Politecnico di Torino

Politecnico di Torino

Department of Environmental, Land and Infrastructure Engineering

Master of Science in Petroleum and Mining Engineering

Physics-guided Deep Learning for Seismic Inversion

Supervisor:

Prof. Laura Valentina Socco

Candidate:

Farzalibayli Toghrul

February 2023

Thesis submitted in compliance with the requirements for the Master of Science degree in Petroleum and Mining Engineering

Abstract

An important part of the geophysical analysis is seismic inversion, which provides insight into the Earth's structure. Conventional seismic inversion methods require enormous computational power, human intervention, long processing time, and uncertainties in predictions. With exponential growth in the Artificial Intelligence field, particularly in the Deep Learning field, there is a possibility to apply various Deep Learning and Machine Learning algorithms on multiple seismic inversion problems. Here Convolution Neural Networks were used to create a model to directly estimate porosities from seismic data. By teaching the model exact physics, the need for large labeled data was removed. The purpose of the thesis is to give a reader all the information to create the model without strong prior knowledge of explora tion geophysics and deep learning algorithms. In the methodology chapter, first, basic information on seismic exploration, seismic inversion, and problems related to it is written. It is followed by an introduction to artificial intelligence, machine learning, and deep learning methods. How deep learning models are created and what their parameters mean. In the end, the model is introduced and all its elements are explained. The model was updated multiple times to find the best predictions and the models' accuracies were shown for each of them. The advantages and disadvantages of this and other models were discussed.

Keywords: Seismic inversion, Porosity estimation, Deep learning in geophysics, 1D Convolutional neural networks, Unsupervised learning, Physics guided networks

Acknowledgement

This project would not have been possible without the support of many people. First and foremost, I wish to thank my supervisor Prof. Laura Valentina Socco for the opportunity to work on this project. Also, many thanks to Assistant Prof. Leonardo Azevedo from Tecnico Lisboa for giving the initial idea of using deep learning methods in geophysics.

I am also very grateful to all professors in the Department of Environment, Land, and Infrastructure Engineering for teaching and guiding me throughout my educational journey at the Polytechnic University of Turin. I have deepest gratitude to all my professors for their passion to teach and their professional work. Their approach to their profession, work discipline and dogmas shaped me and made the person who I am right now.

Finally, I express my deepest gratitude to my family and friends, whose constant love and support keep me motivated and confident.

Table of Contents

List of Figures

List of Abbreviations

CNN	Convolutional Neural Networks
ML	Machine Learning
DL	Deep Learning
1D	One dimensional
OLS	Ordinary Least Squares
Tanh	Hyperbolic function
Relu	Rectified Linear Unit
Adam	Adaptive Moment Estimation
GAN	Generative Adversarial Network
AI	Acoustic Impedance

List of Symbols

$\boldsymbol{\rho}$	Density
λ	Lame constant
$\boldsymbol{\mu}$	Lame constant
I	Acoustic impedance
$\boldsymbol{V_p}$	Primary wave velocity
$\boldsymbol{V_s}$	Secondary wave velocity
φ	Porosity

Chapter 1: Introduction

Seismic inversion is a crucial step in subsurface exploration and resource production, with the primary aim of deriving subsurface properties such as porosity and acoustic impedance from seismic data. These properties are critical in characterizing subsurface reservoirs and making informed decisions about exploration and production (Gerard T. Schuster, 2017).

Traditionally, seismic inversion was performed using deterministic methods, which relied heavily on assumptions and prior knowledge of the subsurface. While these methods have been successful in many cases, they can be limited by the accuracy of the assumptions made and the limited data available (Veeken et al., 2004). With the advent of machine learning, new approaches to seismic inversion have been developed that offer significant advantages over traditional methods (Zhang et al., 2012).

Deep learning models, particularly Convolutional Neural Networks (CNNs), have been widely used in various fields. The field of petroleum geophysics has also seen significant advancements in recent years with the integration of deep learning techniques (A. Adler and T. Poggio, 2021). The authors of the paper have concluded that Deep Learning (DL) has shown promising results for seismic inversion. They have noted that compared to conventional seismic inversion methods, DL training and inference processing time is faster and does not require an initial earth model estimate. The authors have also noted that DL is capable of processing multi-modal input data such as well-logs and seismic data, which facilitates joint inversion. In addition, DL is capable of simultaneously estimating multiclass data such as P-wave and S-wave velocities, density, and other parameters. Deep learning has shown great potential in various applications within the field, including seismic inversion, reservoir characterization, seismic attribute analysis, and seismic denoising.

The authors have highlighted important research directions that will further advance the integration of DL into seismic data analysis workflows, including: physics-guided architectures, unsupervised representation learning, regularization, DL architectures for 3D model inversion, robustness to seismic data degradation, and adversarial robustness of DL inversion. The authors have concluded the article by noting that the availability of large-scale open datasets has been a key factor in the success of DL and that the future

availability of similar datasets for seismic inversion will significantly advance DL solutions in this area.

In this thesis, the application of a 1D CNN to the problem of porosity and acoustic impedance estimation from seismic traces and wavelets was investigated. In order to overcome the limitations of limited datasets, a physics-guided deep learning approach was employed and synthetic data was created work with. The 1D CNN model is trained on the synthetic dataset to learn the complex relationships between subsurface properties and seismic data. The results of the study demonstrate the feasibility of using deep learning models for solving seismic inversion problems and provide a framework for further research in this area.

To achieve this goal, a literature review was conducted to gather information on seismic inversion, porosity, and acoustic impedance estimation. Next, a synthetic seismic data was created and used for the experimentation. A significant amount of effort was put into hyperparameter tuning to determine the best deep learning parameters for this problem. The results showed that the deep learning approach was able to provide excellent estimations of porosity and acoustic impedance, demonstrating the feasibility and effectiveness of using deep learning for this task.

This thesis contributes to the growing body of research in the field of deep learning in geophysics by exploring the use of 1D CNNs in solving the complex inverse problem of porosity and acoustic impedance estimation from seismic traces and wavelets. The results of the study provide a valuable contribution to the field of seismic inversion and demonstrate the potential of deep learning models in this area. Furthermore, the use of a physics-guided deep learning approach and synthetic data highlights the potential for using these techniques to overcome limitations in real-world datasets and improve the accuracy of seismic inversion results. It also presents a systematic study of the performance of 1D CNNs for this task and shows the potential of this approach. The model and information provided in this thesis serve as a valuable resource for new researchers in the field, as it offers a clear and concise introduction to both geophysics and deep learning, allowing them to easily recreate the work and build upon it.

