Trace}(il = 0, xl = 1), \dots, \text{Trace}(il = 0, xl = XLs), \text{Trace}(il = 1, xl = 0), \dots, \text{Trace}(il = ILs, xl = XLs)\}$. This format is therefore not conceived and optimized for a fast access of 3D sub-volumes of data.

To manage the SEG-Y format in Input/Output (IO) we used the *segyio*[10] library which allowed us to read and produce SEG-Y files to be supplied to Eni as input for their visualization and processing tools.
Segyio is a small LGPL[11] licensed C library for easy interaction with SEG-Y formatted seismic data, with language bindings for Python and Matlab. However, segyio has some limitations as it does not support the entire standard or all exotic (but correctly) formatted files. Some assumptions are made, for example: all traces in a file are supposed to be the same size as the sample and all lines are supposed to have the same number of traces.

Visualization Tools

Visualization in this field plays a key role due to the intrinsically 3D nature of geobodies. It was therefore important to be equipped with graphical tools to improve the understanding of the provided dataset. Petrel is not free of charge and have therefore only been exploited through Eni's team.

- **Petrel**

  Petrel is a software platform used in the exploration and production sector of the petroleum industry. It allows the user to interpret seismic data, perform well correlation, build reservoir models, visualize reservoir simulation results, calculate volumes, produce maps and design development strategies to maximize reservoir exploitation. Petrel is developed and built by Schlumberger.

[10]https://segyio.readthedocs.io/en/latest

[11]The GNU Lesser General Public License (LGPL) is a free-software license published by the Free Software Foundation (FSF). The license allows developers and companies to use and integrate a software component released under the LGPL into their own (even proprietary) software without being required by the terms of a strong copyleft license to release the source code of their own components.

Using the latest advanced GPU rendering, the Petrel Seismic Volume Rendering and Extraction module enables quick and interactive blending and rendering of multiple seismic volumes with extreme clarity to detect anomalies, delineate structural and stratigraphic features, isolate areas of interest, and then instantly extract what is visualized into a 3D object called a geobody. One can create complex selection events to delineate complex structural and stratigraphic features such as channels, deltas, or fractures. Accurate interpretation of those features is made possible by the complete set of tools, such as advanced horizon amplitude-based and waveform-based horizon autotracking, multi-Z interpretation, and interactive mesh editing. One can also extract 3D geobodies and assign geological templates to them providing the bodies with instant geological meaning.

- **OpendTect**

  OpendTect is a complete open source seismic interpretation package, which is widely used in the industry and that it can be downloaded at no cost from OpendTect. OpendTect contains all tools, needed for a 2D and/or 3D seismic interpretation: 2D and 3D visualization, horizon and fault trackers, attribute analysis and cross-plots, time-depth conversion, etc.

- **Mayavi**

  Mayavi[12] is a scientific data visualizer written in Python. Mayavi is free and distributed under the BSD[13] license. The latest version of Mayavi is called Mayavi2.

  Mayavi2 seeks to provide easy and interactive visualization of 3D data, or 3D plotting. It does this by the following: an (optional) rich user interface with dialogs to interact with all data and objects in the visualization, a simple and clean scripting interface in Python, including ready to use 3D visualization functionality similar matplotlib, harnesses the power of VTK without forcing you to learn it.

2.2 Task Description

A valuable industrial objective is to quickly and semi-automatically characterize the presence, location and extension of leads, in this case, turbidite systems.
The most natural choice for such a task is the binary classification: object of interest, background. This would naturally fit into the framework of supervised learning, but this kind of task can be set up only if we have, or at least we can define the labels. Therefore

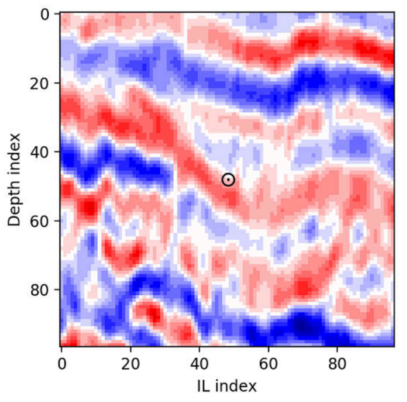
[12]https://docs.enthought.com/mayavi/mayavi

[13]BSD licenses are a family of permissive free software licenses, imposing minimal restrictions on the use and distribution of covered software. The BSD license is a simple license that merely requires that all code retain the BSD license notice if redistributed in source code format, or reproduce the notice if redistributed in binary format.

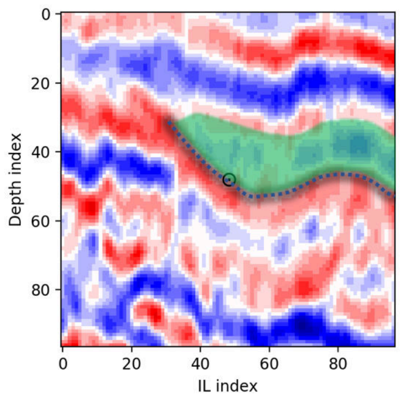
the choice of algorithms task definition must be done taking into account what data is available.

Up to this point, we should ask, are these objects characterized by their sinuous and meandering shape or by their texitural internal layering? Probably both...
The information on where are the turbidite sediments infills are a very difficult one to gain and it requires a lot of time, for geologists and geophysicists to correctly interpret the seismic volume, and money to collect well logs in order to establish high porosity regions. Furthermore, even if this effort is placed at work, a 3D reliable label with sufficient spatial resolution is impractical. On the other hand, the extraction of horizons is a relatively fast and semi-automated process done using 3D visualization and processing software, such as Petrel. This is because the seismics is the most important attribute in interpreting and recognizing horizons.

Since we do not have the information on where the turbidite sediments infills are, we can not set up the classification task as a direct search for this kind of objects. However, we do have three horizons with at least one embedded turbidite channel-base system. But is the information embodied in the channel-base sufficient?



(a) Near XL-slice centered around a channel-base voxel.



(b) Same slice as 2.16a with superimposed channel-base (dotted line) and sediments infill (green region).

Figure 2.16: Form or content dilemma. An interesting slice containing a turbidite channel is presented. We see shape of a turbidite channel-base, the form, and textural pattern of the sediments infill, the content.

In Figure 2.16 we see a slice of seismics centered around a channel-base voxel. We notice two important things that are present: channel-base is characterized by a concave shape and sediments infill with its texitural pattern that lies above. Therefore, even if we are expressing the task as a search for turbidite channel-base systems, using a big enough context around every channel-base voxel we give to the model both pieces of information: shape and content.

After all these consideration, we choose to define the deep learning task as binary classification between turbidite channel-base and background using as input a sub-volume context or a 3D patch. Or, in other words, we choose to build models that for every voxel answer the question:

"*Does this voxel belong to a sub-volume that represents a channel-base system?*"

2.3 Proposed Models

We previously introduced the concept of the 3D patch in the context of a patch-based classification task. The 3D patch is the earlier called sub-volume which is a N_{patch} side volume.

At the end of the sub-sub-section 1.1.2.3 we talked about the curse of dimensionality, which focuses on how the number of features should be kept under control in relation to the available examples in order to ensure a good learning process. In case we are using N_{patch} side volumes as input patches we are looking for a solution to a problem with the complexity that grows as $O(N_{\text{patch}}^3)$, since $N_F = N_{\text{patch}}^3$. Given that N_{patch} must also be large enough to provide an adequate context for the learning process, the risk of not having enough examples is considerably high.

We, therefore, decided to take an alternative route before admitting the need to move to full 3D. It has been decided to work with a certain number of 2D patches, according to what could be defined as a 2.5D approach. This approach makes the complexity of the problem proportional to a $O(N_{\text{patch}}^2)$. The most intuitive choice to transform an intrinsically 3D problem into a 2D one is to take N_{slice} slice along the coordinated axes of the subvolume, in a mode that we have named "windrose". For a graphical representation see Figure 2.18.

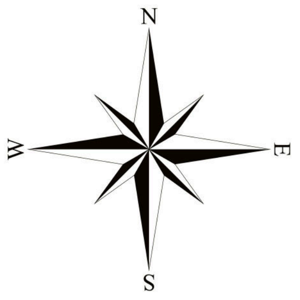
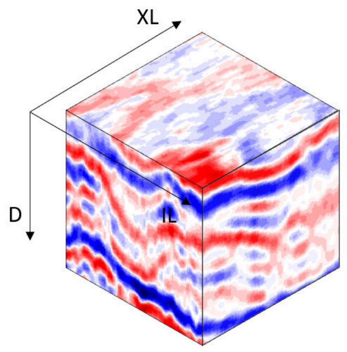


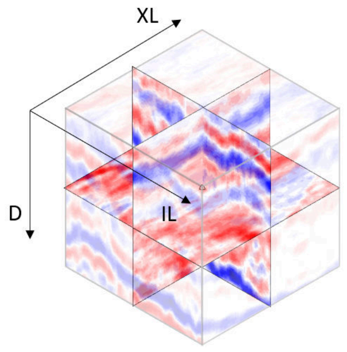
Figure 2.17: Windrose diagram.

2.3.1 Input Patch

Once we decided that we want to use a 2.5D approach, it remains to be defined how to exactly build the input patch from the three selected slices.
One idea could be to place them horizontally side by side to form a single image of size [1]x[N_{patch}]x[$3N_{\text{patch}}$]. Even if this may seem an obvious choice, with respect to the functioning of the CNN models, it implies that the same convolution filters act simultaneously for all three slices. However, they carry very different information, for example the D-slice, and not to mention the fact that there are discontinuities in the junction points that models must learn to handle it.



(a) 3D patch around a channel-base voxel.



(b) Three 2D patches extracted in a windrose fashion. Slices are extracted perpendicular to coordinate axes.

Figure 2.18: Windrose patch description. In this case $N_{\text{slice}} = 3$ but we can generalize easily to the case where $N_{\text{slice}} < N_{\text{patch}}$. The case $N_{\text{slice}} = N_{\text{patch}}$, no matter how slices are selected, implies the same complexity as the direct 3D approach and therefore is not convenient.

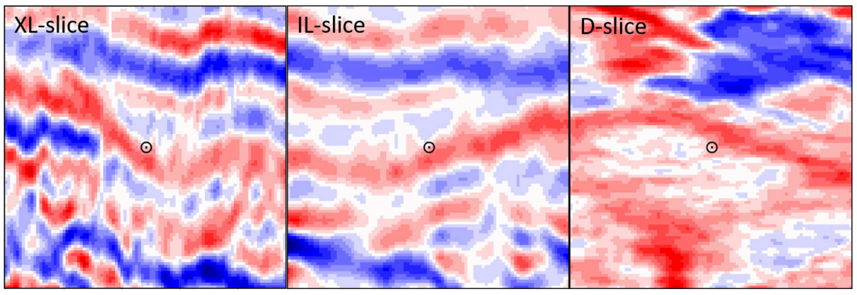


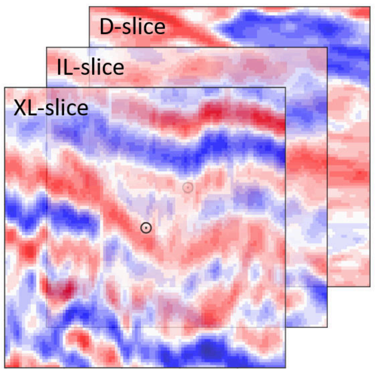
Figure 2.19: Windrose h-stack patch description.

We, therefore, opted for an input patch definition that would allow the network to use different filters for each slice. The result of this reasoning was to stack the slices on top of each other, as happens with RGB image channels. Thus producing patches of size $[3]\text{x}[N_{\text{patch}}]\text{x}[N_{\text{patch}}]$.

2.3.2 Models Architecture

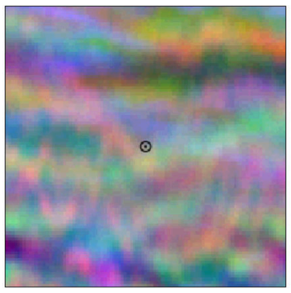
Regarding the architecture of the models, which are naturally 2D CNNs, we decided to study two architecture in particular, called here “CNN A” and “CNN B”.

As you can see in Figure 2.21, the last two connections has a further information in square brackets which is relative to the regularization techniques called dropout. Dropout is drawn in brackets because we analized the CNN A architecture both with and without dropout, called respectively: “CNN A Dropout” and “CNN A”.



(a) Windrose slices are concatenated one above the other in a three-channel way.

$\Longleftrightarrow$



(b) Spectral interpretation of the concatenated windrose slices as a RGB image.

Figure 2.20: Windrose spectrum patch description.

CNN B architecture is described in Figure 2.22. As before we analized architecture both with and without dropout, called respectively: "CNN B Dropout" and "CNN B".

The main difference from the two architectures is the depth, or rather the compositionality of the inner representation. CNN B, in fact, better represents the convolution network paradigm, having many convolutional layers and therefore feature maps built as the composition of the previous feature maps. In addition to having more convolutional layers, CNN B also has an extra layer of max pooling that makes its internal representation more spatially invariant. These differences are strongly reflected in the representational capacity. The representational capacity can be roughly estimated as the number of tunable parameters, the weights $\boldsymbol{\theta}$. Of course, the number of weights depends on the size of the input patch. Convolutional connections, in fact, shrink the size of the incoming feature maps, as max pooling connections naturally do. In Table 2.1 you can see a huge difference, in terms of weights, between the two architectures as the size of the input patch changes.
In summary, we have that although CNN B is a better representative of the convolutional paradigm it has way less representational capacity than CNN A, which reflects however on the number of examples needed to achieve good learning.

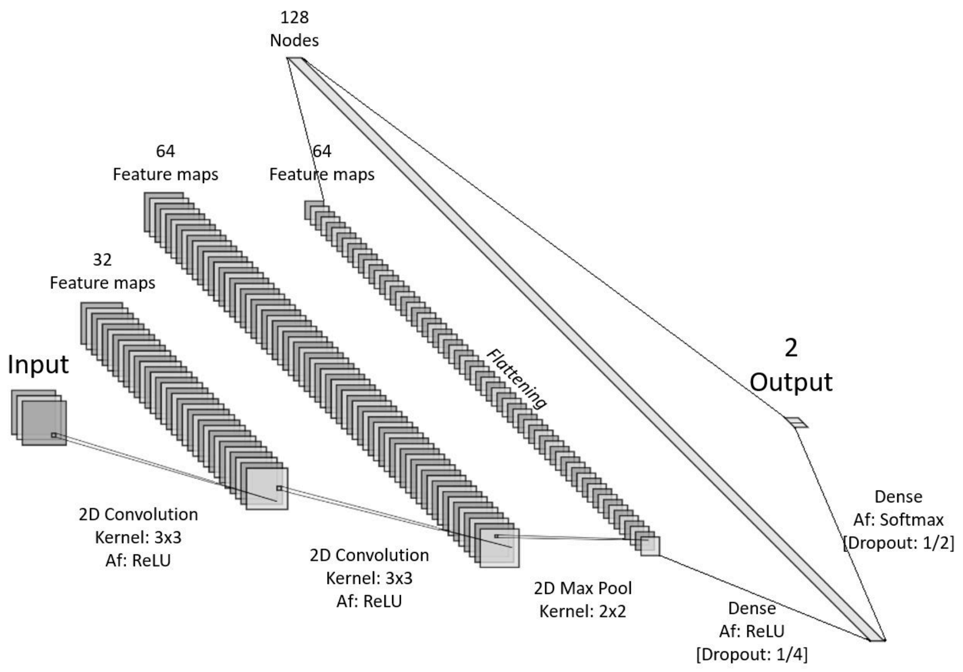


Figure 2.21: CNN A architecture. We can see early convolutional layers, expressed as a certain number of feature maps, connected trough a 2D convolution operation with specified kernel size. Second hidden layer is conned to third hidden layer by a 2D max pooling operation which halves feature maps side. Flattening procedure simply reshapes all elements in a row, so that a dense connection can be made. For all layers but the output one, which has softmax as usual for classification tasks, the activation function is the ReLU function. Output layer has two nodes, one for the background class prediction probability $P_{\text{Background}}$ and one for turbidite channel-base prediction probability P_{Channel}. The last two connections has a further information in square brackets, which is dropout and relative dropping nodes percentage.

	N_{patch}			
Model architecture	17	33	49	65
CNN A	314,690	1,625,410	3,984,706	7,392,578
CNN B	-	205,826	211,970	226,306

Table 2.1: Representational capacity for proposed models. Number of tunable parameters for the two architectures considerated at varying input patch size. CNN B can not operate on patches smaller than $N_{\text{patch}} < 24$ due to feature map shrinkage by convolutional and max pooling layers.

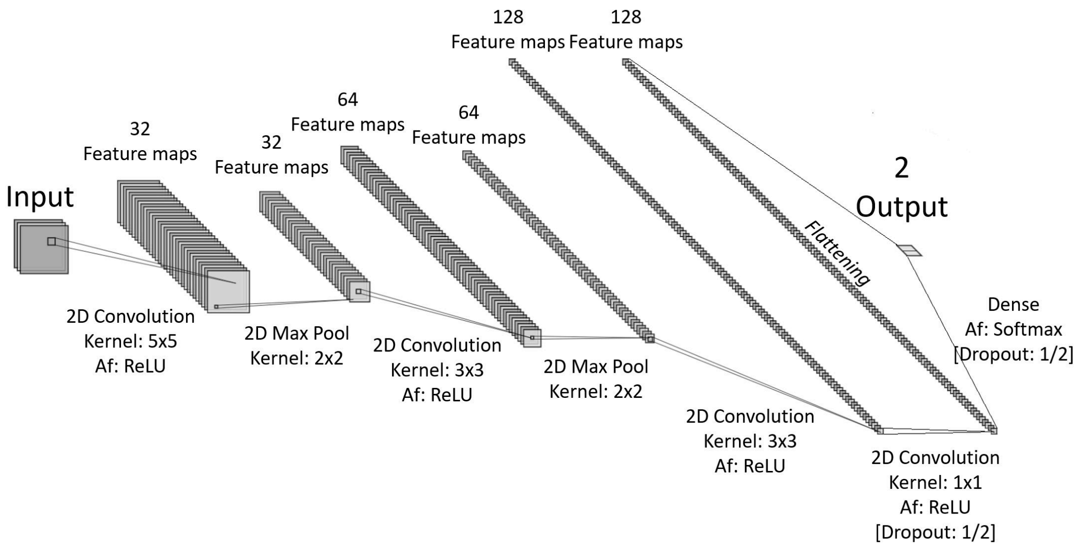


Figure 2.22: CNN B architecture. This architecture is proposed in [10].

Chapter 3

Implementation and Results

This chapter describes the implementation of the deep learning problem described so far and the results obtained.
First of all, we highlight the use of Python language and the TensorFlow library coupled with the use of Nvidia CUDA capable GPUs. As expressed in section 1.1, nowadays any experimental deep learning project can't avoid the use of GPUs because these allow studying multiple possibilities in acceptable deadlines.

TensorFlow is a free and open-source software library developed by the Google Brain team that is also used for machine learning applications such as neural networks. It was released under the Apache License 2.0[1] and it is used for both research and production at Google. TensorFlow can run on multiple CPUs and GPUs with optional CUDA extensions.

In this work we used two GPUs, one relative to my laptop and one relative to the workstation provided by Bioretics. The most intensive tests were conducted entirely on the workstation.

Here's a list of the main characteristics of used GPU's.

- **Laptop**: Nvidia GeForce MX 150 with 2 GB of dedicated memory.
- **Workstation**: Nvidia GeForce GTX 1080 with 8GB of dedicated memory.

3.1 Implementation

In the previous chapter, the deep learning problem has been explicitly defined. It is a binary classification between background and turbidite channel-base. This choice was made considering the available dataset, in fact, we have horizons with at least one channel

[1]The Apache License is a permissive free software license written by the Apache Software Foundation (ASF). It allows users to use the software for any purpose, to distribute it, to modify it, and to distribute modified versions of the software under the terms of the license, without concern for royalties.

immersed. Now, the fundamental step is to generate a labeled training/test set and to do this it is necessary to define which are the channel voxels and which are not.

Is it possible to define channel voxels purely by observing the geometry of the horizon? In the following you can see the three horizons represented through the depth in grayscale pseudocolor, associating black to deep regions and white to shallow ones.

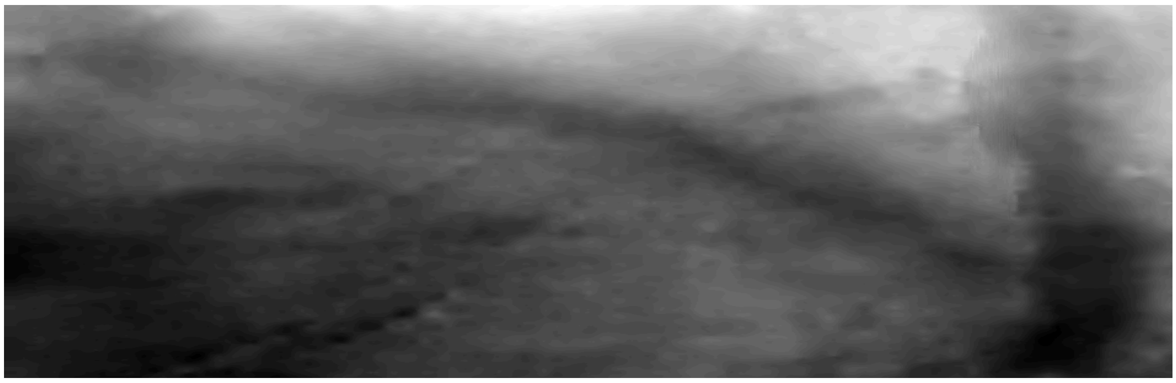


Figure 3.1: Base III horizon with depth as grayscale pseudo color.

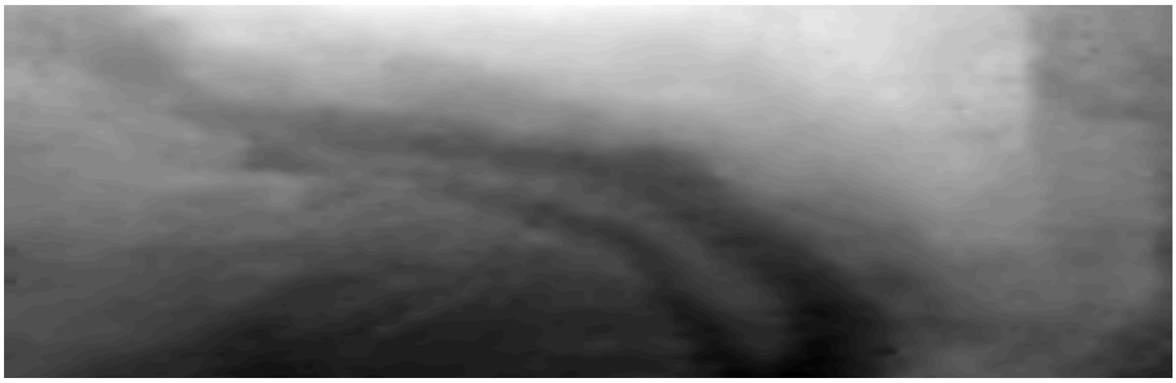


Figure 3.2: Base II horizon with depth as grayscale pseudo color.

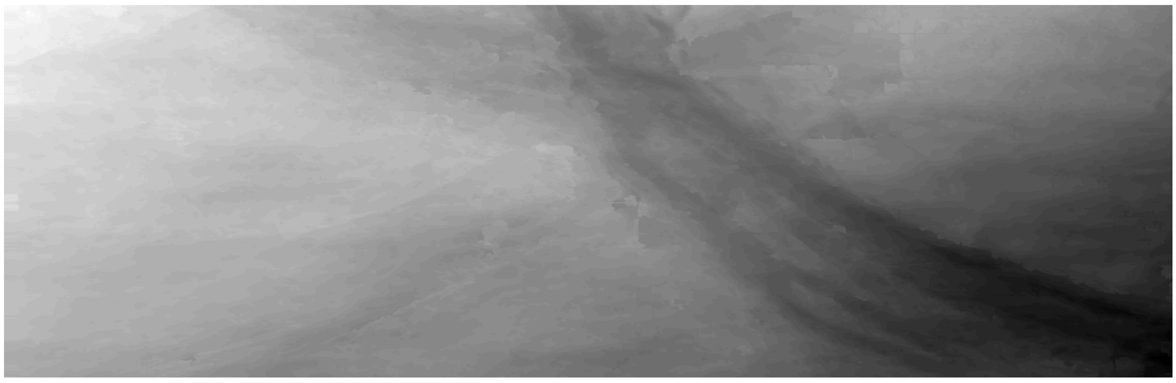


Figure 3.3: Base I horizon with depth as grayscale pseudo color.

From previous images, it is possible to detect the presence of some channels even if a sort of darkening seems to prevent an optimal vision of the horizon topology. The motivation for this aberration lies in the fact that the horizons can have a non-null average slope, in particular, all three horizons show a slope towards increasing IL.
To remove this effect we decided to process the horizons in a way that visually improves the images and helps the human eye to perform segmentation. The idea is to remove the contribution of the average gradient from the horizons as summarized in Figure 3.4.

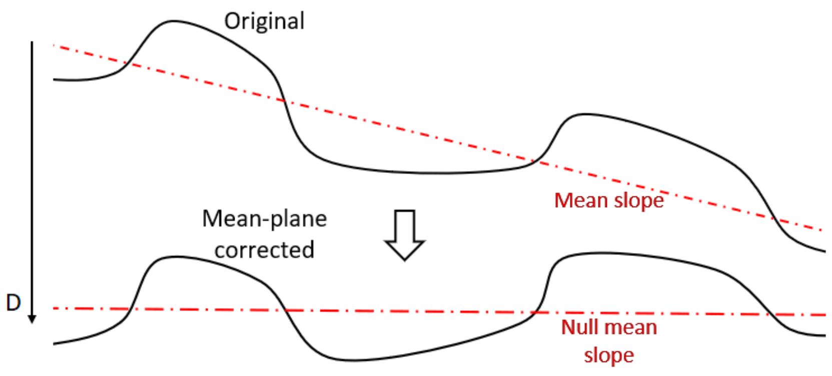


Figure 3.4: Mean-plane correction.

In order to explain how we implemented this correction it's useful to describe the horizon as a differentiable function defined on the XL-IL plane that gives the corresponding depth[2].

$$\begin{aligned} \text{Hor} : \mathbb{R}^2 &\longrightarrow \mathbb{R} \\ (XL, IL) &\longmapsto \text{Hor}(XL, IL) = D_{\text{Hor}} \end{aligned} \tag{3.1}$$

With this mathematical framework we can define the mean gradients with respect to coordinate axes: $\overline{g_{XL}}$, $\overline{g_{IL}}$.

$$\begin{aligned} \overline{g_{XL}} = \overline{\frac{\partial \text{Hor}}{\partial XL}} &:= \frac{1}{(\text{XLs})(\text{ILs})} \sum_{i,j} \frac{\partial \text{Hor}}{\partial XL} \Big|_{(\text{XL}_i, \text{IL}_j)} \\ \overline{g_{IL}} = \overline{\frac{\partial \text{Hor}}{\partial IL}} &:= \frac{1}{(\text{XLs})(\text{ILs})} \sum_{i,j} \frac{\partial \text{Hor}}{\partial IL} \Big|_{(\text{XL}_i, \text{IL}_j)} \end{aligned} \tag{3.2}$$

[2]We can suppose that our horizons are a sampled version of the continuous horizons defined in equation 3.1.

It's easy now to define a function MPlane that assign to every point in the XL-IL plane, the corresponding depth of the mean plane, or even the horizon interpolating plane[3]

$$\text{MPlane}(XL, IL) := \overline{g_{XL}}XL + \overline{g_{IL}}IL + \text{const} \tag{3.3}$$

Mean-plane corrected horizons are defined in equation 3.4 in such a way that $\overline{g_{corr,XL}} = \overline{g_{corr,IL}} = 0$.

$$\text{Hor}_{corr}(XL, IL) := \text{Hor}(XL, IL) - \text{MPlane}(XL, IL) \tag{3.4}$$

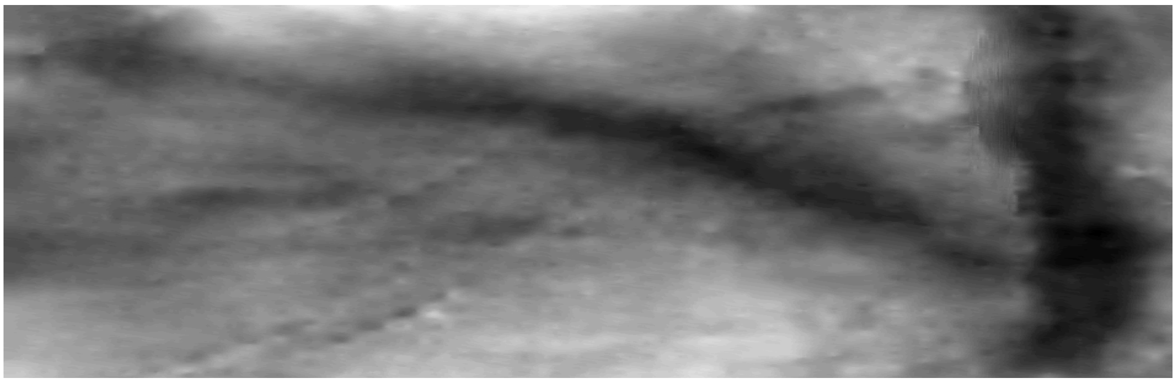


Figure 3.5: Base III mean-plane corrected horizon with depth as grayscale pseudo color. A channel-base is clearly visible on the right hand side that crosses from top to bottom the image. Other channel-bases may be present but not as visible to us.

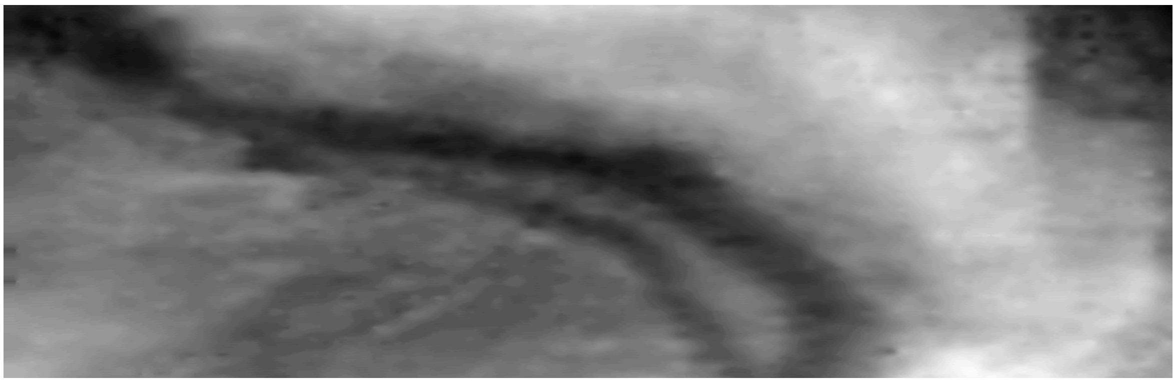


Figure 3.6: Base II mean-plane corrected horizon with depth as grayscale pseudo color. An S-shaped channel-base is clearly visible diagonally across the image. This appears to be a system of two channels that have split in two at the bottom of the image. An uncertain region appears to be the one at the top right.

[3]We defined the MPlane function up to a constant because it is irrelevant to our purpose. The reason is that we are not interested in the actual depth value of mean-plane corrected horizons, but only to enhance their visual representation in grayscale pseudo color which adapts itself to the data range.

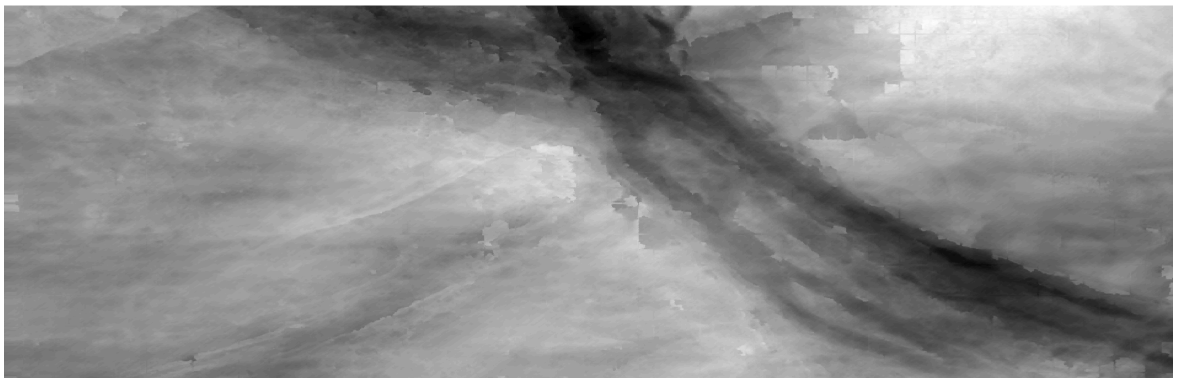


Figure 3.7: Base I mean-plane corrected horizon with depth as grayscale pseudo color. A system of channel-bases is clearly visible in the center of the image crossing from the top center to bottom right.

After examining all the horizons and noticing the presence of evident channels, we decided to focus heavily on Base II. This horizon, in fact, has a widespread channel that crosses a significant part of the horizon and therefore can provide a lot of information to the models. Moreover, Base II, being between Base I and Base III, is an obvious candidate to focus out attention on because it can provide useful information also for the generalization in its surroundings, which we can control by monitoring what models predict on Base I and Base III.
In Figure 3.8 we show how Base II appear in 3D space.

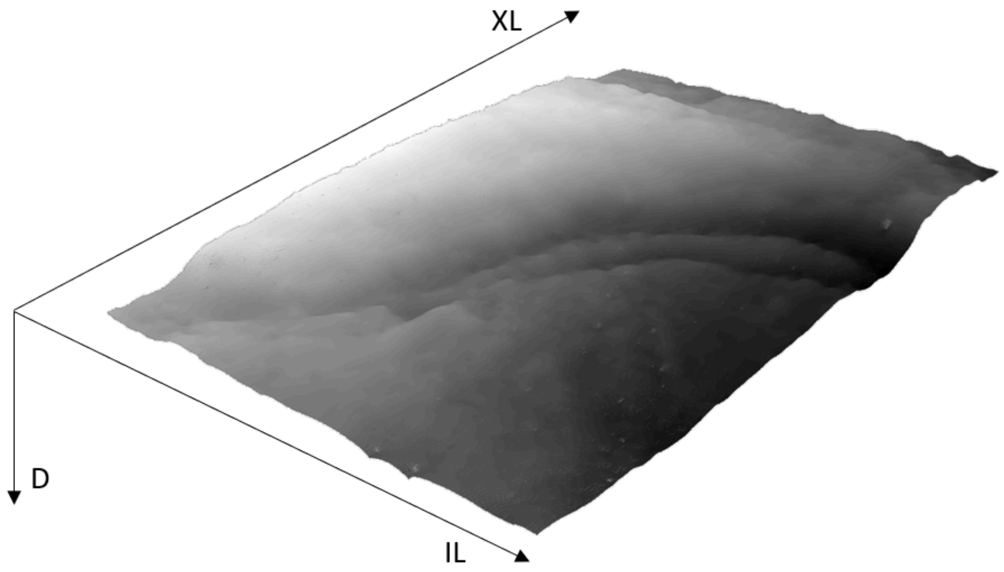


Figure 3.8: Base II horizon in 3D space.

3.1.1 Manual Channel-Base Segmentation

In this section, we describe how we proceeded to extract information about the location of the channel dipped on Base II horizon. Since we want to extract the channel-base using only topological information, we find two useful information: the magnitude of the gradient and the curvature. To calculate them, it was necessary to take into account that the XL and IL axes do not have a homogeneous sampling. As you can see from Figure 2.3 $\Delta_{IL} = 2\Delta_{XL}$, and we had to take this into account defining two variables of scale: $S_{XL} = 1$ and $S_{IL} = 2$. Now you can correctly define the magnitude or modulus of the gradient vector as expressed in equation 3.5.

$$G_{\text{magnitude}}(XL, IL) := \sqrt{\left[\frac{1}{S_{XL}} g_{corr,XL}(XL, IL)\right]^2 + \left[\frac{1}{S_{IL}} g_{corr,IL}(XL, IL)\right]^2} \tag{3.5}$$

Where

$$\begin{aligned} g_{corr,XL}(XL, IL) &= \frac{\partial \text{Hor}_{corr}}{\partial XL}(XL, IL) \\ g_{corr,IL}(XL, IL) &= \frac{\partial \text{Hor}_{corr}}{\partial IL}(XL, IL) \end{aligned} \tag{3.6}$$

We can also calculate a simplified version of curvature, such that in equation 3.7, and visually combine the two information as in Figure 3.9.

$$\text{Curvature}(XL, IL) := \left[\frac{1}{S_{XL}}\right]^2 \frac{\partial^2 \text{Hor}_{corr}}{\partial XL^2}(XL, IL) + \left[\frac{1}{S_{IL}}\right]^2 \frac{\partial^2 \text{Hor}_{corr}}{\partial IL^2}(XL, IL) \tag{3.7}$$

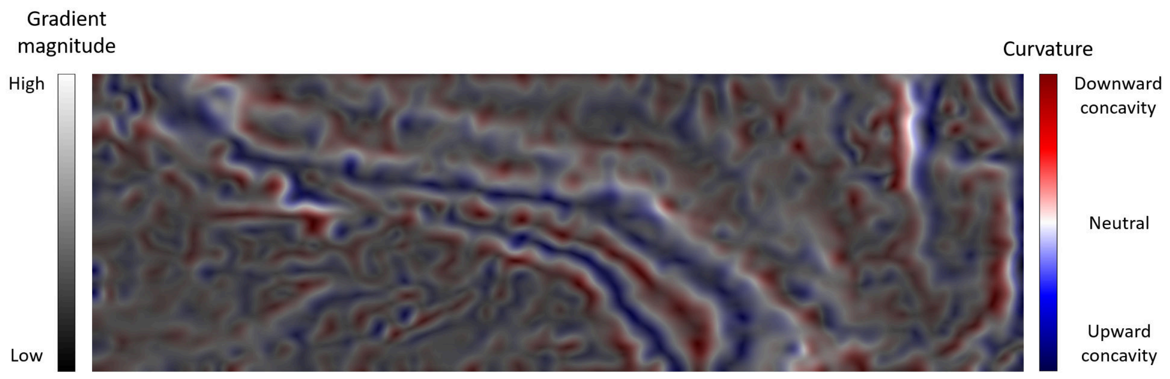


Figure 3.9: Base II gradient magnitude plus curvature. In this image, gradient magnitude and curvature are blended in a certain proportion, respectively 0.75 and 0.25, in order to give a topological hint on where the channel-base is.

Using this information we have manually contoured the region that we believe could correspond to the definition of channel-base, reaching as far as the concavity reaches its maximum downward, i.e. what could be defined as the bank of the channel. The result of this first segmentation can be seen in Figures 3.10. Unfortunately, it is difficult to exactly define where the channel-base ends and it is a part of this work to figure out how this definition affects the classification problem resolution.

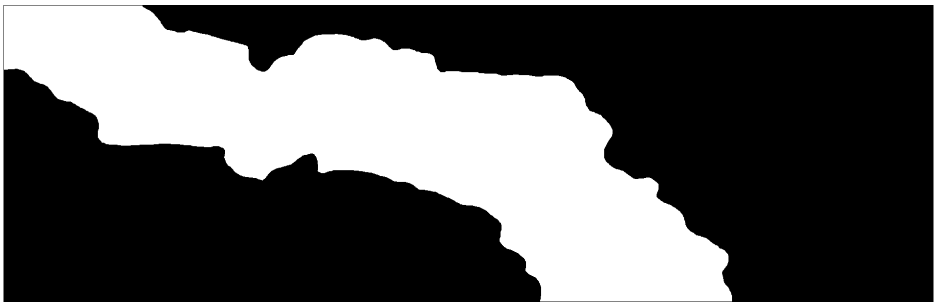


Figure 3.10: Base II first segmentation. As may notice, we completely ignored the topological indications regarding the top right area, considering that part as background. We point out that this choice is not motivated by an a priori knowledge and therefore it should be considered as a possible criticality of this initial segmentation, which adds to the problem of how to define the channel-base boundaries.