Chapter 2: Problem Statement

Traditional seismic inversion methods, such as least-squares inversion, linearized inversion, and full waveform inversion, have several limitations and challenges (Veeken et al., 2004). These include:

- Complexity: Traditional seismic inversion methods can be computationally expensive and time-consuming due to the large amounts of data involved and the need to solve non-linear inverse problems.
- Lack of robustness: Traditional seismic inversion methods can be sensitive to noise and errors in the data, leading to poor results when applied to noisy or low-quality data.
- Limited accuracy: Traditional seismic inversion methods can struggle to accurately estimate complex subsurface structures and geological features due to limitations in the underlying physics-based models and assumptions.

Deep learning methods, specifically Convolutional Neural Networks (CNNs), have shown great promise in addressing these limitations and challenges in seismic inversion (A. Adler and T. Poggio, 2021). By using deep learning, researchers can leverage the ability of neural networks to automatically learn and extract complex features from the seismic data, without being limited by assumptions about the underlying geology. This results in improved accuracy and robustness compared to traditional methods, as well as reduced computation time. The key strengths of deep learning methods are:

- Data-driven approach: Deep learning methods are based on a data-driven approach, where the model is trained on a large dataset and then used to make predictions on unseen data. In contrast, traditional seismic inversion methods rely on physical models that are derived from first principles, which can limit their ability to capture complex subsurface structures and geological features.
- Automated feature extraction: One of the key strengths of deep learning methods is their ability to automatically extract complex features from the seismic data, without the need for manual feature engineering. This can lead

to improved accuracy and robustness compared to traditional methods, which rely on manually defined features.

- Handling of large datasets: Deep learning methods are designed to handle large amounts of data, making them well suited for processing large seismic datasets. This is particularly important for seismic inversion, as it typically involves processing large amounts of data to generate subsurface models.
- Generalization: Deep learning models are capable of generalizing to new, unseen data, meaning they can be used to make predictions on data that was not present in the training dataset. In contrast, traditional seismic inversion methods can be limited by their assumptions and may not be able to generalize to new data.
- Real-time processing: Deep learning methods can be implemented in real-time, allowing for the rapid updating of subsurface models as new data becomes available. This is particularly important in the context of seismic inversion, where subsurface models are updated regularly as more data is collected.

Overall, deep learning methods offer several advantages over traditional seismic inversion methods, including improved accuracy and robustness, ability to handle large datasets, and real-time processing capabilities. These advantages make deep learning a promising approach for solving seismic inversion problems in geophysics.

2.1 Research aim, objectives, and questions

The aim of this research is to explore the feasibility and effectiveness of using deep learning methods, specifically 1D Convolutional Neural Networks (CNNs), in addressing the issue of porosity and acoustic impedance estimation from seismic traces, wavelets and prior models derived from well logs. The research will examine the limitations of traditional methods in seismic inversion problems and investigate the potential benefits of using deep learning techniques for improving the accuracy and reliability of porosity and acoustic impedance estimation. The research will create synthetic seismic data, create a 1D CNN model, perform hyperparameter tuning, and evaluate the performance of the 1D CNNs for

porosity and acoustic impedance estimation. The results of this study will provide valuable insights into the use of deep learning methods for geophysics and contribute to the development of advanced techniques for seismic inversion problems.

Additionally, this research will contribute to the ongoing efforts in the geophysics community to incorporate deep learning techniques into traditional methods for seismic inversion. By demonstrating the feasibility and effectiveness of using 1D CNNs for porosity and acoustic impedance estimation, this research has the potential to inspire further advancements in this field and drive the development of new techniques for seismic inversion problems. Furthermore, the results of this study will provide valuable insights into the application of deep learning in the petroleum geophysics, thereby strengthening the overall understanding of this topic and paving the way for future research and innovation.

Chapter 3: Theory

In this section, I will provide a brief overview of seismic exploration, inversion, and deep learning as they relate to our porosity estimation model. First, we will cover the process of acquiring seismic data, including the underlying physics. We will then delve into the processing of this data to prepare it for seismic inversion. Finally, I will explain both forward and inverse models.

In the subsequent part of this chapter, we will examine the impact of machine learning and deep learning algorithms on the field of science. I will provide a brief and simple explanation of these algorithms, which have revolutionized the way we analyze data. This information is crucial in understanding the background and motivations for our porosity estimation model.

3.1 Seismic Exploration

Seismic exploration is widely used for hydrocarbon exploration. The first step of seismic exploration is data acquisition. For data acquisition, the most commonly used method is reflection seismology (Bacon M., 2005). It is known as the best seismic exploration method due to its high investigation depth and high resolution and relatively cheap costs compared to invasive methods, like well logs. To gather seismic data seismic waves artificially generated on either subsurface or marine (figure 1). For hydrocarbon exploration vibratory sources are good sources of artificial perturbations. Those propagated waves are then recorded by thousands of transducers such as geophones, which supply electric signals, proportional to the seismic vibration. The electric signals then are measured, digitized and recorded.