3.1.2 Dataset Handling

Now that we have defined, at least on the Base II horizon, which are channel-base voxels and which are background, we just have to define how to build a windrose patch labeled dataset in practice. First of all, the number of voxels on each horizon is $M_{\text{max}} = (XLs)(ILs) = 1,901 \text{ x } 606 = 1,152,006$ but only a part of them has a windrose patch associated with, this is because the latter must be within[4] the seismic data volume. So the number of voxels from which it is possible to extract windrose patches of N_{patch} side, and therefore the maximum number of effective extractable examples, is equal to:

$$M_{\text{eff}} = M_{\text{max}} - \left(2 \left\lfloor \frac{N_{\text{patch}}}{2} \right\rfloor + 2 \left\lfloor \frac{N_{\text{patch}}}{2} \right\rfloor \right) \tag{3.8}$$

As an implementation choice we decided to save on disk 3D patches of side $N_{\text{saved}} > N_{\text{patch}}$ centered around the labelled voxels from which we later extract the windrose patches during simulation. Therefore M_{eff} is actually:

$$M_{\text{eff}} = M_{\text{max}} - 4 \left\lfloor \frac{N_{\text{saved}}}{2} \right\rfloor \tag{3.9}$$

[4]Unless you put into practice volume padding procedures.

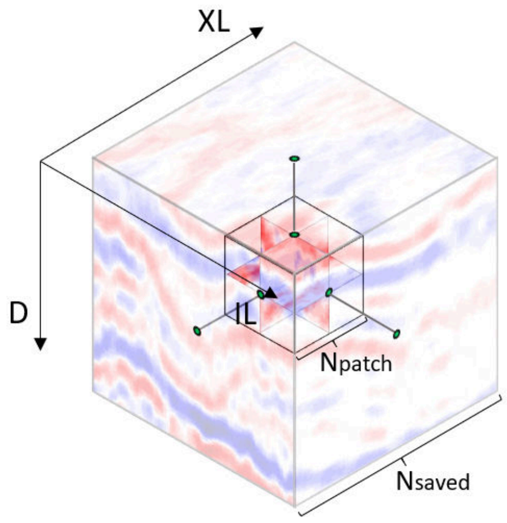


Figure 3.11: 3D central cropping of saved volume and windrose patch extraction.

The decision of saving 3D patches along with the large amount of extractable 3D patches M_{eff} induced, for reasons of memory storage capacity, to actually save a fraction of them. We decided to not select this fraction of 3D patches randomly because it could happen that in some regions patches might be closely picked, and therefore highly correlated, while in others spatially distant. The set of selected 3D patches was therefore defined using the concept of 2D stride: i.e. 3D patches selected having central voxel lying on a 2D grid of step $(\delta_{XL}, \delta_{IL})$.

The number of 3D patches of side N_{saved} saved on disk is therefore:

$$M_{\text{saved}} = \frac{M_{\text{eff}}}{\delta_{XL}\delta_{IL}} \tag{3.10}$$

In our case the trade-off between the request for a sufficiently large dataset and the memory storage requirements led to the choice of $\delta_{XL} = \delta_{IL} = 4$: that is $M_{\text{saved}} = M_{\text{eff}}/16 \sim 70k$ examples.

3.1.3 Preprocessing

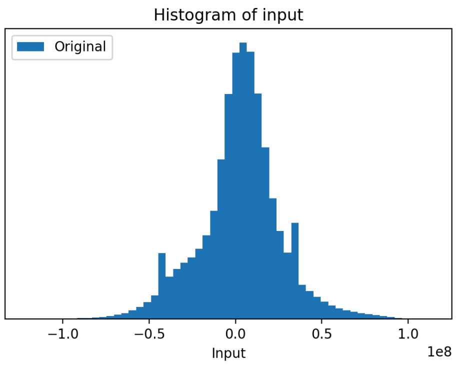
Before implementing a deep learning algorithm it is good practice to study how the numerical range of examples behaves in the training context. The training algorithms in fact, due to numerical crunching reasons, do not work properly if the input values are either too small or too large[5]. In our case values contained in seismic volumes have an extremely wide dynamic range in the order of 10^8:

- **Far**: Far_{min} = -623,911,936.00, Far_{max} = 409,188,992.00.

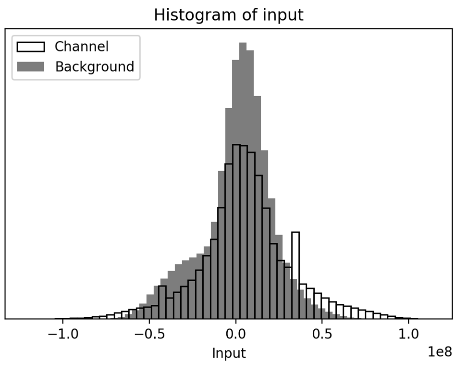
[5]Typically the range $[0, 255]$ of the 8-bit unsigned integers is a good one for CNNs.

- **Near**: Near_{min} = -505,458,176.00, Near_{max} = 402,230,816.00.

To study how the dynamic range of the data affects training, and thus develop a preprocessing procedure if necessary, we have built a toy problem. We chose to focus on the left half of Base II and extract M = 6,255 windrose patches of side $N_{\text{patch}} = 33$ from the available ones, on the Far volume. In Figure 3.12 you can see the distribution of input values for the selected windrose patches and the contributions given by the two classes.



(a) Histogram of input values of the 6,255 selected windrose patch.



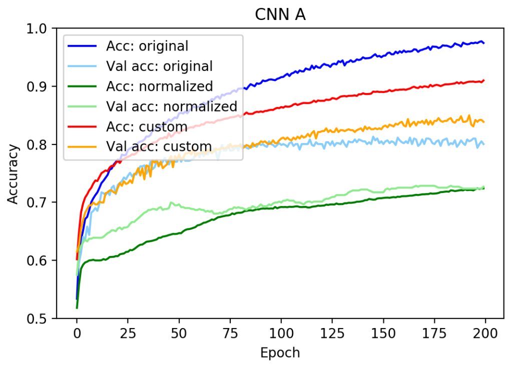
(b) Histogram of input values of the 6,255 selected windrose patch splitted by class.

Figure 3.12: Histogram of input values for the preprocessing toy problem.

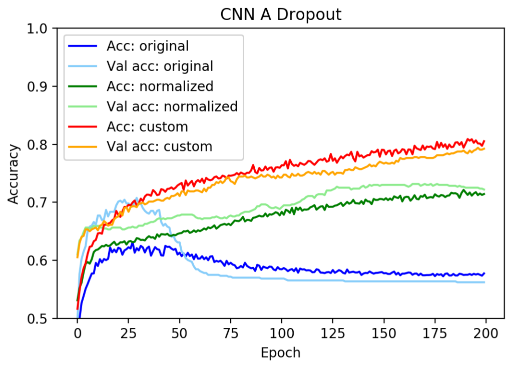
Since the seismic volumes contain physical information about the R_C reflection coefficients, we decided to study three preprocessing cases that would preserve the zero-crossing characteristic of the original data, i.e. a trivial rescaling dividing original data by a certain factor F_{proc}.

- **Original**: $F_{\text{proc}} = 1$. We keep the original data unchanged.
- **Normalized**: $F_{\text{proc}} = F_{\text{normalized}}$. We normalize the whole seismic volume, while preserving zero-crossing, dividing by $F_{\text{normalized,Vol}} = \max\{\mid \text{Vol}_{\text{min}} \mid, \mid \text{Vol}_{\text{max}} \mid\}$ where $\text{Vol} = \text{Far}, \text{Near}$.
- **Custom**: $F_{\text{proc}} = F_{\text{custom}}$. We adjust the normalized version by increasing data range by a factor of 100: $F_{\text{custom}} = F_{\text{normalized}}/100$.

We have therefore studied how these three preprocessing factors affect the training process applied to the architectures analyzed in this work. We have divided datasets of this toy problem into a training set (90%) and validation set (10%) to observe the robustness of the training process from the generalizing ability of the trained models.

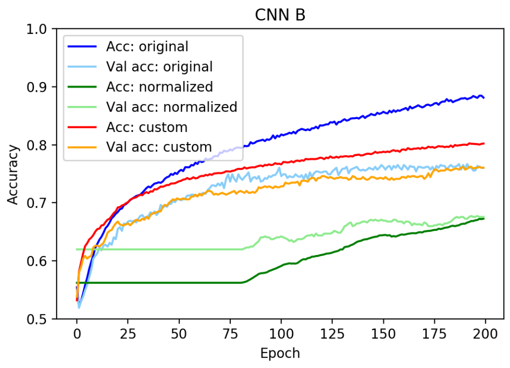


(a)

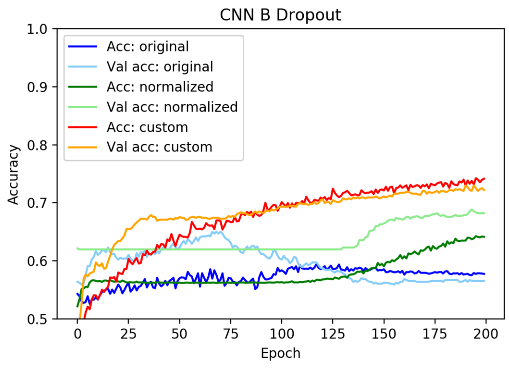


(b)

Figure 3.13: Effects of preprocessing on training CNN As architectures. 3.13a We can see that the normalized version behaves worse, probably the values are too small. It is very interesting to note that although the original version performs better on the training set, the custom version has the best generalizing ability. 3.13b The dropout seems to compromise the training on the original data while the custom version performs pretty well. We notice how the regularizing ability of the dropout keeps the performance on the training set and validation set close.



(a)



(b)

Figure 3.14: Effects of preprocessing on training CNN Bs architectures.

Considering the results expressed in Figures 3.13 and 3.14 and especially the good performance of the custom preprocessing version on Dropout architectures, we decided to use this preprocessing technique in all the simulations of this work.

3.1.4 Dataset Augmentation

Given the scarcity of labeled examples compared to the number of possible physically acceptable configurations in which 3D patches can be found. We decided to implement the dataset augmentation in order to improve the generalization of the models, providing artificially generated examples generated from real ones through transformations that preserve their realism.

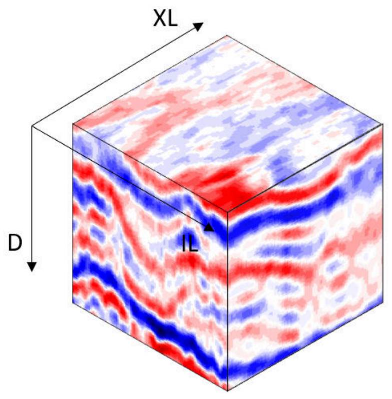
The chosen transformations are:

- **XL-flipping**: The 3D patch is mirrored on a plane passing through the center of the 3D patch and orthogonal to the XL direction.
- **IL-flipping**: The 3D patch is mirrored on a plane passing through the center of the 3D patch and orthogonal to the IL direction[6].
- **D-translation**: The 3D patch is shifted to increasing or decreasing depths of a certain number of voxels, respectively positive or negative. This transformation is important because horizons are the result of human interpretation and therefore not certain and not perfect. We account for this uncertain introducing a random noise in depth.
- **D-rotation**: The 3D patch is rotated with respect to the axis passing through the center of the 3D patch and parallel to D direction, by a certain angle. This transformation implies that there's not a preferential direction between XL and IL. Every object can be found orientated in all direction of the XL-IL plane.
- **Scaling**: The 3D patch is zoomed in or out by a certain factor, respectively larger than or smaller than one[7]. This transformation tries to account for the depth variation of seismic wavelet resolution (see Figure 2.10) and hence to help generalize on surrounding depths.

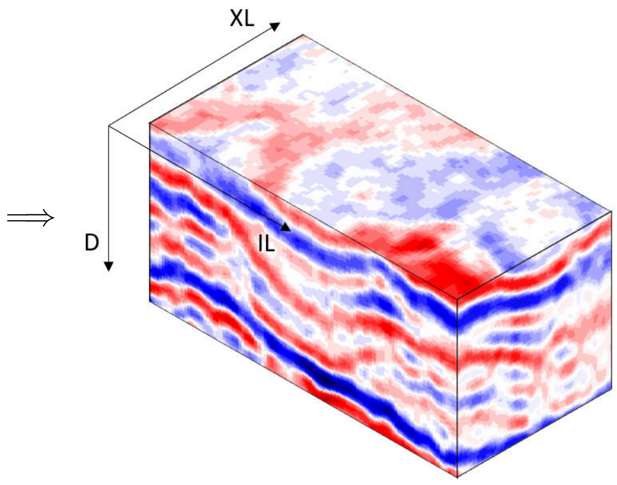
Thinking about the D-rotation transformation, we realized it hides a trap. In fact, the seismic volume sampling is not isotropic in the three directions, and especially XL and IL. This characteristic of the seismic volume implies that by rotating an original 3D patch you get another 3D patch that does not conform to the original volume due to this sampling asymmetry. For this reason, we decided to implement a pre-treatment procedure for original 3D patches in order to correctly perform dataset augmentation, as graphically described in Figure 3.15.

[6]You can't do the same thing with the D direction because it would produce unrealistic examples, a bit like seeing a tree upside down. The D direction is therefore substantially different from the other two.

[7]You may notice that in order to apply this transformation, the 3D patch must be larger than the final windrose patch; hence the request $N_{\text{saved}} > N_{\text{patch}}$. The same goes for the D-translation transformation.



(a)



(b)

Figure 3.15: XL-IL homogenization. 3.15a The original 3D patch extracted from seismic volume. As you can see in the IL axis there's more information respect to XL, this is caused by difference in axes sampling rates. 3.15b We expanded IL axis by a factor $S_{IL} = 2$, homogenizing XL and IL axes.

Once the XL-IL homogenized patch is produced, we proceed to extract the central region of side N_{patch} from which the windrose patch is generated, as shown in Figure 3.16 and Figure 3.17.

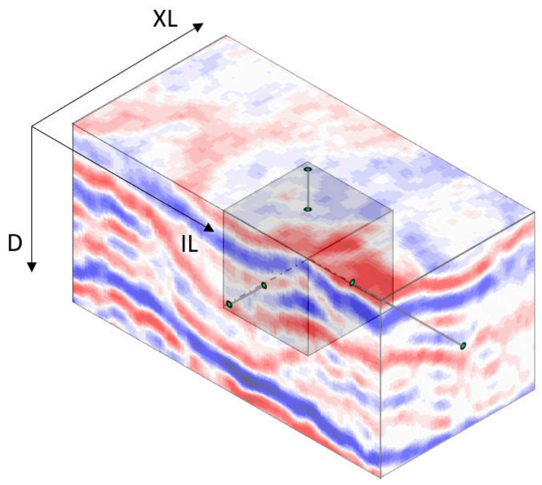


Figure 3.16: N_{patch} central cropping of the axis-homogenized 3D patch.

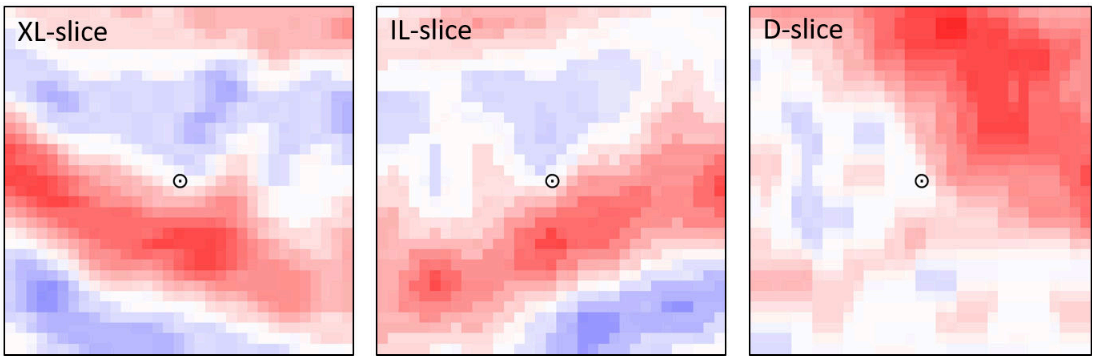


Figure 3.17: Original windrose patch of size $N_{\text{patch}} = 33$.

XL-Flipping

Flipping is a reflection transformation and cannot be described as a rotation, so it is necessary. Transformations are performed on the XL-IL homogenized 3D patch, as visible in Figure 3.18, from which it is then extracted a cropped sub-volume of side N_{patch}. In Figure 3.19 we show the XL-flipping data-augmented windrose patch.

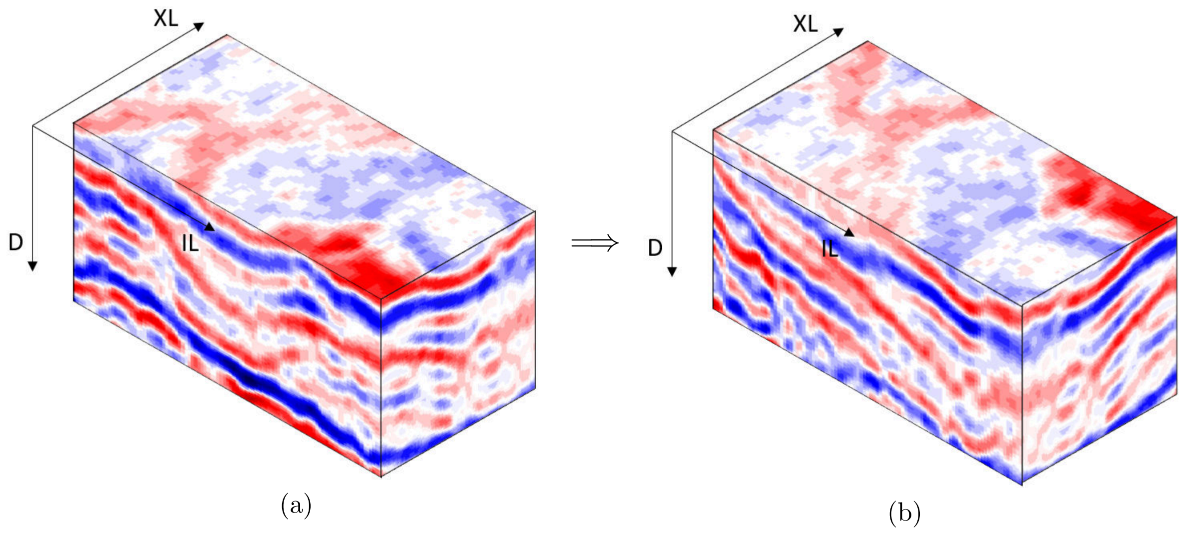


Figure 3.18: Dataset augmentation: XL-flip.

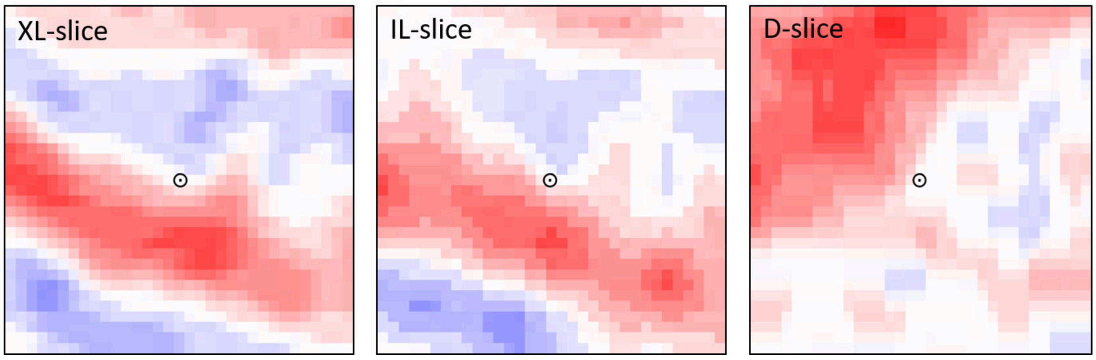


Figure 3.19: XL-flipping data-augmented windrose patch of size $N_{\text{patch}} = 33$.

IL-Flipping

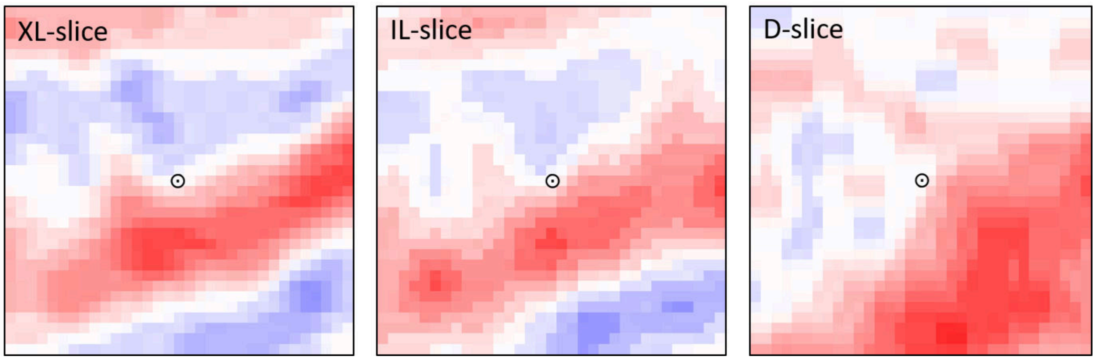


Figure 3.20: IL-flipping data-augmented windrose patch of size $N_{\text{patch}} = 33$.

D-Translation

This trasformation translates the 3D patch in depth of a random number of voxel within the range $[-5, +5]$.

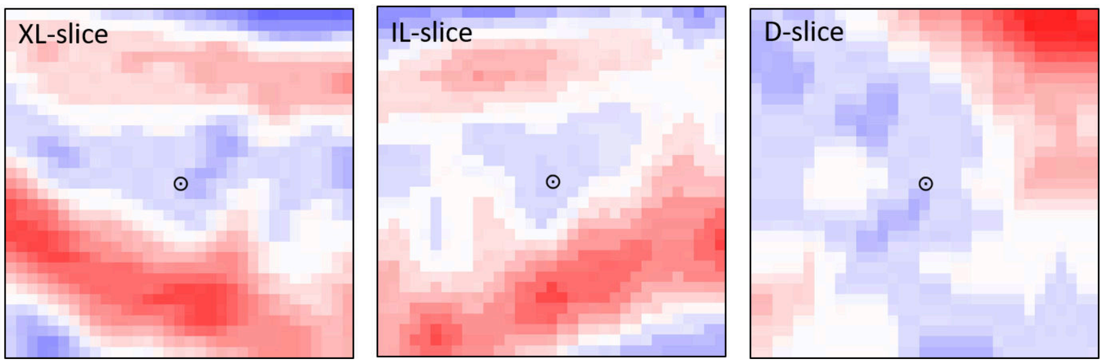


Figure 3.21: D-translation data-augmented windrose patch of size $N_{\text{patch}} = 33$. This windrose patch is associated to a translation of +5 voxel in depth.

D-Rotation

This transformation rotates the 3D patch of a random angle within the range $[-180°, +180°]$, with respect to the axis passing through the center and parallel to D.

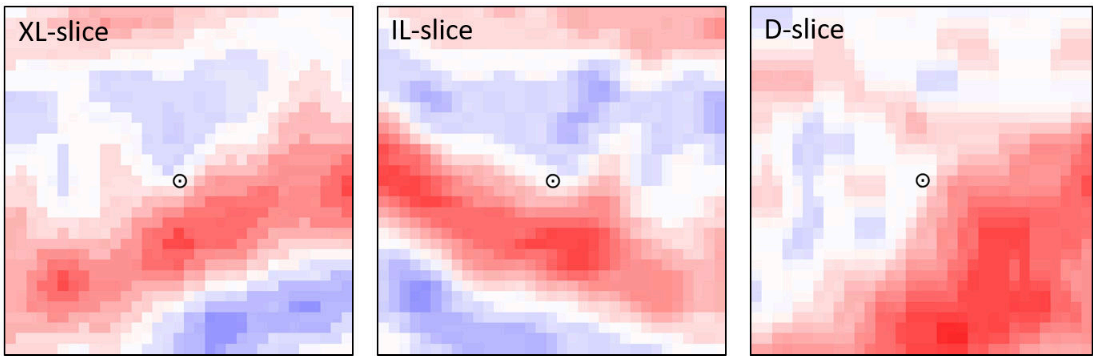


Figure 3.22: D-rotation data-augmented windrose patch of size $N_{\text{patch}} = 33$. This windrose patch is associated to a 90 degree anticlockwise rotation.

Scaling

This transformation zooms out or in by a random factor within the range $[0.9, 1.1]$, or equivalently $[-10\%, +10\%]$.

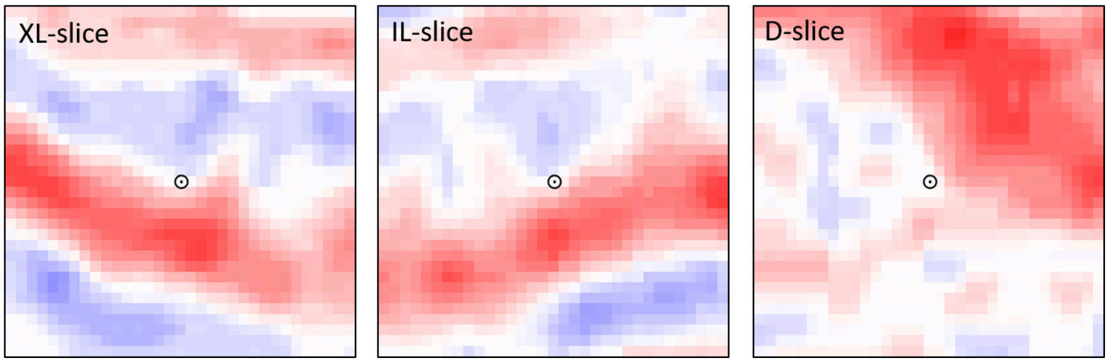


Figure 3.23: Scaling data-augmented windrose patch of size $N_{\text{patch}} = 33$. This windrose patch is associated to a scaling factor of 0.9, or equivalently to a zoom out of 10%.

Operatively, we implemented the dataset augmentation by sequentially applying all the selected transformations, each with a certain probability[8] $P_{\text{transformation}}$. So that a varied and representative dataset can be obtained without explicitly encoding every combination.

[8]We implemented it in such a way that the original windrose patch is always kept. Therefore we can also use unit-valued probabilities.

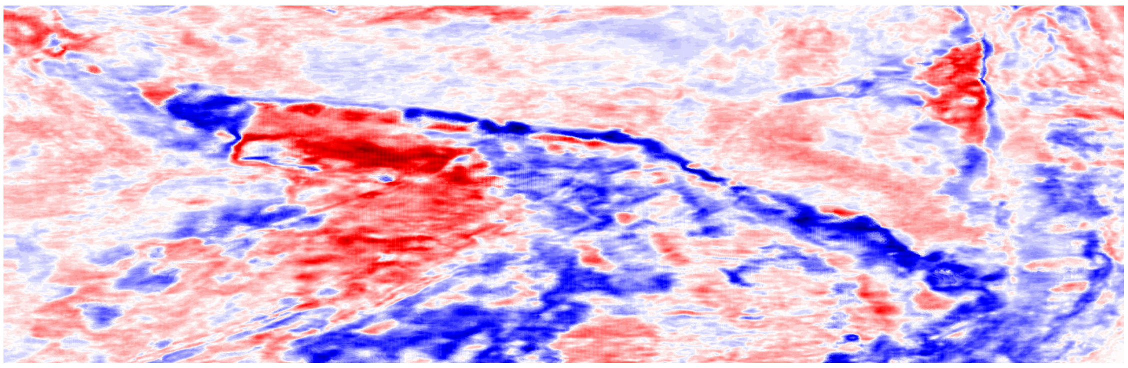
3.2 Training on Base II

In this section, we describe the results of the simulations carried out by training the models on portions of Base II horizon and by studying the predictive behavior of the models on previously unseen ones.
In this results presentation, we follow the logical path of reasoning that has guided us in our research. By exploring the key critical points identified, in particular:

- On which seismic volume is it better to set up the DL task[9]?
- Is dataset augmentation useful for this DL approach?
- How susceptible is this DL approach to channel-base segmentation?
- How and how much does the context extension N_{patch} affects DL performance?

To better understand the problem we are going to study and the differences between the two seismic volumes, we can observe the following two images which show the seismics values of Base II horizon voxels.

Observing Figures 3.24 and 3.25 we notice two important facts: the difference in resolution between the two volumes and that this horizon does not perfectly follow a constant seismics value. On the horizon, in fact, seismics take both negative and positive values, contrary to what one might think. This depends on one hand to the fact that the horizon is the result of a semi-automatic manual interpretation and segmentation and on the other to the fact that the horizontal seismic bands representing strata are discontinuous so that the operator interpreting horizons extends these regions crossing zones of discontinuity and inconsistency. However, the volume from which the horizons are segmented (Near) is consistent with the established channel-base as this seems to be more or less defined by a positive (red) seismics. This is not found on the Far volume as proof that the horizon has been segmented using Near seismics.



[9]We studied the performance of the models on both Far and Near volumes, although we know that since the horizons have been segmented from the seismics of Near, the latter is the logically coherent seismic volume on which to base the DL approach outlined so far.

Figure 3.24: Far seismics on Base II horizon.

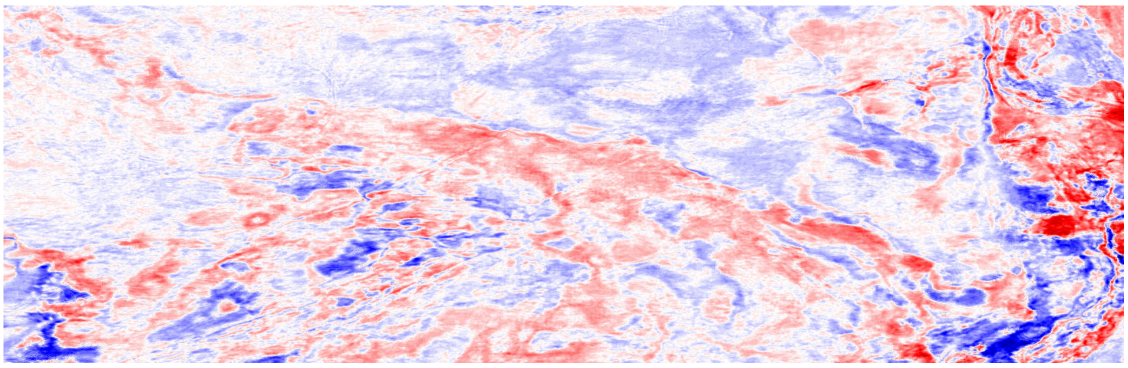


Figure 3.25: Near seismics on Base II horizon.

3.2.1 Left Training and Right Inference

We initially observed the importance of performing some form of dataset augmentation. For this purpose, we compared the results of two experiments conducted by training on the left half of the Base II horizon based on the Far volume.

The first experiment consists of training only on original examples while the second one consists of training on the same number of data-augmented examples.

	# Examples	
Label	Original	Data-augmented (x5)
Channel	13,133	2,626 x 5
Background	15,795	3,158 x 5
% Channel	45.40 %	45.40%
Tot	**28,928**	**28,920**

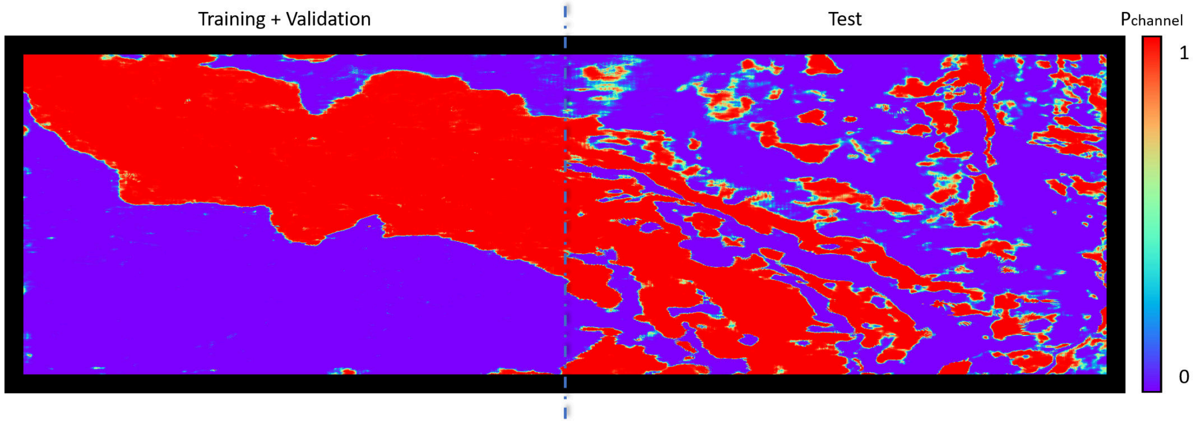
Table 3.1: Left training dataset composition. The symbol x5 is the data-augmentation factor which means that for each extracted original example 5 are produced (1 original example unchanged plus 4 artificial data-augmented examples). The examples are randomly selected from the M_{saved} ones located in the training area, i.e. the left half in this case.

In these experiments and all the following ones, we chose to divide the training data into an effective training set (90%) and a validation set (10%) to monitor the training process. Data augmentation features implemented in this experiment are expressed in Table 3.2. These parameters are used for all the simulations in this work so that results can be compared.

Transformation	$P_{\text{transformation}}$
D-rotation	1
XL-flipping	0.5
IL-flipping	0.5
D-traslation	0.5
Scaling	0.5

Table 3.2: Data augmentation probabilities. The D-rotation has an associated unit probability and this is allowed because we implemented that the original data is always kept unchanged by data augmentation. The order in which the transformations are expressed is the sequential order in which the data augmentation is implemented. Hence, first, we execute the rotation, then on the rotated data, we execute the XL-flipping with 0.5 probability, etc.

In the following figures we show how data-augmentation affects the generalization ability of the model here analyzed, CNN A on $N_{\text{patch}} = 65$ sided windrose patch[10]. The model is trained in both experiments for 200 epochs[11]. To better understand the results, we recommend checking the ground truth image valid for these experiments in Figure 3.10, remembering however that this is a segmentation made by the author and therefore should not be considered certain and unmodifiable.



[10]We chose to use the architecture without dropout to highlight the generalization power of data augmentation. Also, we chose windrose patches large enough to allow the models to overfit given the high representational capacity induced by such a big patch in order to reveal discrepancies between these two experiments.

[11]By epoch we mean a complete iteration of stochastic gradient descent. In other words, we can say that during an epoch the model sees as many examples as the total number of examples provided for training. The stochastic gradient descent, in fact, works on randomly sampled minibatch of M_B examples.

Figure 3.26: Left training and right inference: original examples.

In Figure 3.26 we notice that in the training region the model almost perfectly overfits the shape of the segmented channel, showing how the model representational capacity is high enough for this kind of problem. In the right half, i.e. the test half, we notice first of all that the model does not predict at random, this means that useful information is contained within the training set. However, we observe how the prediction has several high probability spots that are described in this segmentation as false-positives. We also notice that the learned channel-base is extended as pseudo-straight filaments that come out from the training boundaries and do not follow the correct downward meandering shape of ground truth. According to our interpretation, this is due to the lack of an enough representative dataset of all the possible spatial orientations, in the Xl-IL plane, where a channel-base can be found.

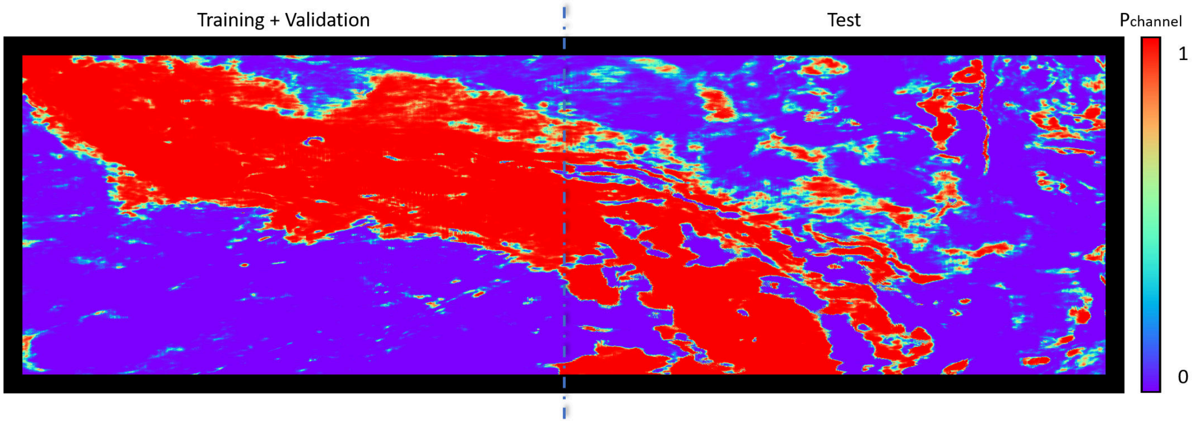


Figure 3.27: Left training and right inference: data-augmented examples.

In Figure 3.27 we notice that in the training region despite the spatially dilution[12] of training examples by a factor of 5, the shape of the segmented channel continues to be well represented. In the test region instead, we notice how the number of false-positive spots has decreased and especially how the channel is now better extended as the expected downward trend is here present. At the top right, we continue to see the presence of a high probability channel-base region that is consistent with the topological information inferable by examining Base II horizon in Figure 3.6. This might mean that the segmented channel-base is not the only one present in Base II horizon.

Given the results produced, we can argue that for the DL task described in this work, the data augmentation is of fundamental importance and cannot be omitted because the

[12]In Figure 3.26 the mean minimum distance, expressed in number of voxels, between central voxel of two training windrose patches is 4.22 while in Figure 3.27 is 9.43.

number of physically acceptable configurations that can be encountered far exceeds those that can be extracted from a single horizon. Therefore, for all the experiments carried out in this work, data augmentation has always been applied.

3.2.2 K-Fold-Cross Validation

Cross-validation is any of various similar model validation techniques for statistical assessing how the results of an algorithm will generalize to an independent data set. It is mainly used in settings where the goal is prediction, and one wants to estimate how accurately a predictive model will perform in practice. The goal of cross-validation is to test the model's ability to predict new data that was not used in estimating it, in order to flag problems like overfitting or selection bias[13] and to give an insight on how the model will generalize to an independent dataset.

Two types of cross-validation can be distinguished: exhaustive and non-exhaustive cross-validation. Exhaustive cross-validation methods are cross-validation methods which study models that are respectively trained and tested on all possible ways to divide the original dataset into a training and a test set. Non-exhaustive cross-validation methods do not compute all ways of splitting the original dataset. The most famous of this class of methods is K-fold cross-validation.

In K-fold cross-validation, the original dataset is randomly partitioned into K equal sized subsamples. Of the K subsamples, a single subsample is retained as the test dataset for testing the model, and the remaining K-1 subsamples are used as training the dataset. The cross-validation process is then repeated K times, with each of the K subsamples used exactly once as the test data, generating K different models. The accuracies of the K models can then be averaged to produce a single accuracy estimation.

In summary, K-fold cross-validation combines, or averages, measures of accuracy in prediction to derive a more accurate estimate of model prediction performance with associated uncertainty.

$$\begin{aligned} Acc_{\text{K-fold}} &= \frac{1}{K}\sum_{i=1}^{K} Acc_i \\ \sigma_{Acc,\text{K-fold}} &= \sqrt{\frac{\sum_{i=1}^{K}\left(Acc_i - Acc_{\text{K-fold}}\right)^2}{K}} \end{aligned} \tag{3.11}$$

[13]Selection bias is a distortion in a measure due to a sample selection that does not accurately reflect the overall target ensemble.

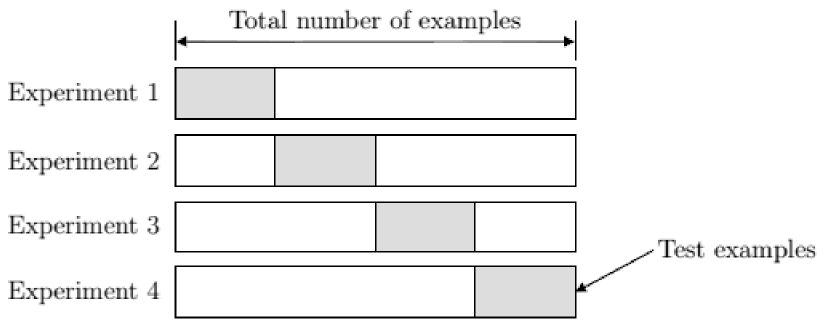


Figure 3.28: 4-fold cross-validation graphical representation. In this representation dataset examples are randomly placed in a list and therefore there is no concept of samples spatial location.