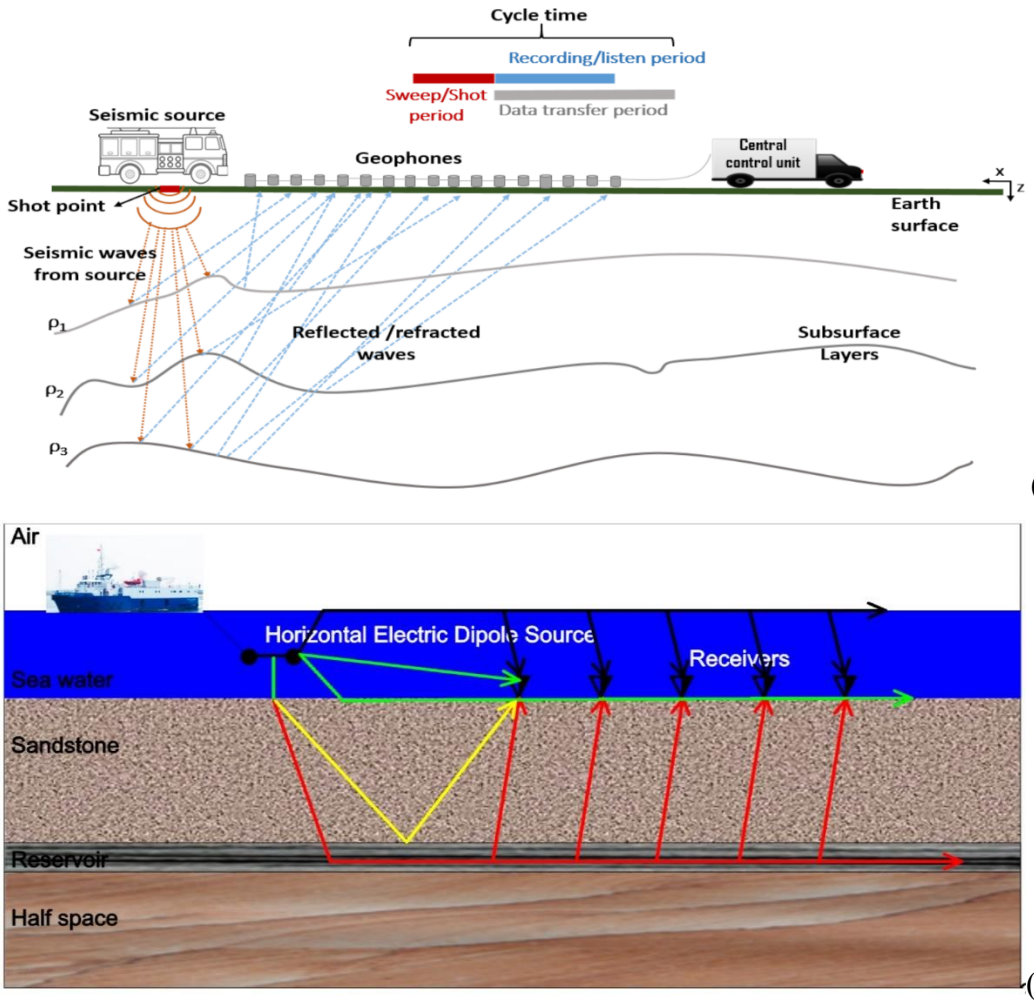


Figure 1, Seismic data acquisition on (a) surface, (b) marine (Barkaszi 2019)

When a seismic wave passes through the lithologies with different acoustic impedances, their direction changes thus resulting in different velocities and travel times. Knowing the travel times from the source to various receivers, and the velocity of the seismic waves, a geophysicist then attempts to reconstruct the pathways of the waves in order to build up an image of the subsurface (Sheriff et al., 1982). There are two types of seismic body waves captured by seismic recordings: compressional and shear waves. They are also called P(primary) and S(secondary) waves, because of compressional waves reaching the receivers first. The difference in propagation is shown in figure 2.

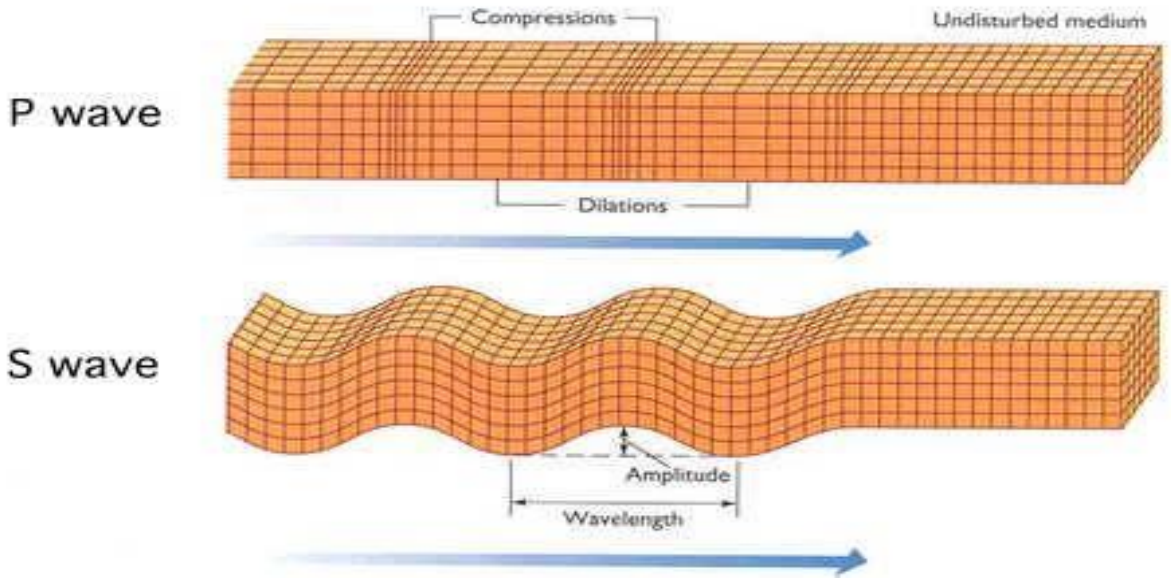


Figure 2, Compressional (P) and Shear (S) waves (A.E. Charola et al., 2014)

Velocity of the propagated waves can be described by the formulas (Robinson E. A., 1983):

$$V_p = \sqrt{\frac{\lambda+\mu}{\rho}} \quad (1)$$

$$V_s = \sqrt{\frac{\mu}{\rho}} \quad (2)$$

Where:

ρ is density

λ is Lame's first parameter and μ is Lame's second parameter, which describe stress-strain relations within a solid medium.

As mentioned before, when seismic waves pass through different rock surfaces some of the energy reflects off the interface. It is due to differences in acoustic impedances (figure 3). The amplitude of the reflections depends on the velocities and densities at the interface. The relation is described by the following formulas (Sheriff et al., 1982):

$$I = \rho * V_p \quad (3)$$

Where:

I is the acoustic impedance

ρ is density

V_p is the velocity of compressional waves

$$R = \frac{I_2 - I_1}{I_{2+I_1}} \quad (4)$$

Where:

R is reflection coefficient

I1 and I2 are acoustic impedances of the upper and lower subsurface layers.

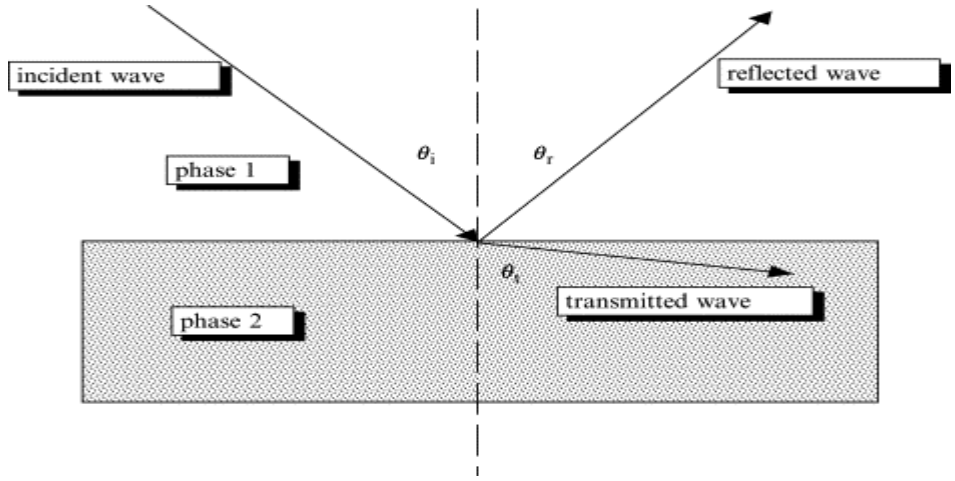


Figure 3, Illustration of the sound propagation through a phase boundary (Andrei S. Dukhin 2010)

After the raw seismic data is acquired, the next step is processing the data. The most important applications in data processing are deconvolution, stacking and migration. It is used to improve the resolution, signal/noise ratio and to correct locators' spatial position. At first seismic data is preprocessed by filtering and muting unwanted portions of data and amplitude recovering (Yilmaz Öz, 2001). Then the mentioned methods are applied (figure 3). The difference between raw and processed data is shown in figure 4.