In our case, a random partition of the dataset is not the most statistically correct way to apply K-fold cross-validation. Windrose patches that are located close on the horizon have high correlation because they share part of their 3D context and this can distort accuracy estimates if K is small. We, therefore, decided to split the dataset according to a spatial criterion as in Figure 3.29.

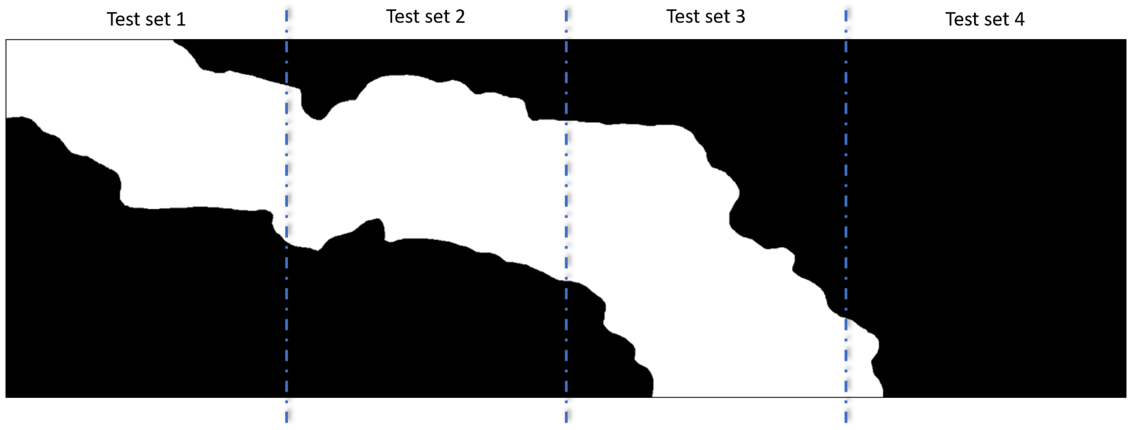


Figure 3.29: 4-fold cross-validation subsamples on Base II first segmentation.

The results obtained from our simulations are shown below. All the prediction images refer to the CNN A Dropout architecture trained for 1,000 epochs.

3.2.2.1 Small Context Extension

Initially we decided to study simpler cases so we set to a low value the extension of the 3D context associated with each voxel through the input patch. In particular we selected $N_{\text{patch}} = 33$.

	# Channel voxel	# Background voxel	% Channel voxel
Test set 1	105,752	182,704	36.66 %
Test set 2	122,627	165,223	42.60 %
Test set 3	137,526	150,324	47.78 %
Test set 4	6,309	281,541	2.19 %

Table 3.3: 4-fold cross-validation subsamples on Base II first segmentation composition. This table is relative to Figure 3.29. As you can see the test set 4 has a class disproportion compared to the other test sets and this is a critical point of this subsampling. We opted for this solution because it allows us to investigate via experiment 4 if and how the information contained therein we have high confidence in segmentation is reflected in the leftmost margin where we have some doubt about the presence of a second channel-base. However, in order to obtain better statistical results, we suggest implementing different subsampling in the future.

First Version Segmentation

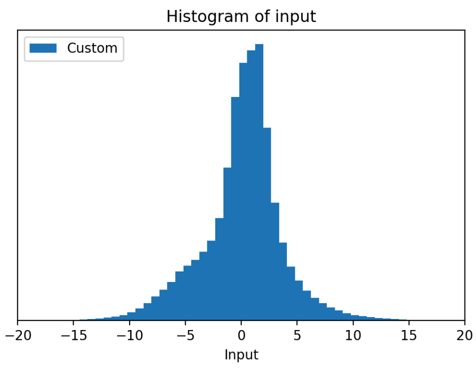
	# Examples		
	Channel	Background	% Channel
Test set 1	1,164 x 10	1,855 x 10	38.56 %
Test set 2	1,538 x 10	1,698 x 10	47.53 %
Test set 3	1,753 x 10	1,483 x 10	54.17 %
Test set 4	54 x 10	2,964 x 10	1.79 %
Tot	**125,900**		

Table 3.4: $N_{\text{patch}} = 33$ Base II first segmentation dataset composition. For a graphical representation of Base II first segmentation see Figure 3.29.

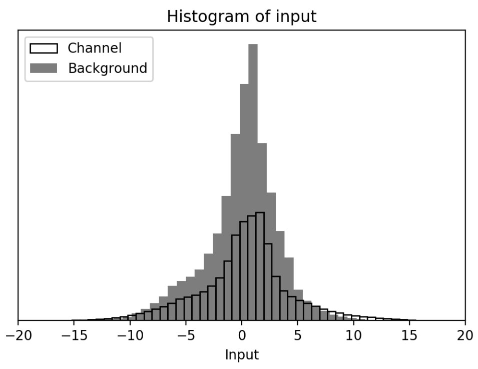
We analyzed the problem set on both Far and Near volume.

Far Volume

Here we report the result of the 4-fold cross-validation. For a clearer explanation we now also show the result of experiment 4.



(a) Histogram of preprocessed input values.



(b) Histogram of preprocessed input values splitted by class.

Figure 3.30: Histogram of preprocessed input values for the Far volume $N_{\text{patch}} = 33$ Base II first segmentation simulation.

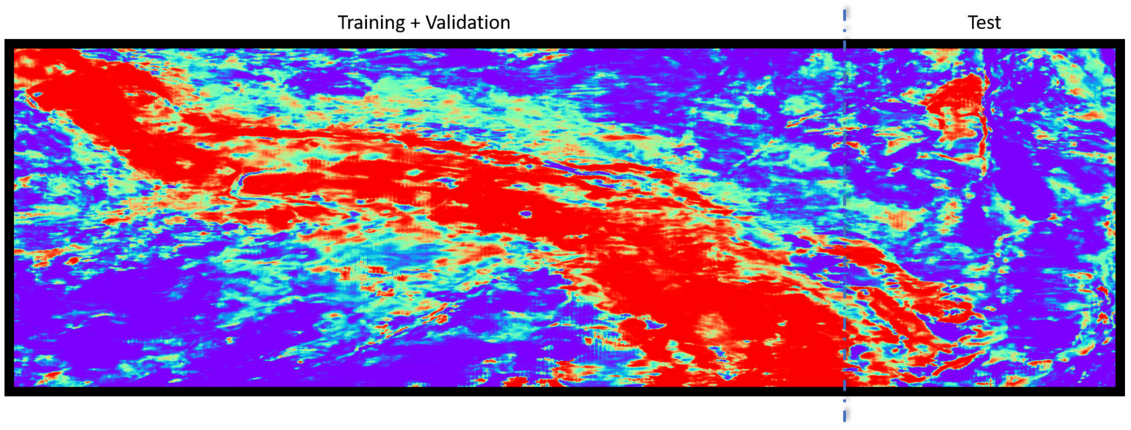


Figure 3.31: Far volume $N_{\text{patch}} = 33$ Base II first segmentation: experiment 4.

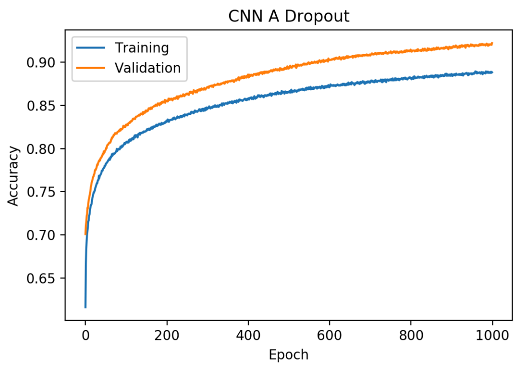


Figure 3.32: Far volume $N_{\text{patch}} = 33$ Base II first segmentation: accuracy evolution of experiment 4.

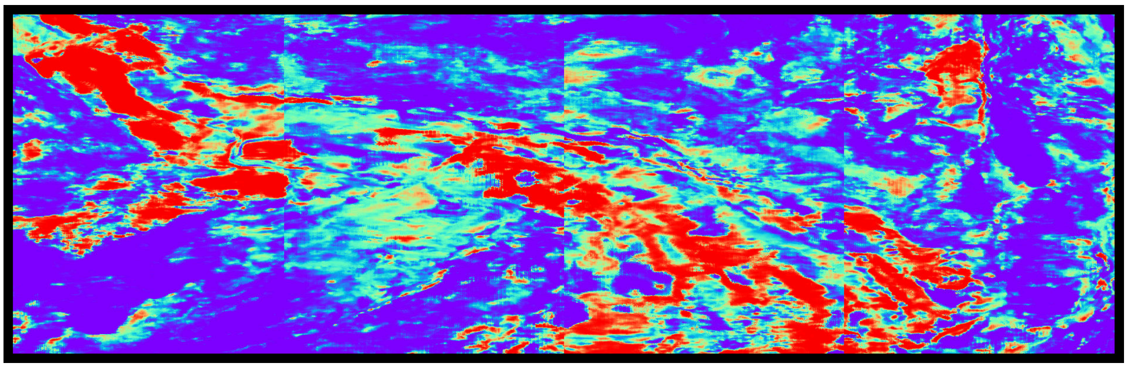
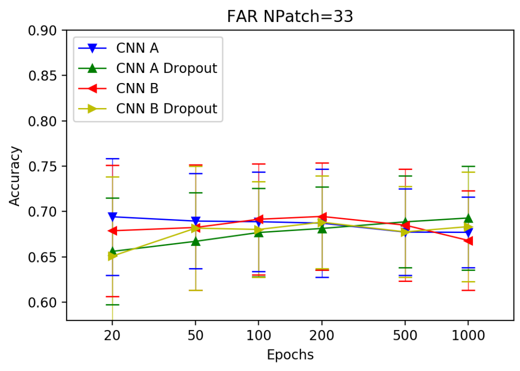


Figure 3.33: Far volume $N_{\text{patch}} = 33$ Base II first segmentation: prediction image.

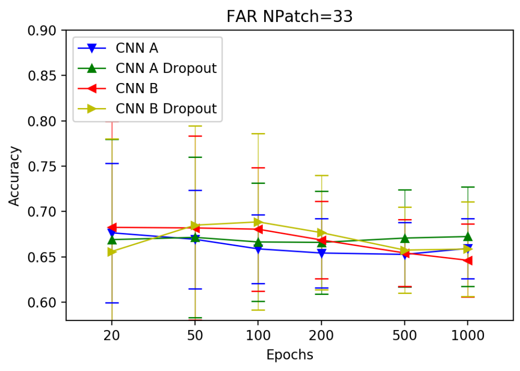
Figure 3.33 is obtained by composing the inference images of the four models defined by the 4-fold cross-validation on the respective test sets. Therefore, the accuracy calculated on this image is the same as $Acc_{\text{K-fold}}$. By studying this image we see some interesting facts. We observe that not considering test set 4 which has already pointed out critical issues, in test set 2 the DL task seems to be more difficult than elsewhere. We think that the cause of this lies in the non-coherent application of the Base II horizon on Far seismics, in fact, as previously pointed out, the horizons have been segmented according to Near seismics. Furthermore, we observe how in test set 1 there seems to be a channel-base confluence/bifurcation. From the topological information we have, there are no channel-bases in that area. However, it is possible that because the context of the windrose patches also extends in the depth direction and because through training on the horizon the model is not well trained on class discontinuities in depth, this channel-base is either shallower or deeper than the training horizon.

Since the classes within each test set are neither balanced nor in equal proportion across different test sets, as seen in Table 3.4, we decided to also calculate the weighted accuracy $WAcc$ as defined in Equation 3.12. Unweighted accuracy is simply defined as the ratio between the number of correctly predicted examples, true positive (True Channel-base, TC) and true negative (True Background, TB), and the total number of tested examples. The weighted accuracy version, on the other hand, consists of the average of compartmentalized accuracy on each class, thus balancing the prediction's contribution even on sparsely populated classes.

$$\begin{aligned} UAcc_i &= \frac{T\mathrm{C}_i + T\mathrm{B}_i}{\#\mathrm{C}_i + \#\mathrm{B}_i} \\ WAcc_i &= \frac{1}{2}\left(\frac{T\mathrm{C}_i}{\#\mathrm{C}_i} + \frac{T\mathrm{B}_i}{\#\mathrm{B}_i}\right) \end{aligned} \qquad i = 1, 2, \ldots, K \tag{3.12}$$



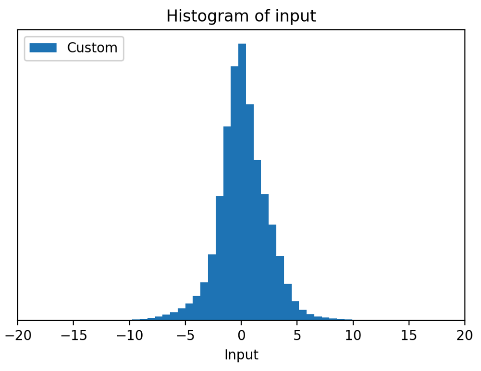
(a) Unweighted accuracy $UAcc_{\text{K-fold}}$.



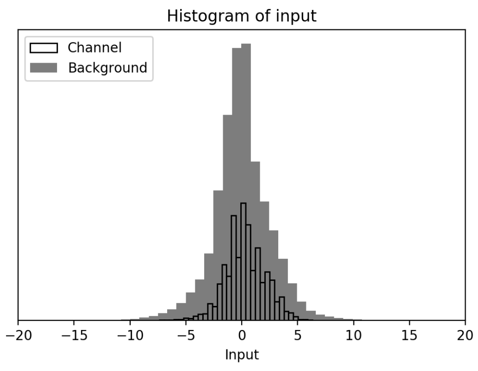
(b) Weighted accuracy $WAcc_{\text{K-fold}}$.

Figure 3.34: Far volume $N_{\text{patch}} = 33$ Base II first segmentation: accuracies. We notice that accuracies do not seem to increase in a statistically significant way as the number of training epochs increases. This is a strong indication that in this configuration the problem is not well-posed. Another important aspect that emerges is that the standard deviation associated with unweighted accuracy does not decrease in a statistically significant way as the number of epochs increases. This means that the K weighted accuracies are converging toward the same value as training proceeds.

Near Volume



(a) Histogram of preprocessed input values.



(b) Histogram of preprocessed input values splitted by class.

Figure 3.35: Histogram of preprocessed input values for the Near volume $N_{\text{patch}} = 33$ Base II first segmentation simulation. We point out that compared to the histogram of the dataset extracted on the Far seismics in Figure 3.30, here the distribution of the inputs associated with the Channel-base class seems to be more symmetric around zero. This is a rough indication that there is a profound difference in the application of the DL task on the two seismic volumes.

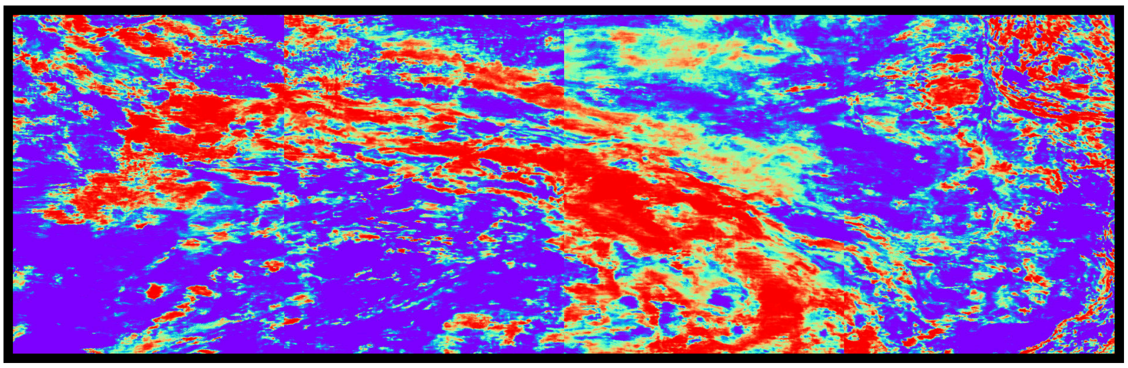
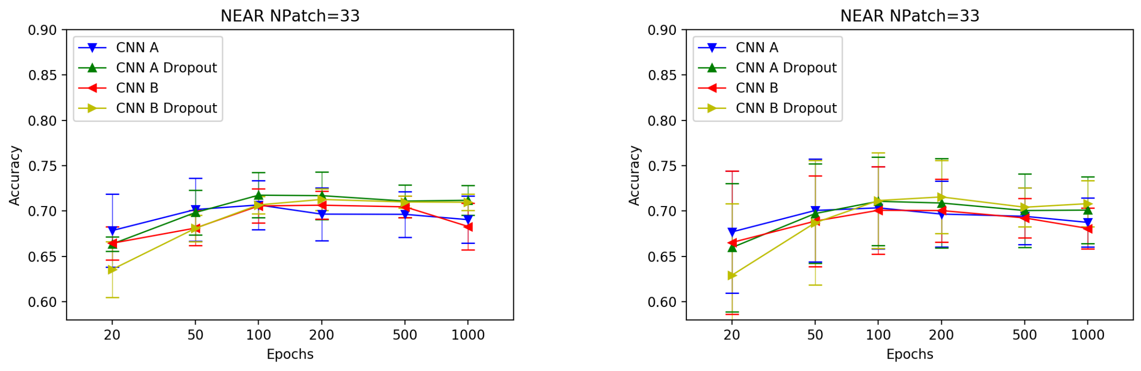


Figure 3.36: Near volume $N_{\text{patch}} = 33$ Base II first segmentation: prediction image.

In Figure 3.36 we initially observe how this image looks more resolute than Figure 3.33, this is directly related to the higher resolution of Near seismics. We notice that in the upper portion of test set 4 there is a predicted channel-base system that might be compatible with the horizon topology in Figure 3.6, but that has been considered as Background in this segmentation. We observe that there is a channel-base that consistently crosses test sets 1 and 2 and runs parallel at the top of the segmented channel-base. We consider this to be a false positive and we think, looking more closely at the segmentation in Figures 3.29, that this false positive may be due to an excessively broad classification of channel-bases that includes excessive channel banks.



(a) Unweighted accuracy $UAcc_{\text{K-fold}}$. (b) Weighted accuracy $WAcc_{\text{K-fold}}$.

Figure 3.37: Near volume $N_{\text{patch}} = 33$ Base II first segmentation: accuracies. We observe that in this case, accuracies show an initial growth trend that stabilizes around 100 epochs at an approximate value of 70 %. The accuracy value on its own brings little information because the channel-base segmentation is not considered certain, so what is actually significant is this growth trend as epochs progress. This is a further indication that the DL task should be set on Near seismics.

Second Version Segmentation

From considerations developed on Figures 3.36 about the effects of an excessive segmentation, we decided to study how sensitive this DL approach is to segmentation. In particular, we decided to slightly modify the first segmentation version of Base II excluding what we believe to be critical portions of channel banks that lead models to predict false positives.

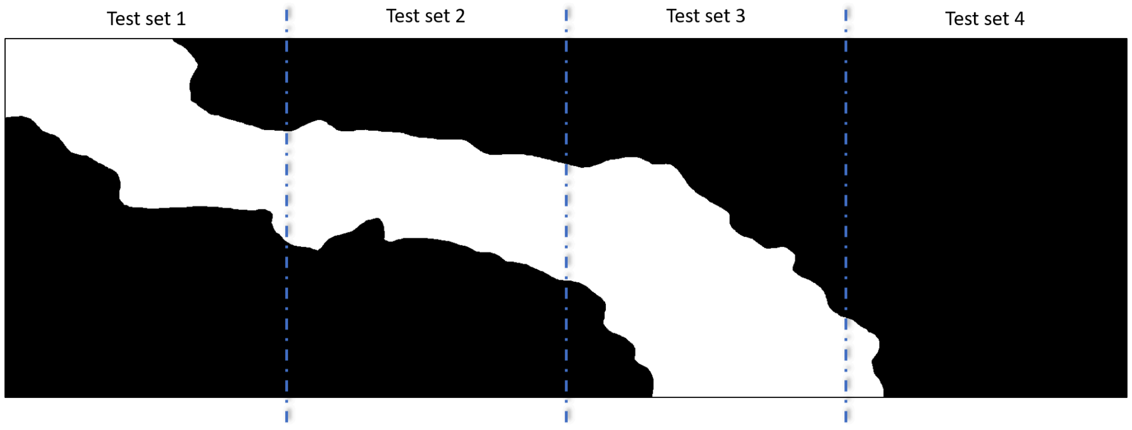


Figure 3.38: 4-fold cross-validation subsamples on Base II second segmentation.

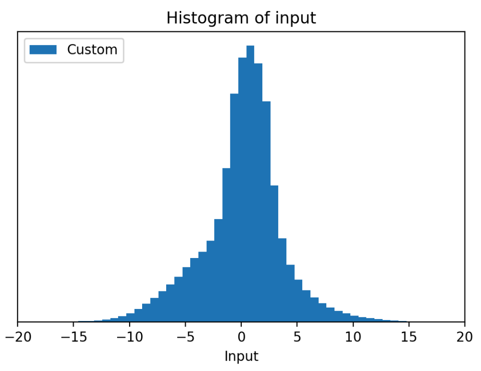
	# Channel voxel	# Background voxel	% Channel voxel
Test set 1	93.043	195.413	32,26 %
Test set 2	85.732	202.118	29,78 %
Test set 3	130.181	157.669	45,23 %
Test set 4	6.309	281.541	2,19 %

Table 3.5: 4-fold cross-validation subsamples on Base II second segmentation composition.

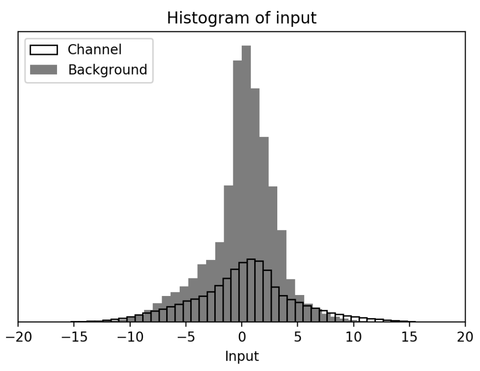
	# Examples		
	Channel	Background	% Channel
Test set 1	1,006 x 10	2,013 x 10	33.32 %
Test set 2	1,075 x 10	2,161 x 10	33.22 %
Test set 3	1,500 x 10	1,736 x 10	46.36 %
Test set 4	54 x 10	2,964 x 10	1.79 %
Tot	**125,900**		

Table 3.6: $N_{\text{patch}} = 33$ Base II second segmentation dataset composition.

Far Volume



(a) Histogram of preprocessed input values.



(b) Histogram of preprocessed input values splitted by class.

Figure 3.39: Histogram of preprocessed input values for the Far volume $N_{\text{patch}} = 33$ Base II second segmentation simulation.

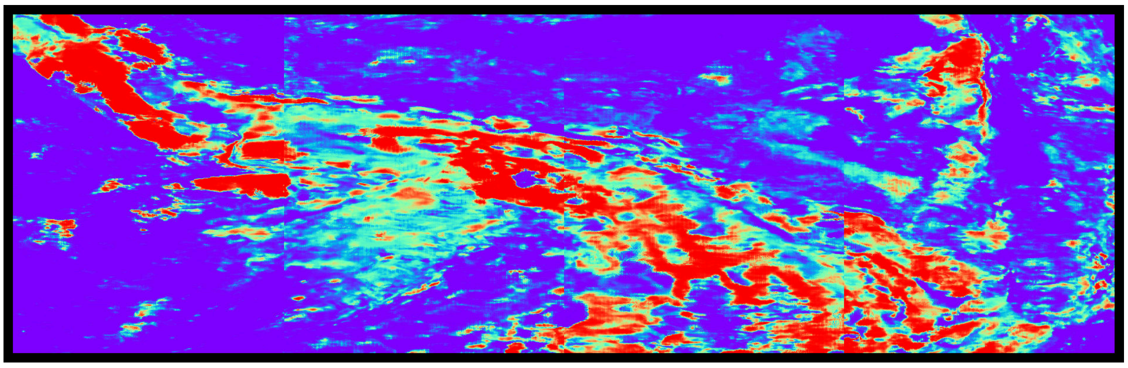
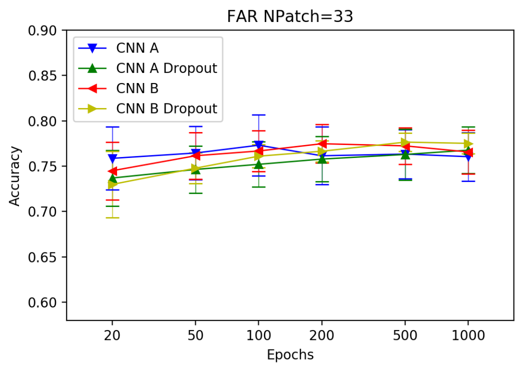
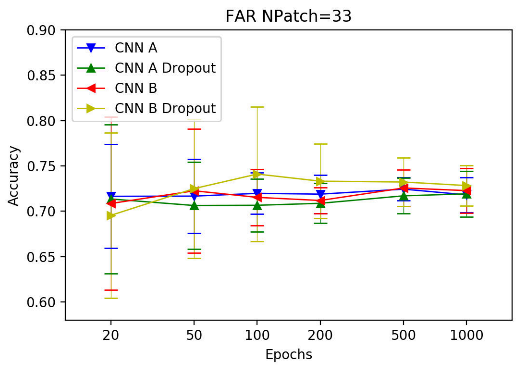


Figure 3.40: Far volume $N_{\text{patch}} = 33$ Base II second segmentation: prediction image.

In Figure 3.40 we observe how the image looks cleaner compared to Figure 3.33 since many false positives (FC) are no longer present. We notice that the prediction in test set 4 is substantially unchanged. We also notice that in test set 2 the problem is still more difficult than in the other test sets, although it is better solved. Finally, we observe that the new segmentation allowed the model in test set 1 to remove the false positive channel-base that seemed to merge/fork the segmented channel-base. From this visual result, we can say that segmentation plays a fundamental role and even small changes have a great impact on the behavior of DL models.



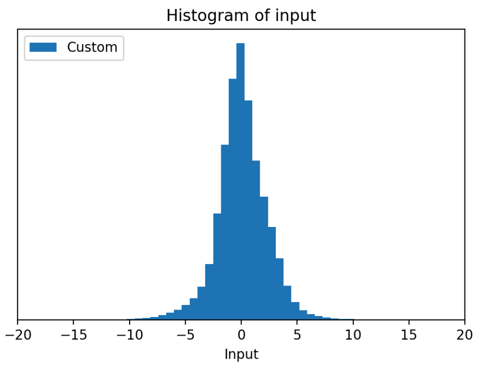
(a) Unweighted accuracy $UAcc_{\text{K-fold}}$.



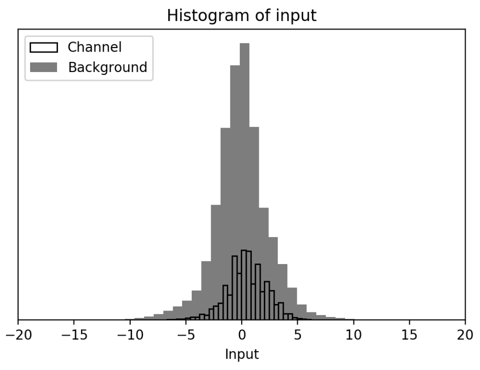
(b) Weighted accuracy $WAcc_{\text{K-fold}}$.

Figure 3.41: Far volume $N_{\text{patch}} = 33$ Base II second segmentation: accuracies. We observe that unweighted accuracies with this new segmentation are higher than accuracies in Figure 3.34 and also than weighted accuracies. The weighted accuracies instead remain statistically unchanged with respect to the first version segmentation case. However, the first point is only an indication that the problem induced by the new segmentation is more easily representable through the representational capacity of the studied models. And the fact that the unweighted accuracies are statistically higher than weighted ones means that although the number of true negative (TB) has increased, the number of true positive (TC), which in this kind of metrics counts more, has decreased. These two effects balance each other perfectly keeping the weighted accuracy unchanged compared to the case with the first segmentation.

Near Volume



(a) Histogram of preprocessed input values.



(b) Histogram of preprocessed input values splitted by class.

Figure 3.42: Histogram of preprocessed input values for the Near volume $N_{\text{patch}} = 33$ Base II second segmentation simulation.

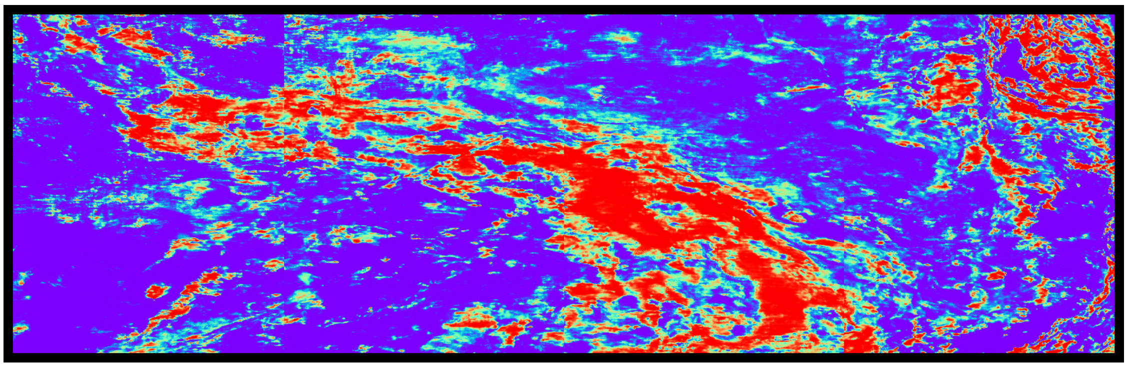
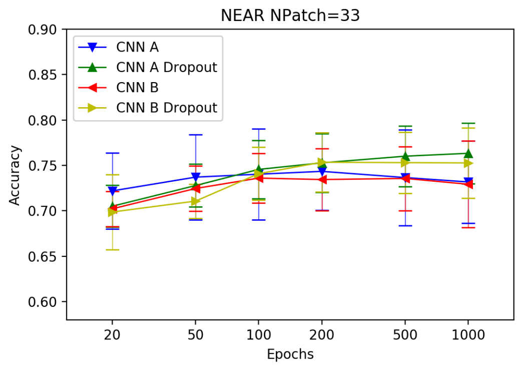
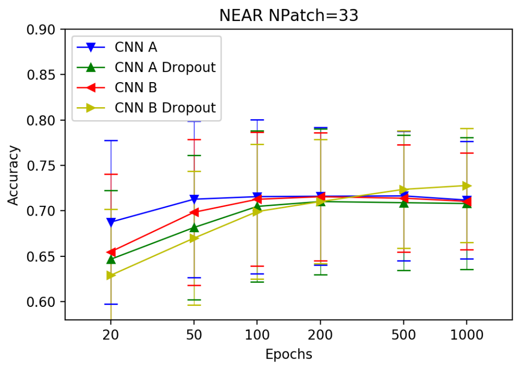


Figure 3.43: Near volume $N_{\text{patch}} = 33$ Base II second segmentation: prediction image.

For Figure 3.43 similar considerations hold true to those expressed for Figure 3.40. We observe how a small variation in the segmentation has a significant effect on the predictive behavior of the models. Therefore we can conclude that for this DL approach segmentation plays a determinant role and thus proper effort and care must be invested in this process.



(a) Unweighted accuracy $UAcc_{\text{K-fold}}$.



(b) Weighted accuracy $WAcc_{\text{K-fold}}$.

Figure 3.44: Near volume $N_{\text{patch}} = 33$ Base II second segmentation: accuracies. We observe how in the case of unweighted accuracies dropout architectures show a better generalizing ability as the associated accuracies continue to increase even beyond 100 epochs, where instead the other architectures seem to saturate.

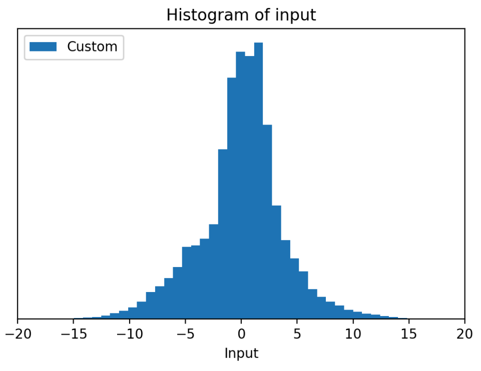
3.2.2.2 Big Context Extension

Lastly, we explored how the extension of input patches affects the performance of algorithms. We wondered if a windrose patch of side Npatch $= 33$ would provide enough context to models for this DL task. For this purpose we studied the case Npatch $= 65$, and to allow comparability with previous simulations we kept the same number of examples. We are well aware however of the danger represented by the curse of dimensionality. This aspect is surely a critical point of these experiments.

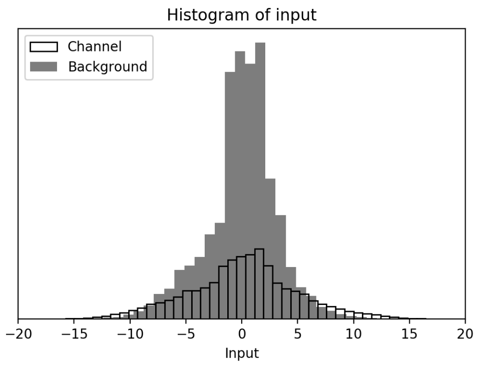
	# Examples		
	Channel	Background	% Channel
Test set 1	930 x 10	1,809 x 10	33.95 %
Test set 2	1,075 x 10	1,970 x 10	35.30 %
Test set 3	1,434 x 10	1,612 x 10	47.08 %
Test set 4	45 x 10	2,694 x 10	1.64 %
Tot	**125,900**		

Table 3.7: $N_{\text{patch}} = 65$ Base II second segmentation dataset composition.

Far Volume



(a) Histogram of preprocessed input values.



(b) Histogram of preprocessed input values splitted by class.

Figure 3.45: Histogram of preprocessed input values for the Far volume $N_{\text{patch}} = 65$ Base II second segmentation simulation.

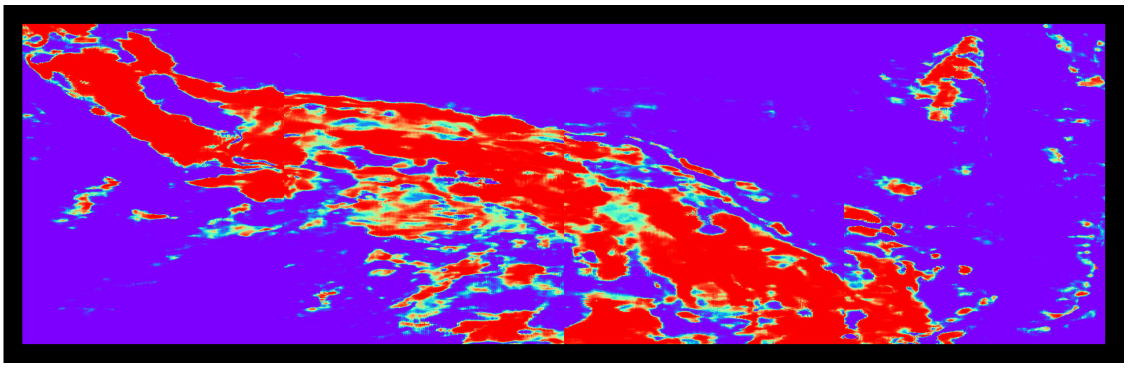
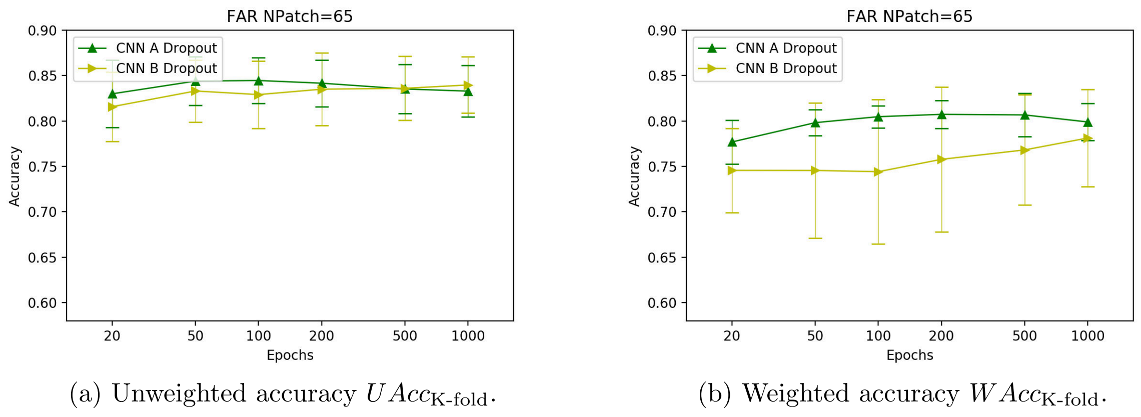


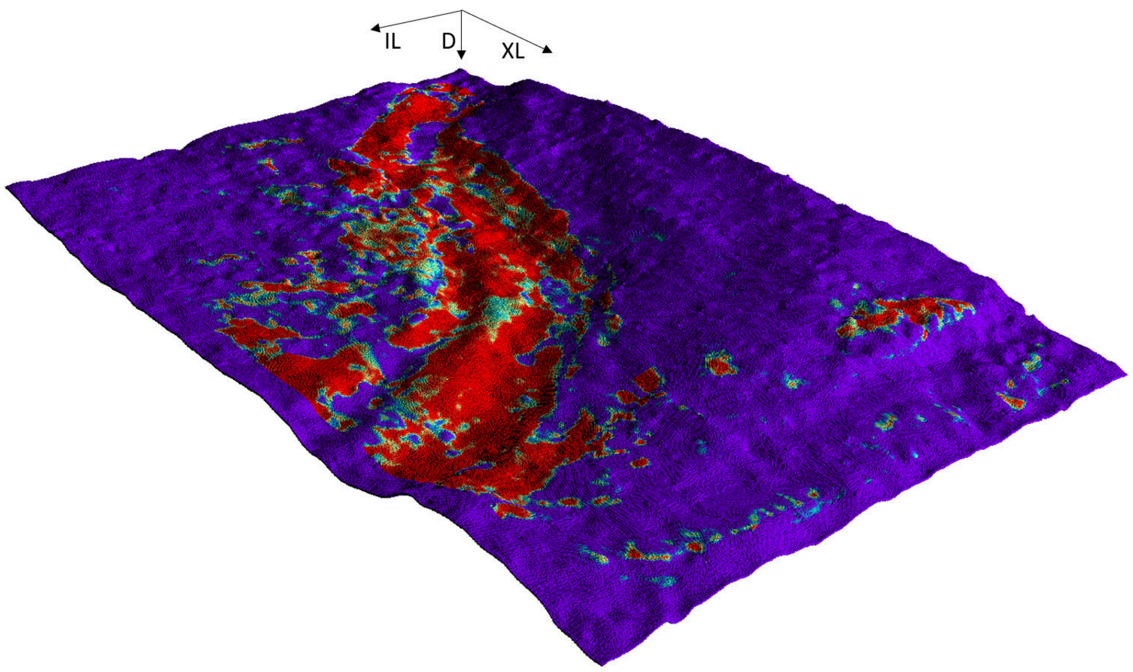
Figure 3.46: Far volume $N_{\text{patch}} = 65$ Base II second segmentation: prediction image.

In Figure 3.46 we observe how the visual quality of the prediction has increased. In fact, the many false-positive FC spots in Figure 3.36 are almost completely gone. The increased context provided to the models also made possible a better solution to the problem in test set 1.