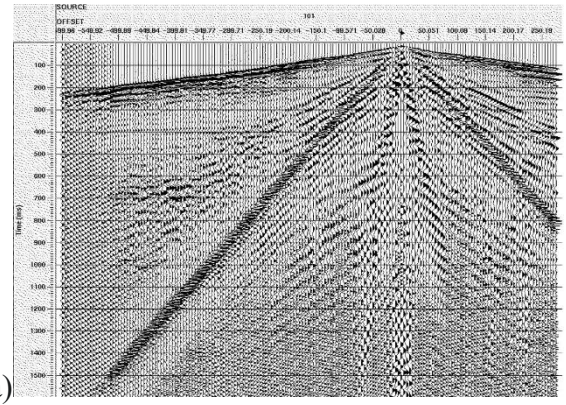
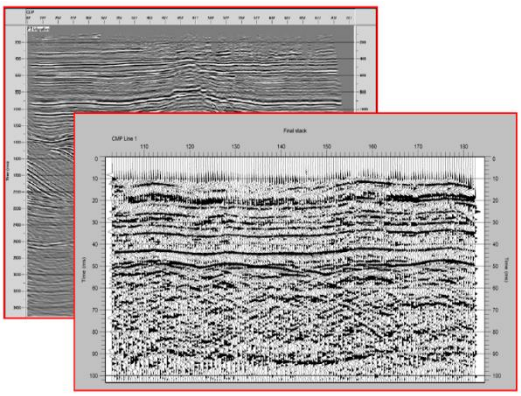
(a)   (b)

Figure 4, (a) Raw seismic data, (b) Post-stack seismic data

The last stage of the seismic exploration is data interpretation. In order to get the most detailed view of the subsurface to make the most correct interpretation it is needed to apply seismic inversion. This method is capable of predicting rock properties (lithology, fluid content, porosity) across a field survey by combining seismic and well data. For a better solution to inverse problems, it is important to solve forward problems well. Seismic inversion needs forward modeling to obtain synthetic seismic data that matches real seismic data. The relation between forward and inverse problems is shown in figure 5.

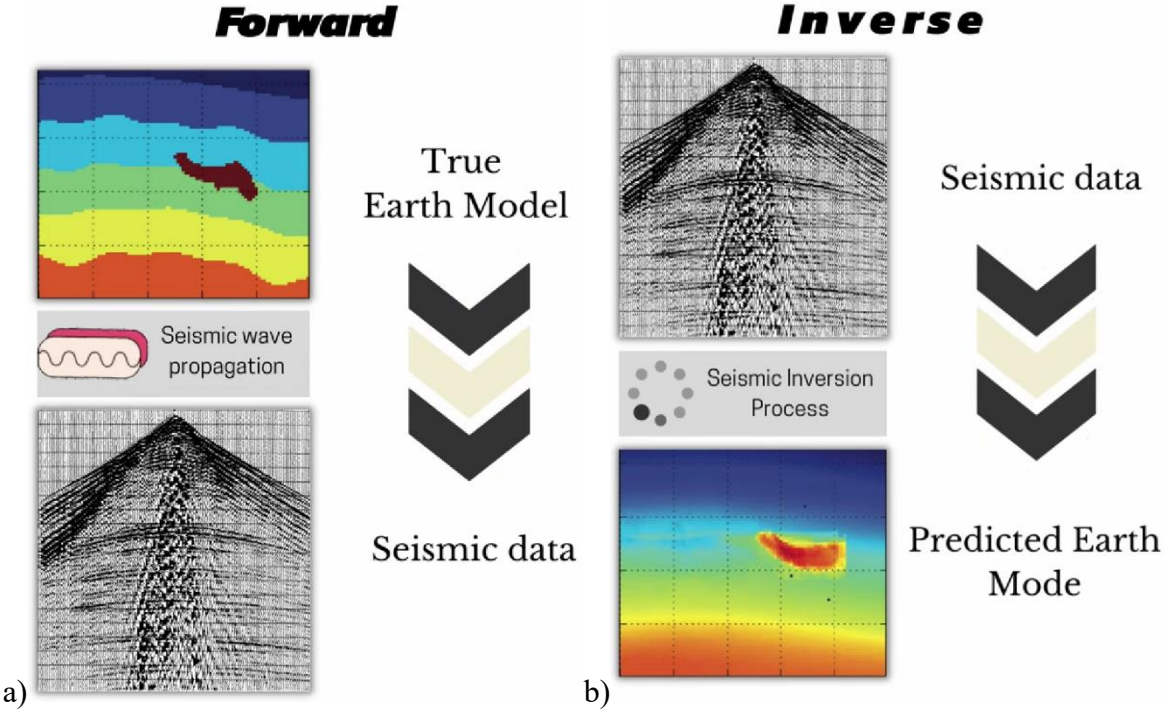


Figure 5, In a forward model seismic data, is recorded to model the subsurface. Inverse problem is applied to estimate the subsurface model from seismic data (A. Adler and T. Poggio, 2021).

Before understanding, how seismic inversion is used, it is important to understand how forward model is used in seismology. Seismic traces we get through data acquisition methods, mentioned before, are assumed to be dependent on 3 variables: reflectivity, wavelet and noise. The general formula used to describe it is this (Sheriff et al., 1982):

$$S(t) = R * W + N \tag{5}$$

Where S is seismic data, W is wavelet and N is noise.

This formula will be used to create a synthetic seismic data in the model created for porosity estimation. When we have p velocities and densities from well logs, we can find

acoustic impedance and calculate reflectivity series. After picking the wavelet we use convolution method to derive synthetic seismic trace, by also adding the noise (figure 6).

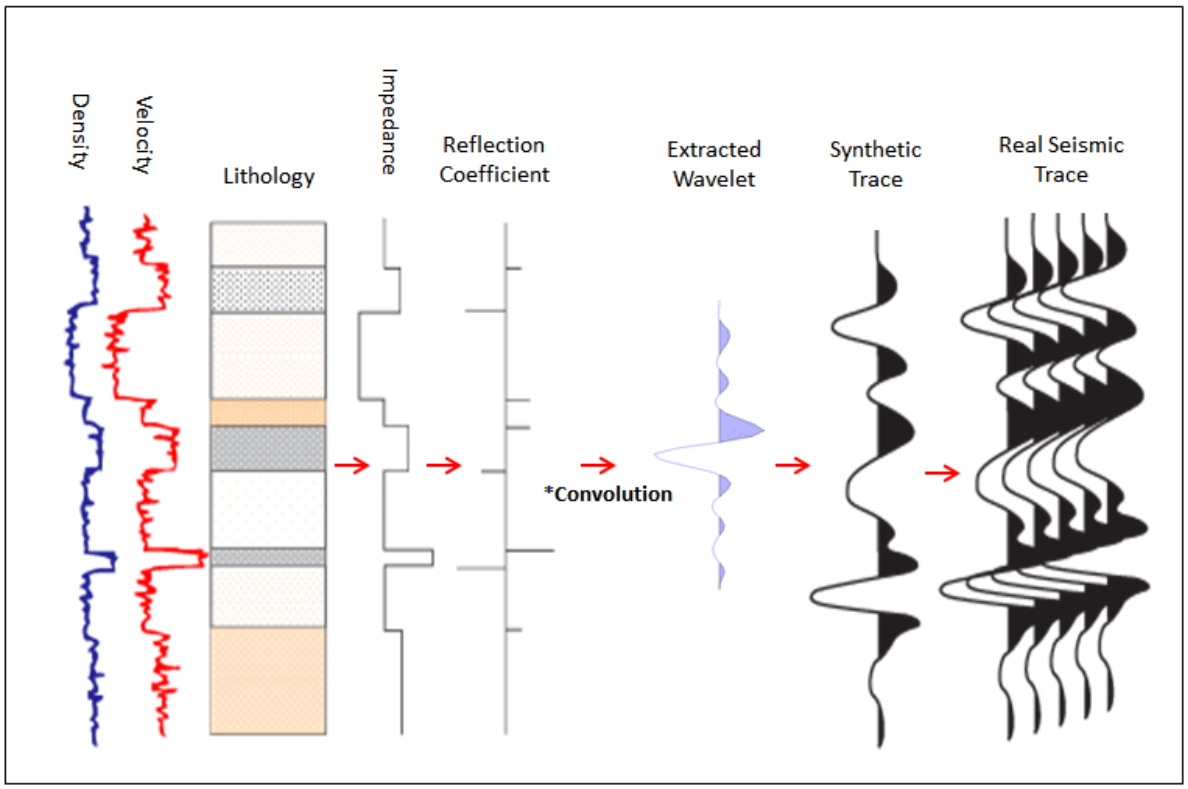


Figure 6, Seismic forward model illustration (Tankönytár, 2014)

Seismic inversion involves the same procedure but in opposite order. Here we don't use convolution method, but deconvolution. We try to derive physical properties from seismic traces. The general workflow of seismic inversion is shown in figure 7.

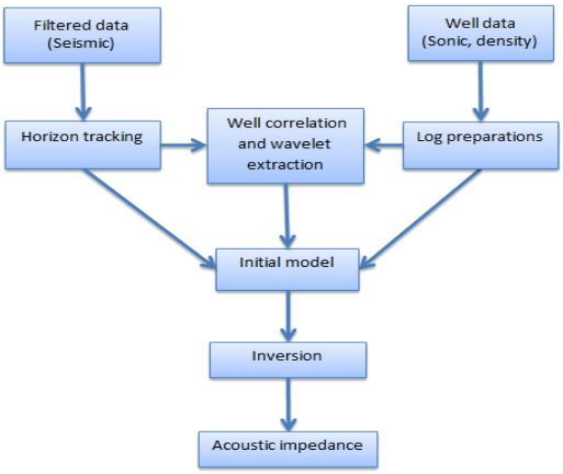


Figure 7, Seismic inversion workflow (Farfour, 2015)

A lot of the properties of the rocks can be determined by using well-log data (water saturation, gamma rays, shale volume, etc.) or by using seismic data (which is more difficult to obtain). There is, however, another aspect to seismic inversion that allows us to obtain additional rock properties such as impedance and its corresponding attributes (P-impedance, S-impedance, Vp/Vs, Poisson's Ratio, Mu*Rho, Lambda*Rho). Those properties are related to a number of factors, such as lithology, porosity or fluid content. Using the P-impedance of a given lithology, for example, we can calculate its porosity if we know the P-impedance of the lithology. Based on this relationship, if we combine the impedance at the well with the impedance calculated from the seismic data, it is possible to predict hydrocarbons throughout the survey.

There are two types of seismic inversion techniques that are used for seismic inversions, the post-stacking inversion, and the pre-stacking inversion (Russell and Brian H., 1988). Most seismic inversions are performed by using the first technique, Post-Stacking inversion. Using acoustic impedance data, seismic data, and basic knowledge of stratigraphy for interpretation, this technique transforms a single seismic information volume into an acoustic impedance data volume (figure 8(a)). A high-resolution subsurface image is created by removing the wavelet from seismic data. This is the method used in the model.

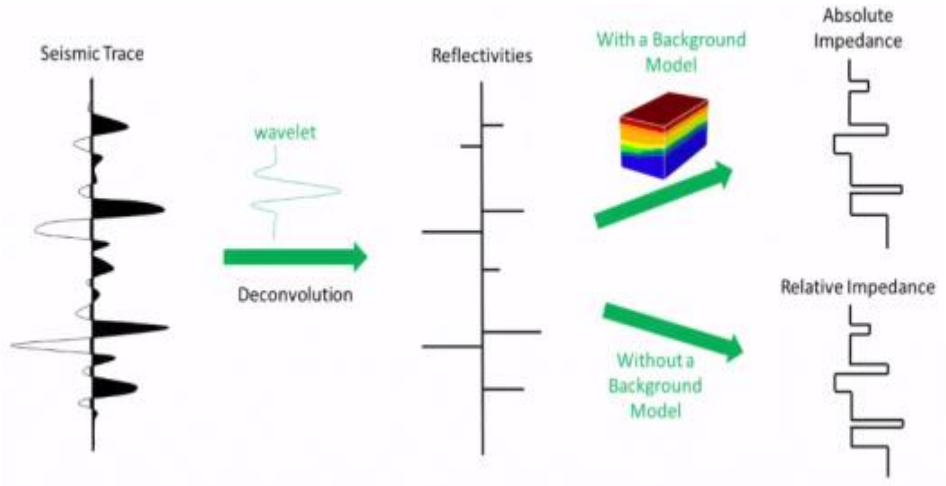


Figure 8, (a) Post-stack inversion, (b) Pre-stack inversion

Common post-stack inversion methods are colored, model-based and sparse-spike inversions (Maurya S.P. and Sarkar P., 2016). A model-based inversion consists of constructing a simple acoustic impedance model from the seismic trace, then convolving it with the wavelet in order to generate a synthetic response that is then compared with the

actual seismic traces. In order to reduce the difference between inverted trace and seismic trace to a threshold value, the acoustic impedance model is modified iteratively over a number of repetitions until the difference is reduced enough. An acceptable solution is a model with a very small difference between the two models. Model-based inversions have the advantage of giving satisfactory results, even when there is a limited control of the well and poor-quality seismic data is available. Seismic datasets themselves serve as guides for inversion, and by using the seismic data to invert, wavelets can be easily derived directly from the seismic data. Least-squares inversions are based on model-based inversions and have a threshold value equal to the smallest least-squares error. This is the method used in the model to create porosity model.