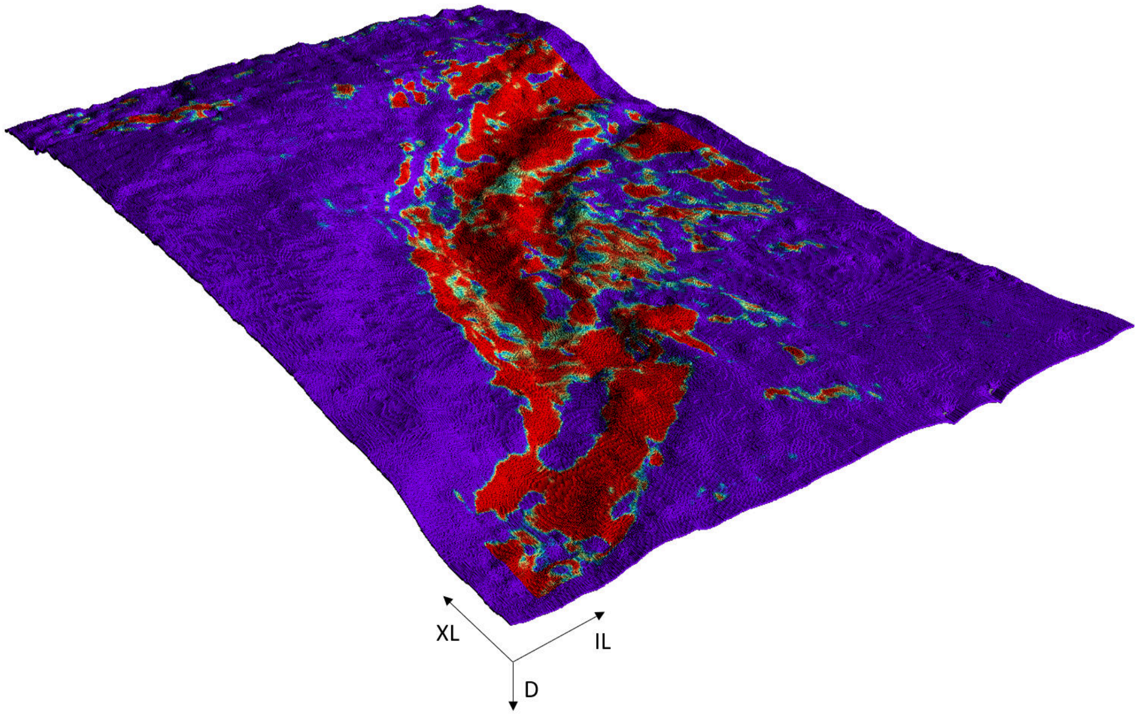


(a) Unweighted accuracy $UAcc_{\text{K-fold}}$. (b) Weighted accuracy $WAcc_{\text{K-fold}}$.

Figure 3.47: Far volume $N_{\text{patch}} = 65$ Base II second segmentation: accuracies. Although the number of tunable parameters has significantly increased for CNN As architectures, see Table 2.1, and the number of examples provided is minimal in this respect, CNN A Dropout seems to perform very well at least here working on Far seismics. An interesting fact that can be seen by observing the weighted accuracies of the CNN A Dropout architecture is that the associated standard deviations are sensibly smaller than those of the CNN B Dropout architecture accuracies.



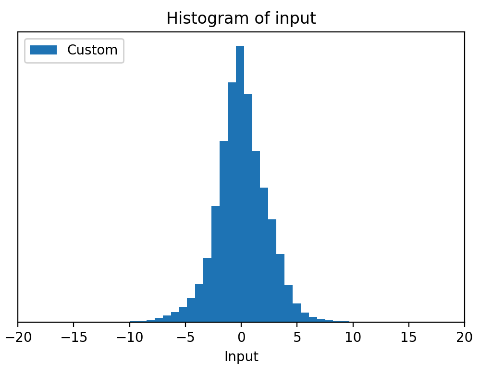
(a) First 3D view.



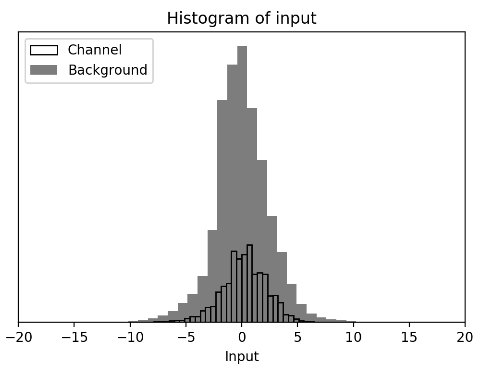
(b) Second 3D view

Figure 3.48: Far volume $N_{\text{patch}} = 65$ Base II second segmentation: 3D view prediction image.

Near Volume



(a) Histogram of preprocessed input values.



(b) Histogram of preprocessed input values splitted by class.

Figure 3.49: Histogram of preprocessed input values for the Near volume $N_{\text{patch}} = 65$ Base II second segmentation simulation.

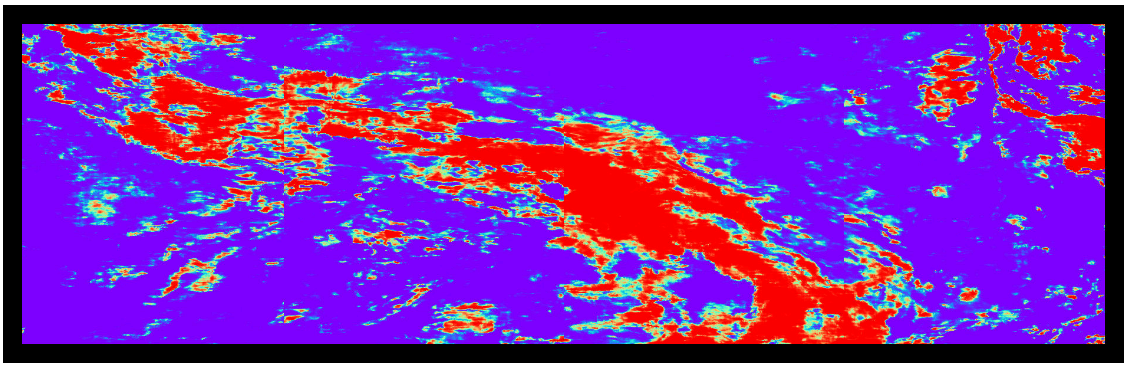
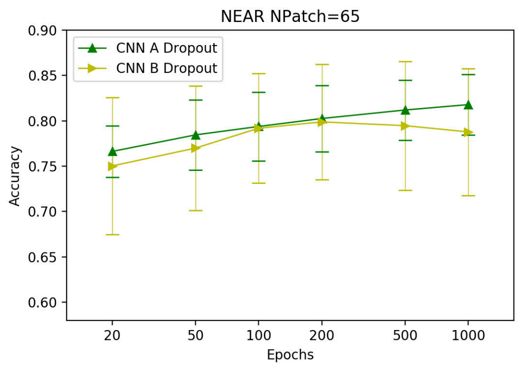
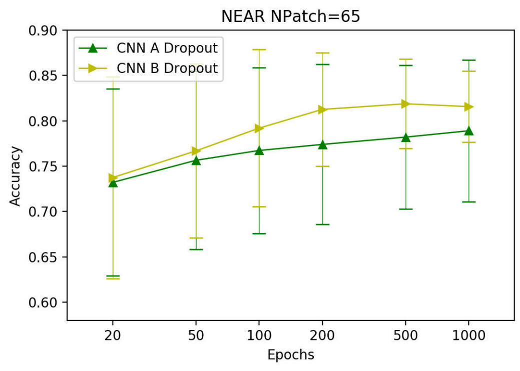


Figure 3.50: Near volume $N_{\text{patch}} = 65$ Base II second segmentation: prediction image.

For Figure 3.50 similar considerations hold true to those expressed for Figure 3.46.

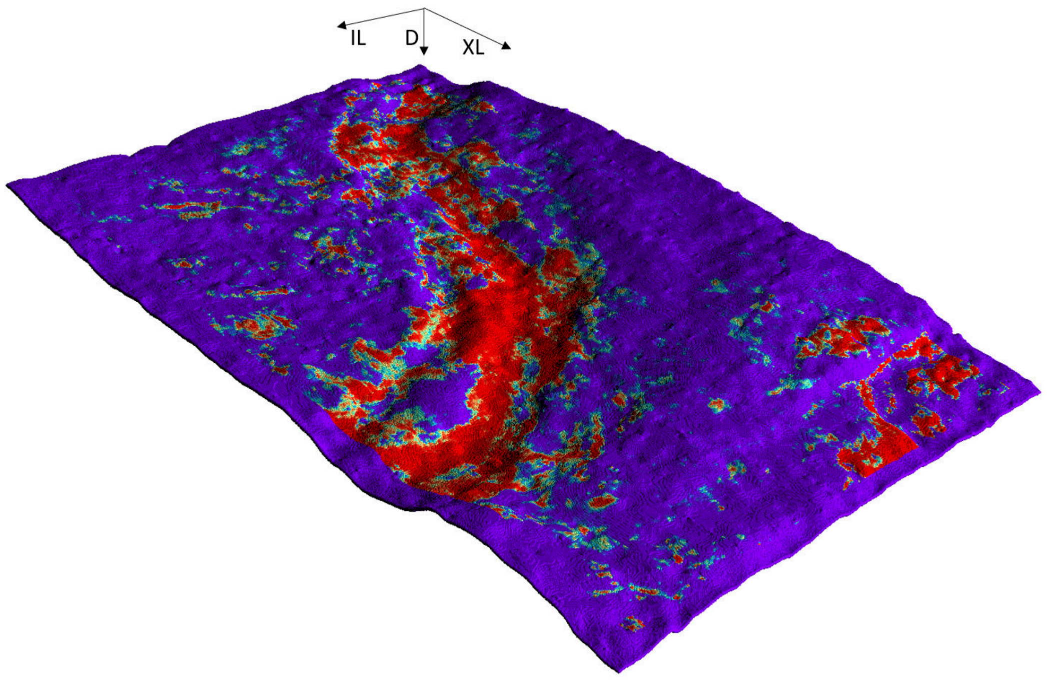


(a) Unweighted accuracy $UAcc_{\text{K-fold}}$.

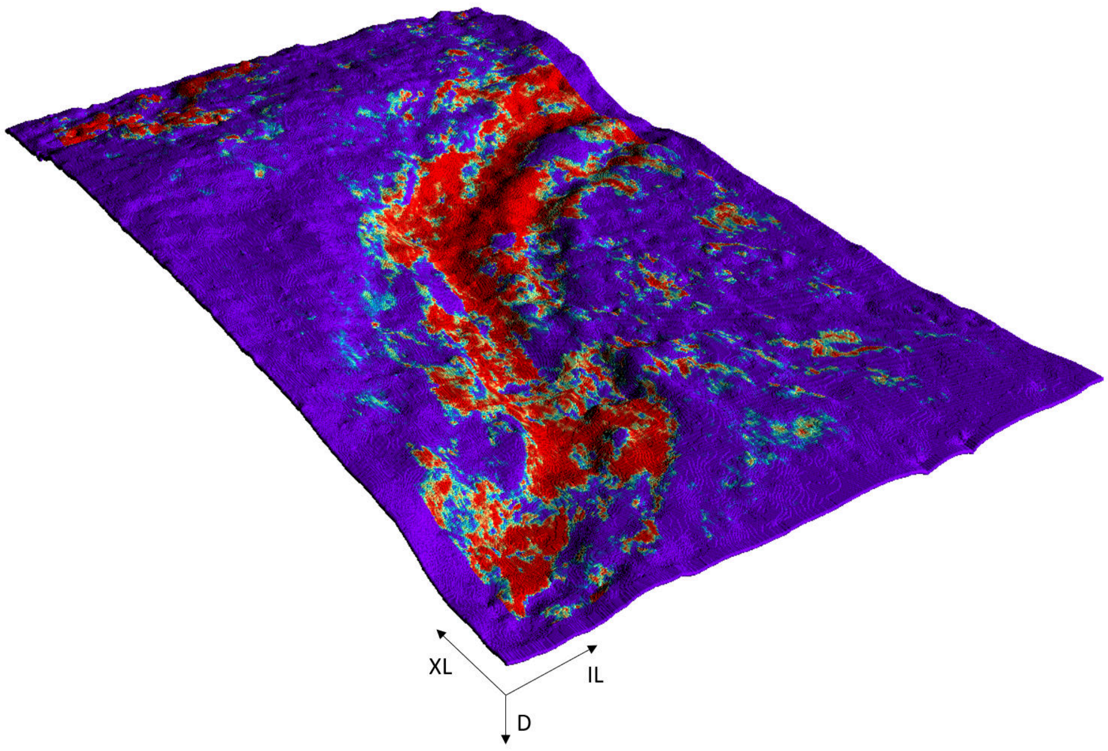


(b) Weighted accuracy $WAcc_{\text{K-fold}}$.

Figure 3.51: Near volume $N_{\text{patch}} = 65$ Base II second segmentation: accuracies. We observe that the increase in input context extension results in increased accuracy, as occurred with Far. We also notice the overall increasing trend that is consistently observed in all Near simulations. It is interesting to note how the unweighted accuracy gives the indication that CNN B Dropout has an insufficient representational capacity as its performance saturates and tends to decrease compared to that of CNN A Dropout. However, this consideration seems to be reversed when considering weighted accuracy. Therefore, we cannot at this point make quantitative considerations about whether one architecture is better than the other.



(a) First 3D view.



(b) Second 3D view

Figure 3.51: Near volume $N_{\text{patch}} = 65$ Base II second segmentation: 3D view prediction image. Looking at Figure 3.52a it is noteworthy how the model generalized in the lower left region where there seems to be an internal bank separating two channel-bases. In this region, in fact, although our raw segmentation tells us that there is a channel-base, the model gives a more realistic response by classifying it as background.

3.3 Inference on Surroundings

In this section, we tried to extend the results obtained in Base II training/testing horizon to its surroundings, in particular to Base III and Base I horizons. It is important to point out that such inference is intrinsically badly posed as training occurred on a single horizon, i.e. on a surface with little depth variation. This means that models have not received examples that provide significant class discontinuities in depth. Therefore we believe that the results that we are going to expose are affected by this point of criticality and that this is reflected on predictions in multiple false positives that in some cases are artifacts, due to the unrepresentative training dataset, and in other cases the recognition of characteristic patterns of Base II that fall within the depth context of the windrose input patches. However, we believe that the HSV infographic methodology developed to analyze these predictions provides interesting information.

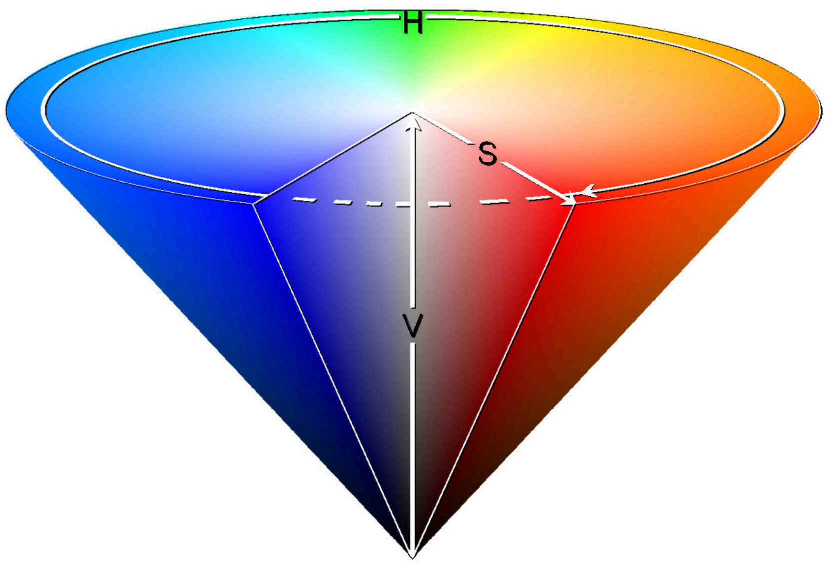


Figure 3.52: HSV color cone space. HSV (Hue, Saturation, Value) is an alternative representation of the RGB color model. In this model, colors of each Hue are arranged in a radial slice, around a central axis of neutral colors which ranges from black at the bottom to white at the top. The Saturation dimension resembles various shades of brightly colored paint, and the Value dimension resembles the mixture of those paints with various amounts of black or white paint.

We assigned to the Hue dimension the standard deviation associated with the channel-base predictions of the four models trained in the 4-fold cross-validation. Therefore voxels whose predictions are consistent between the four models will be colored blue (low standard deviation) while those more uncertain of red (high standard deviation). Assigning to the Value dimension the average probability it is possible to "turn on" only the most interesting voxels. To generate these images we have fully saturated colors, i.e. assigned to Saturation the maximum value.

The following images are produced by CNN A Dropout architectures trained for 1,000 epochs.

3.3.1 Base III

We recommend to review Figure 3.5 in order to comprehend the expected results for the following images. We also show the value of the seismics over the horizon to better interpret the model's behaviour in this area.

Far Volume

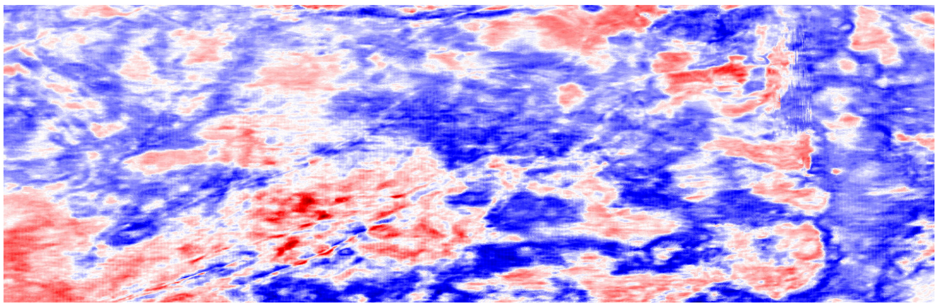


Figure 3.53: Far seismics on Base III horizon.

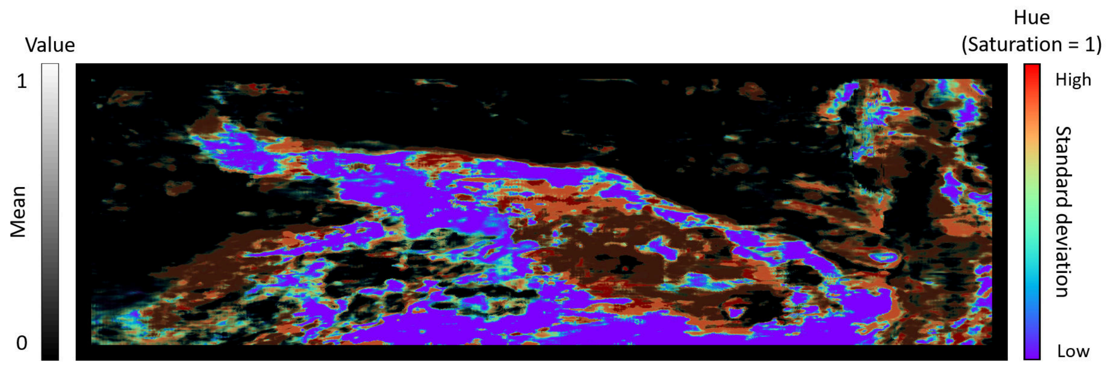


Figure 3.54: Base III inference image on Far volume. CNN A Dropout, $N_{\text{patch}} = 65$ trained on Base II second segmentation.

First of all, we notice that from the seismics over the horizon in Figures 3.53 a characteristic channel-base pattern is not as clearly visible as in Figure 3.24. Observing the prediction image in Figure 3.54 we gain information about the average channel-base probability of the four models and the uncertainty associated with this prediction. We notice how the models identify a region of interest on the right side where from topological information we know there is a channel-base. In this regard, we point out that the areas of high uncertainty and low probability (low intensity dark red areas) are often originated by the contribution of the model that has not been trained on subsample 4 (Test set 4), which is also the area where we have strong doubts on the validity of our manual segmentation. Therefore, these areas also carry interesting information about the consistency of our segmentation. We observe the presence of a channel-base that reminds for position and morphology (see the intra-channel bank) much that present in Base II. It is then possible that this is a false positive induced by the training performed on Base II.

Near Volume

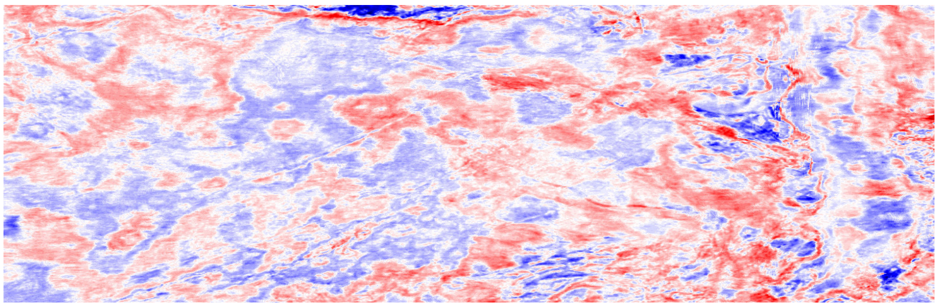


Figure 3.55: Near seismics on Base III horizon.