An inversion using colored data is a method of comparing the amplitude spectrum of a seismic dataset with one obtained from a well log. There is an operator created that matches seismic amplitude spectrums with the relative acoustic impedance data measured in the wells. In order to approximate an acoustic impedance equivalent, this operator is applied to the seismic traces and then convolved with them. As a result of the simple, fast, and robust nature of this method, it is ideal for quick and preliminary inversions, as it works even when there are noises present, thereby making it ideal for rapid and preliminary inversions. A disadvantage of the method is that output is somewhat imprecise, it is a band-limited method, and the input seismic needs to be in zero phase since there is no wavelet used. Moreover, it is assumed that the reflectivity spectra of one well represent the reflectivity spectrum of the entire area of interest. The ideal outcome, if the simulated trace is convolved with a wavelet, would be the reproduction of the actual seismic response. Using band limited seismic data, a high-resolution acoustic impedance profile can be generated that is directly related to the lithology or the rock properties.

An inversion method based on sparse spikes is used to simulate a seismic trace by integrating a minimum number of AI interfaces in order to model the subsurface reflectivity. A fundamental assumption of this formulation of acoustic impedance is that the reflectivity coefficient series linked with it is sparse. It is thus possible to model a seismic trace with fewer reflection coefficients; the only meaningful portions of the seismic trace will be those spikes with large impedance contrasts.

Seismic inversion is not as easy to do as forward problems and it has its own problems. Unlike forward problems, inverse problems are generally nonlinear, non-unique and unstable. Deep learning models are known to be good in solving inverse problems (Kamyab S., 2022). In seismic inversion, deep learning models can be trained on synthetic data, where the subsurface properties are known, and the corresponding seismic data is generated. This enables the model to learn the relationship between the subsurface properties and the seismic data and make predictions for real-world data. Incorporating the physics with the model would make the model predict even better. No matter how non-linear is the relationship, with enough time given to the model to learn it would find the relationship. The drawback would be over-generalizing, but with constant tuning of the model this problem can be solved too. The next chapter will introduce you AI and its subsets, where you will gather more insights on the power of AI and its impact in today's world.

3.2 Artificial Intelligence, Machine Learning and Its Subsets

Artificial intelligence (AI) is a machine that simulates the actions and thoughts of a human by simulating these actions and thoughts through a computer program designed to think like a human and mimic their actions. The term can be also applied to any machines that exhibit traits linked with a human mind like learning and problem-solving. A key feature of AI is that it is capable of rationalizing and taking actions that will give it the best chance for accomplishing a specific goal.

As a specific component of AI, machine learning (ML) is associated with the notion that computers can automatically learn from new data and adapt to it without human intervention. Statistical methods and computer science field are used to train algorithms to be able to make classifications or predictions in data mining projects, which can lead to the discovery of crucial insights.

A typical algorithm for Machine Learning may be something like a simple linear regression, for example. Imagine you are a real estate agent and your goal is to predict the price of houses in an area based on multiple features such as number of rooms, area, neighborhood, etc. First step would be to start by collecting a dataset of housing prices with their features. Then, you would build and train a linear regression model on the data, by using the feature values to predict the prices. The model would try to learn the relationship between the features and the prices, and as a result you would be able to use this relationship to predict new, unseen house prices. For example, if you have a house that has 3 rooms, an area of 1500 square feet, and is located in a desirable neighborhood, you would be able to use the trained model to predict the price of this house. There are various machine learning algorithms already written. Scikit learn is a package containing many algorithms for regression or classification problems.

These algorithms are used in multiple businesses and scientific circles, especially in the economy. With the growth and expansion of big data, it is clear that the demand for data scientists will also increase in the future. Nowadays, almost all fields try to take advantage of machine learning models to reduce the costs and times of computations. The use of machine learning models in geophysics is not a new thing too.

The terms machine learning and deep learning are often used interchangeably, however, it's important to understand the differences between the two as well. Neural networks, Deep learning, and Machine learning are sub-sets of artificial intelligence. The field of neural networks is a branch of machine learning, whereas the field of deep learning is a branch of neural networks.

Deep learning algorithms, like all machine learning algorithms, train computers by feeding them with data and making them find patterns by showing them input and output variables. The difference between DL methods and other ML ones is that DL uses neural networks to train models. Neural networks designed in an attempt to mimic the biological neural networks in the brain, providing a far more powerful process of learning compared to standard machine learning models. A typical NN consists of an input layer, hidden layers and output layer (figure 9). In the training set, the values of the middle layers aren't observable, so the layers are called hidden layers. Basically, a hidden layer is a calculated value that is used by the network to perform its "magic". As the number of hidden layers inside the input and output layers increases, the deeper the network becomes.

To understand how neural networks work, imagine we want to train a model which can classify human body parts. As a first step, the NN identifies the relevant characteristics of the body parts, also known as features. A feature can be a specific structure in an image, such as a point, an edge, or an object. Machine Learning algorithms require software engineers to select relevant features, but NNs can automatically select features. A first hidden layer may learn how to see edges, a second layer may learn to be able to distinguish colors, and the third layer may learn to detect more complex shapes according to the object's shape. Being fed with training set, the DL algorithms, at the end, will learn from errors whether the prediction was fine, or whether it should adjust.

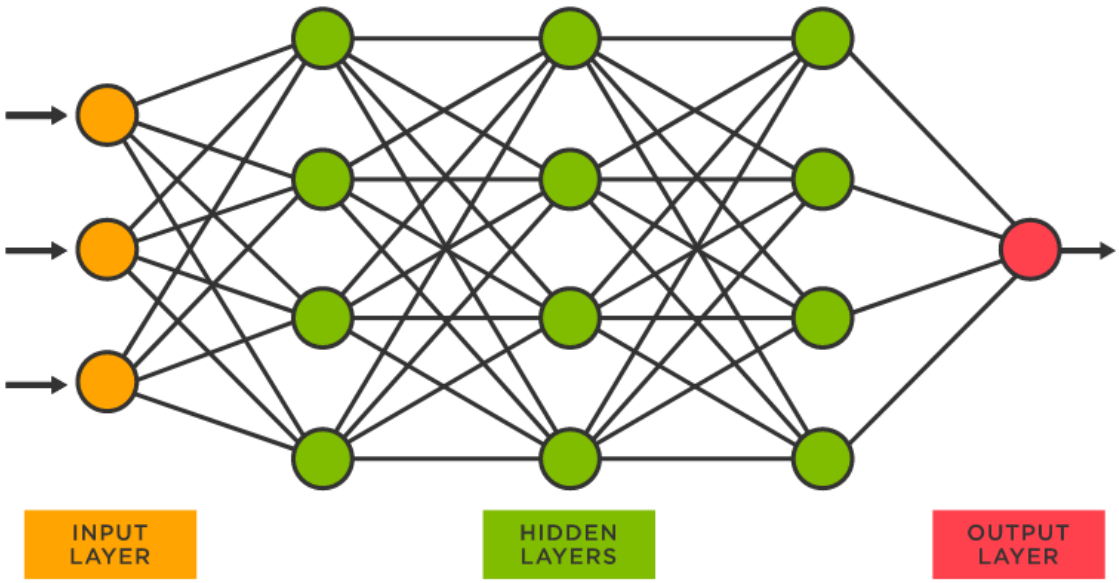


Figure 9, Structure of Neural Networks

Now when we have general understanding of what deep learning is, I would like to use it. In order to perform deep learning on a problem it is needed to go through 7 steps (figure 12):

1. Data acquisition
2. Data cleaning (preprocessing)
3. Splitting the dataset in to training and testing sets
4. Building and training the model
5. Evaluating
6. Hyperparameter tuning

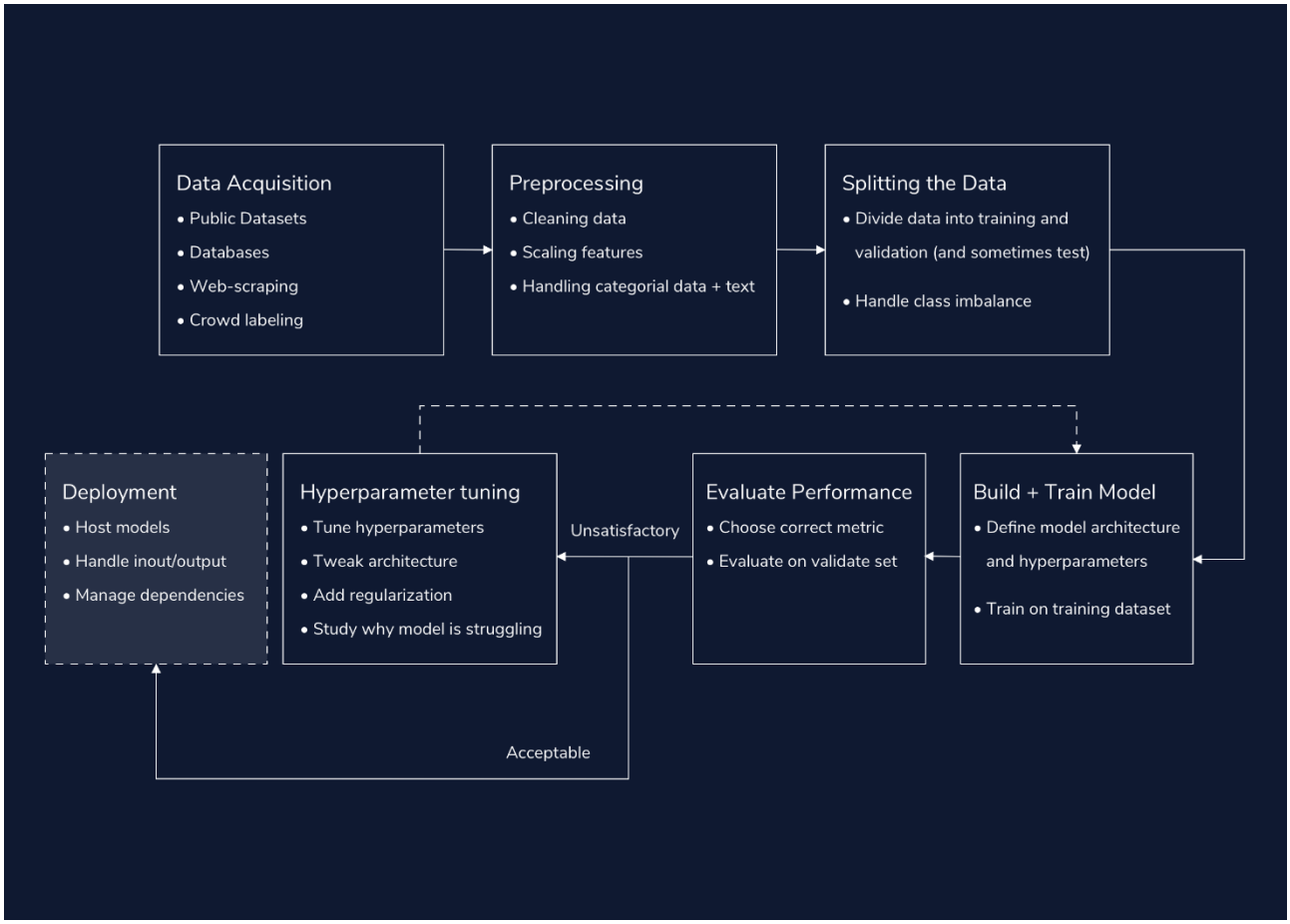


Figure 10, Deep Learning workflow (team s.d.)

Obviously, in order to make the model work and make predictions, we need to first feed it a data. Data can be acquired in various ways, like Web Scraping, public datasets or creating your own. Python is the best programming language to do all the steps mentioned and train machine learning or deep learning models. The best tool for creating deep learning models is Google's Tensorflow and Keras modules. Another very strong tool is Numpy and Pandas. These are the best packages in Python for doing mathematical operations and manipulations with datasets. Python has all the tools to upload acquired data and work with it.

Most of the time the acquired data is untidy and it needs to be processed, so it can be best used for feeding the model. Another part of data preprocessing is rescaling the data. Input features with large values will cause our models to struggle due to the fact that neural networks are initialized with small weights to stabilize training. Therefore, we often normalize and standardize real-valued features so that they have a mean of zero and

variance of one: that is, we normalize them between 0 and 1. Both methods were used in the model.