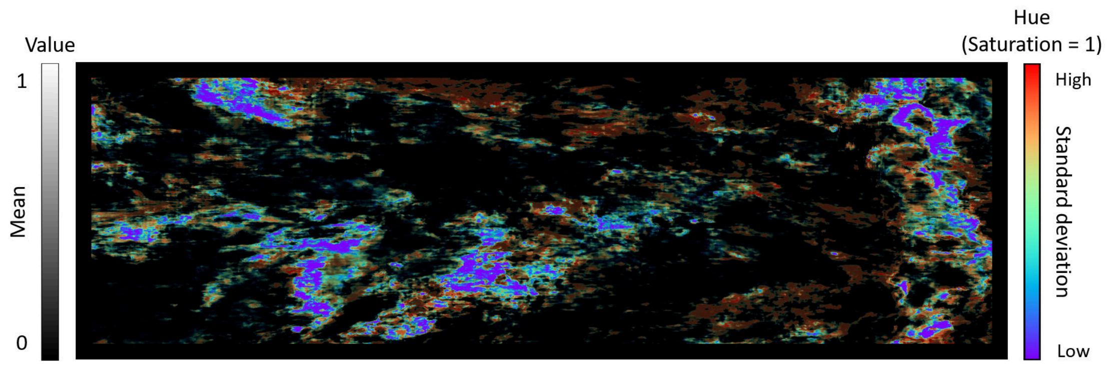


Figure 3.56: Base III inference image on Near volume. CNN A Dropout, $N_{\text{patch}} = 65$ trained on Base II second segmentation.

We notice that from the seismics over the horizon in Figure 3.55 a characteristic channel-base pattern is not as clearly visible as in Figure 3.25. We observe in Figure 3.56 how in this case the channel-base on the right side of the image has been well recognized. However, we identify multiple false positives in the center of the image that are probably due to the criticality exposed at the beginning of this section regarding the applicability of this inference.

3.3.2 Base I

We recommend to review Figure 3.7 in order to comprehend the expected results for the following images. We also show the value of the seismics over the horizon to better interpret the model's behaviour in this area.

Far Volume

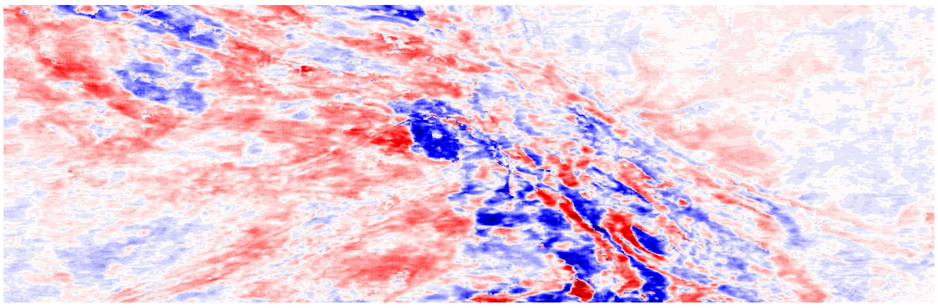


Figure 3.57: Far seismics on Base I horizon.

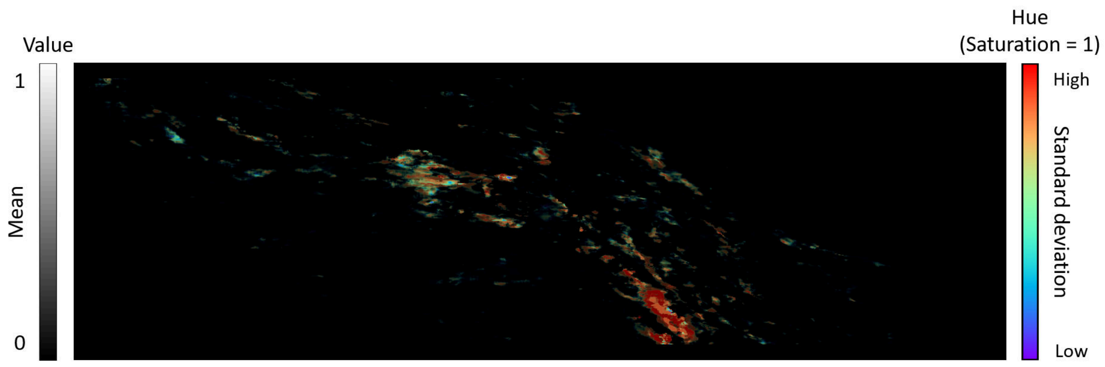


Figure 3.58: Base III inference image on Far volume. CNN A Dropout, $N_{\text{patch}} = 65$ trained on Base II second segmentation.

We observe that compared to the case of Base III here in Figure 3.57 a well recognizable channel-base structure is visible. In Figure we notice how on Base I horizon the models do not predict any channel-base, except for the model that has not been trained on subsample 4. The contribution of that model is the only that seems to slightly predict a channel-base in the region that we know to be a channel-base. This is a strong indication that the segmentation used in the rightmost subsample is not consistent with the other subsamples of Base II as well as with the implicit learned definition of channel-base.

Near Volume

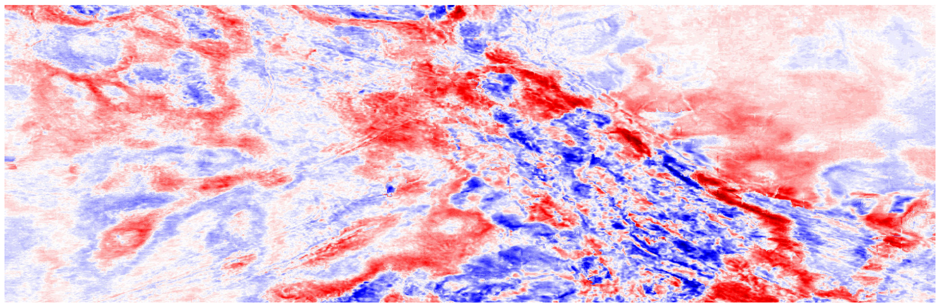


Figure 3.59: Near seismics on Base I horizon.

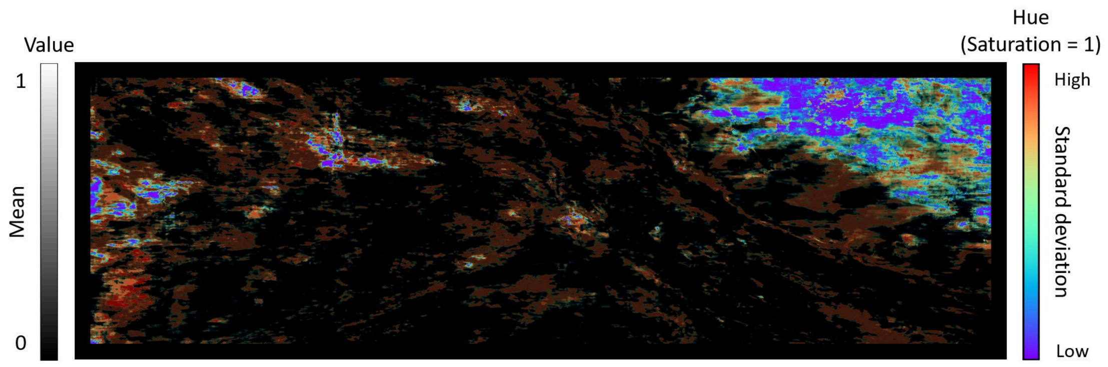


Figure 3.60: Base III inference image on Near volume. CNN A Dropout, $N_{\text{patch}} = 65$ trained on Base II second segmentation.

We notice that from the seismics over the horizon in Figure 3.59 a characteristic channel-base pattern clearly visible. We observe a large region of false positives in the upper right corner whose cause lies in the insufficiently representative case history of training examples provided by Base II. Once again we point out that the only model that predicts channel-base in the region where we know there is channel-base, although it predicts numerous false positives throughout the horizon, is the one that has not been trained on subsample 4.

According to these images, we believe that the problem defined on the Near volume is more difficult than the one defined on the Far volume. However, we also believe that the problem should be defined on the same volume from which the horizons have been segmented.

Chapter 4

Conclusion

This work consisted in the study and application of volumetric Deep Learning (DL) approach to seismic data provided by Eni S.p.A., with an industrial utility perspective. After a series of fruitful meetings with the Upstream & Technical Services team, we clearly defined the final objective of this approach: the automatic search for geological structures such as turbidite channel-bases, as potential regions of interest for the Oil & Gas industry.

The dataset Eni gave us comprises of:

- Two high quality 3D Pre-Stack Depth Migrated (PSDM) Volumes: one *Near* reprocessed angle stack and one *Far* angle stack.
- Three manually interpreted surfaces or *horizons* that have at least one embedded turbidite channel-base: Base I, Base II, Base III.
- Four high quality exploration porosity *well logs* ($\mathrm{W}_1, \mathrm{W}_2, \mathrm{W}_3, \mathrm{W}_5$) each one composed of: the actual measured porosity log and the relevant frequency filtered version.

The information on where the turbidite sediments infills are, is a very difficult one to gain and it requires a lot of time, for geologists and geophysicists to correctly interpret the seismic volume, and money, to collect well logs in order to establish high porosity regions. Furthermore, even if this effort is placed at work, a 3D reliable label with sufficient spatial resolution is impractical. On the other hand, the extraction of horizons is a relatively fast and semi-automated process done using 3D visualization and processing software, such as Petrel. This is because the seismics is the most important attribute in interpreting and recognizing horizons.
Since we do not have the information on where the turbidite sediments infills are, we can not set up the classification task as a direct search for this kind of objects. However, we do have three horizons with at least one embedded turbidite channel-base system.

After all these consideration, we choose to define the deep learning task as binary classification between turbidite channel-base and background using as input a sub-volume context or a 3D patch. Or, in other words, we choose to build models that for every voxel answer the question:

"*Does this voxel belong to a sub-volume that represents a channel-base system?*"

Since implementing this DL approach through 3D input patches means that the complexity of the problem grows very quickly with the extension of this 3D context (like third-degree polynomial) and thus undermining its success due to the *curse of dimensionality*, we decided to take an alternative route before admitting the need to move to full 3D. It has been decided to work with a certain number of 2D patches, according to what could be defined as a 2.5D approach. The most intuitive choice to transform an intrinsically 3D problem into a 2D one is to take slices along the coordinated axes of the subvolume (Cross Line XL, In Line IL, Depth D), in a mode that we have named "windrose".

In this work, we have analyzed and studied the following aspects and points of criticality.

- **Preprocessing**

  Before implementing a deep learning algorithm it is good practice to study how the numerical range of examples behaves in the training context. The training algorithms in fact, due to numerical crunching reasons, do not work properly if the input values are either too small or too large. In our case values contained in seismic volumes have an extremely wide dynamic range in the order of 10^8. We analyzed how the training process was influenced by three types of preprocessing: original (original data unchanged), normalized (we brought the dynamic range of the entire seismic volume within the range $[-1, 1]$, while preserving zero-crossing) and custom (we adjusted the normalized version by increasing data range by a factor of 100).

  Results showed how an appropriate data preprocessing step substantially improves both the training process, especially for models integrating the dropout regularization technique, and the generalizing ability of the models.

- **Dataset augmentation**:

  Given the scarcity of labeled examples compared to the number of possible physically acceptable configurations in which 3D patches can be found. We decided to implement the augmentation dataset in order to improve the generalization of the models, providing artificially generated examples generated from real ones through transformations that preserve their realism. We implemented the following transformation: XL-flipping, IL-flipping, D-translation, D-rotation and Scaling.

We compared two simulations with the same number of examples, one using only the native ones and the other with a mixture of original and augmented data. The results showed how the use of artificially generated data allows models to better generalize, at least in a qualitative way. In this approach, the number of configurations accessible through the original examples is much lower than the number of configurations physically plausible or even findable in the seismic volume. Just think of all the possible orientations in the XL-IL plane in which a geo-object can be found. We have therefore concluded that the dataset augmentation for this type of DL approach is fundamental.

- **Channel-base segmentation**:

  We manually segmented from Base II horizon a channel-base using only topological information. Unfortunately, it is difficult to exactly define where the channel-base ends and it was part of this work to figure out how this definition affects our DL classification problem.

  We studied qualitatively and quantitatively through K-fold cross-validation two similar segmentations of the previously mentioned cannel-base: an extended one that also included bank areas and a more stringent one that excluded those few areas that we considered possible outliers. We found that even a small variation in ground truth has a great effect on both the visual consistency of the prediction images and the accuracy measured by K-fold cross-validation. In particular, we observed an increase in accuracy of about 10% using the most stringent segmentation. We cannot say that the second segmentation is better because the accuracy is better, what we can say instead is that this type of problem is very sensitive to segmentation and therefore it is necessary to invest sufficient effort in the generation of reliable labels.

- **Context extension**:

  We previously introduced the concept of the 3D patch in the context of a patch-based classification task. This context must be large enough to provide adequate information about the voxel's surroundings for the classification task, but at the same time it must be small enough to keep the complexity of the problem under control and allow good generalization.

  We studied qualitatively and quantitatively two context extensions using K-fold cross-validation and found that, at least for the number of training examples chosen in our experiments ($\sim 80k$), the larger version allowed a 5-7% increase in accuracy and a large suppression of false positives. This probably depends on the spatial resolution of the seismics around Base II, which limits the informative content of the input patches. We concluded that, at least for the extensions analyzed, the damage caused by the increase in complexity due to the greater number of tunable parameters was outweighed by the benefits brought by the increase in information content due to a wider context extension. Therefore, we think that the results presented here can provide a point of reference for defining patch extensions in areas with different seismics resolution.

- **Far volume or Near volume**: As expressed above we worked on two seismic volume, Far and Near. These two volumes are different types of stacks and they are sensitive to different seismic properties; for instance we could say that the Far is more sensitive to fluid presence.

  We studied qualitatively and quantitatively through K-fold cross-validation the problem set on both Far and Near volume. From our experiments we observed that, from a statistical point of view, the performances of the models trained on Near are more convincing than those on Far. We noticed that the accuracy as measured by K-fold cross-validation tends to increase with the number of training epochs, which is not the case with Far. We can not say that it is better to set the problem on the Near since horizons used in this work have been segmented from the Near seismics. For this reason, we believe that the problem appears more consistent on Near. Therefore what we can say from the studies carried out is that the problem appears to be better set on the volume used for the horizons segmentation.

Given the results achieved and the potential shown by this DL approach, we have the following recommendations to proceed with a further study.

- From our experiments, we observed that the problem set on the Far volume appears easier than the one set on the Near volume. Therefore, we suggest to carry out a study similar to the one carried out in this work, in order to finally establish whether or not it is useful to set the problem on the Far volume, using horizons segmented on the Far volume.

- Given the poor ability of the models to generalize on other horizons at different depths, we suggest doing a multi horizon training. We believe that this can provide the necessary information of strong depth discontinuity to better infer in all three spatial directions. In this regard, we recommend making the following multi horizon training to compare it with the results expressed in this work: training on Base II plus Base I and inference on Base III.

- We believe that the approach developed in this work involving the windrose input patches has proven to be satisfactory. We consider interesting to explore new input patch configurations by adding slice at different angles to windrose patch as a sequence of gradually increasing complexity approaches up to full 3D.

From a qualitative point of view, the generalization results on Base II were judged by Eni's experts to be consistent. Therefore, the proposed method is able to extract valuable information from the seismic data volume. However, a scale-up of examples and computing power is necessary to unleash a credible result on the entire seismic volume.

Bibliography

[1] F. Castelli, *3d cnn methods in biomedical image segmentation*, Master's thesis, University of Bologna, 2018/2019.

[2] V. Cherkassky and F. M. Mulier, *Learning from Data: Concepts, Theory, and Methods*, Wiley-IEEE Press, 2007.

[3] D. Cielen, A. Meysman, and M. Ali, *Introducing Data Science: Big Data, Machine Learning, and more, using Python tools*, Manning Publications, 2016.

[4] V. Dumoulin and F. Visin, *A guide to convolution arithmetic for deep learning*, 2016. cite arxiv:1603.07285.

[5] X. Glorot and Y. Bengio, *Understanding the difficulty of training deep feedforward neural networks.*, in AISTATS, Y. W. Teh and D. M. Titterington, eds., vol. 9 of JMLR Proceedings, JMLR.org, 2010, pp. 249–256.

[6] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*, Adaptive Computation and Machine Learning series, MIT Press, 2016.

[7] P. Keary, M. Brooks, and I. Hill, *An Introduction to Geophysical Exploration*, Blackwell, 3 ed., 2002.

[8] M. Lin, Q. Chen, and S. Yan, *Network in network*, 2013.

[9] Z. Liu, P. Chow, J. Xu, J. Jiang, Y. Dou, and J. Zhou, *A uniform architecture design for accelerating 2d and 3d cnns on fpgas*, Electronics, 8 (2019), p. 65.

[10] G. Mazzamuto, I. Costantini, M. Neri, M. Roffilli, L. Silvestri, and F. Pavone, *Automatic segmentation of neurons in 3d samples of human brain cortex*, K. Sim, P. Kaufmann (eds) Applications of Evolutionary Computation. EvoApplications 2018. Lecture Notes in Computer Science, Springer, 10784 (2018).

[11] W. McCulloch and W. Pitts, *A logical calculus of the ideas immanent in nervous activity*, Bulletin of Mathematical Biology, 5 (1943), pp. 115–133.

[12] A. Mikołajczyk and M. Grochowski, *Data augmentation for improving deep learning in image classification problem.* International Interdisciplinary PhD Workshop (IIPhDW), Swinoujścia, 2018.

[13] M. Minsky and S. Papert, *Perceptrons: An Introduction to Computational Geometry*, MIT Press, Cambridge, MA, USA, 1969.

[14] T. M. Mitchell, *Machine Learning*, McGraw-Hill International Editions, McGraw-Hill, 1997.

[15] T. Mukerji, A. Jorstad, and G. Mavko, *Near and far offset impedances: Seismic attributes for identifying lithofacies and pore fluids*, K. Sim, P. Kaufmann (eds) Applications of Evolutionary Computation. EvoApplications 2018. Lecture Notes in Computer Science, Springer, 10784 (1998), pp. 4557–4560.

[16] A. Ottaviani, *Faster and efficient 3d interpretation for geological and stratigraphic delineation of deep-water turbiditic systems by integration of recent geomorphologic visualization and subsurface images from seismic attributes*, Master's thesis, University of Perugia, 2015/2016.

[17] M. Roffilli, *Advanced Machine Learning Techniques for Digital Mammography*, PhD thesis, Tech. Rep. UBLCS-2006-12, University of Bologna, March 2006.

[18] F. Rosenblatt, *The perceptron: A probabilistic model for information storage and organization in the brain*, Psychological Review, (1958), pp. 65–386.

[19] J. Seri, *Recoinnaissance of geoobjects in a turbiditic sedimentation enviroment (east africa deep offshore) using machine learning approaches*, Master's thesis, University of Perugia, 2018/2019.

[20] C. Shorten and T. M. Khoshgoftaar, *A survey on image data augmentation for deep learning*, Journal of Big Data, 6 (2019), pp. 1–48.

[21] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, *Dropout: A simple way to prevent neural networks from overfitting*, Journal of Machine Learning Research, 15 (2014), pp. 1929–1958.

[22] S. Suslick and D. Schiozer, *Risk analysis applied to petroleum exploration and production: an overview*, Journal of Petroleum Science and Engineering, 44 (2004), pp. 1–9.

[23] V. N. Vapnik and A. Y. Chervonenkis, *On the uniform convergence of relative frequencies of events to their probabilities*, Theory of Probability and its Applications, 16 (1971), pp. 264–280.

[24] P. Werbos, *Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences*, Harvard University, 1975.

[25] Ö. Yilmaz and S. M. Doherty, *Seismic Data Analysis: Processing, Inversion, and Interpretation of Seismic Data*, Crisp Fifty-Minute Books, Society of Exploration Geophysicists, 2001.

[26] M. D. Zeiler, *Adadelta: An adaptive learning rate method*, 2012.

[27] G. Zhiqian, S. Zhihuan, X. D. Steven, and H. Biao, *Data mining and analytics in the process industry: The role of machine learning*, IEEE Access, 5 (2017), pp. 20590–20616.