If we train whole dataset and then verify it on the same dataset, we obviously will get higher results than how the model is actually accurate. To remove this problem, some of the dataset (generally 20%) is used as test (or validation) data set.

To build a model it is needed to import Tensorflow and Keras api to Python. As mentioned, before they are the best tools to create neural networks and improve them. I will give more information about the bodies of the model in the section where I explain the model created, but the basic thing needed to be known is that a NN model is created by giving the number of layers, the number of hidden units (neurons), input and output variables, loss function and an optimizer to minimize the loss. When we feed the model with a dataset and show the input and output variables, a loss function is required to the model in order to be able to minimize it. By also giving and optimizer function we let the model to find relationship between input and output variables in the best way available.

We evaluate the model's performance on our test set every time it is trained. Based on our performance on the test set, we can determine how our model will perform on new, untested data. The right metric is essential when evaluating performance.

It is almost always necessary to iterate upon our initial hyperparameters several times before we have the optimal value. Our model is trained and evaluated by experimenting with a variety of learning rates, architectures, batch sizes, and regularizations as part of the training process. These are hyperparameters of the models. When training and testing a new model, it is common practice to start with a smaller model and increase the hyperparameters until the training and testing performance diverges, which means we have to overfit the model to the data.

In the end, if we get satisfactory results, we can deploy the model.

Nowadays various Deep Learning algorithms are being developed. The most famous ones are:

1. Convolutional Neural Networks (CNNs)
2. Recurrent Neural Networks (RNNs)
3. Generative Adversarial Networks (GANs)
4. Autoencoders
5. Deep Belief Networks (DBNs)
6. Multilayer Perceptrons (MLPs)
7. Self-Organizing Maps (SOMs) and etc.

The first 4 algorithms have been applied in various seismic inversion problems (A. Adler 2021). In the model Convolutional Neural Networks have been used. To better understand how these models differ from typical neural networks, the next chapter will discuss these methods.

Chapter 4: Materials and Procedures

In this chapter Convolutional Neural Networks will be described before introducing you the model created for porosity and acoustic impedance estimation.

Chapter 4.1: Convolutional Neural Networks

In deep learning, Convolutional neural networks (CNN) are network architectures that learn directly from data. CNNs are especially useful for identifying objects in images by finding patterns within them (Patel et al., 2020). In addition to classifying image data, they can also effectively classify other types of data, such as audio, signal data and time series data (M. Mohammadi et al., 2019). They work by applying a set of filters, called kernels or filters, to the input data to extract meaningful features. These filters slide across the input image, computing the dot product between the filter values and the corresponding pixel values in the input data. The result of this operation is a feature map that represents the responses of the filter to different parts of the input image.

After the convolution operation, the feature maps are down-sampled using a pooling operation, which reduces their spatial resolution and reduces the computational cost of the network. This is done to make the predictions way faster and reduce the load. The resulting feature maps are then passed through multiple fully connected layers, which use weights to combine the features into a final prediction. Finally, an activation function is applied to the output of each neuron in the fully connected layer, introducing non-linearity into the network.

During training, the weights of the network are updated to minimize a loss function that measures the difference between the predicted output of the network and the true output. This process allows the network to learn the most meaningful features for the task at hand.

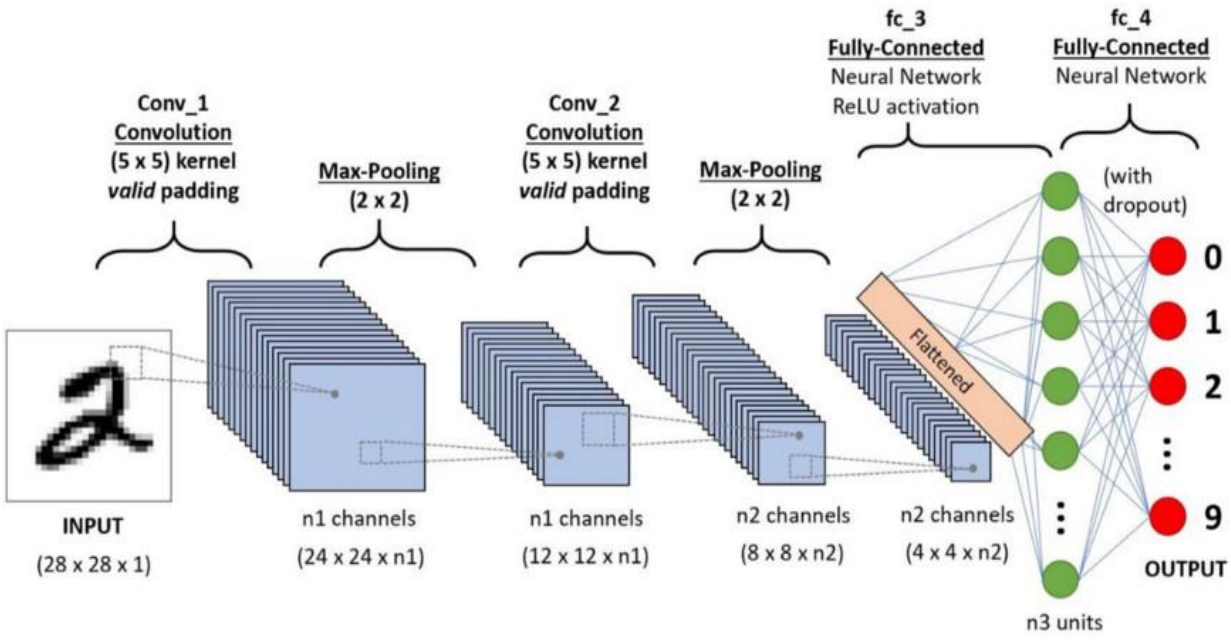


Figure 11, general architecture of CNN models (Artur Luczak 2022)

Creating a Convolutional Neural Network (CNN) model involves the following steps:

- Kernel (or filter)
- Stride
- Padding
- Convolutional Layer
- Pooling Layer
- Flatten
- Fully Connected (Dense) Layer
- Activation Function
- Dropout
- Loss Function
- Optimizer

In Convolutional Neural Networks (CNNs), a kernel (also known as a filter) is a matrix of weights that is used to extract features from an input image. During the convolution operation, the kernel slides over the input image, multiplying the entries of the

kernel with those of the input and then summing up the results. This process is repeated for every possible position of the kernel over the data, resulting in a feature map that represents the responses of the kernel to different parts of the input image (figure 14).

The kernels are learned during the training process, allowing the CNN to automatically identify meaningful features in the input image. The number of kernels determines the depth of the feature maps created, with each kernel producing a separate feature map. These feature maps can be passed through other CNN layers to further extract high-level features from the input image.

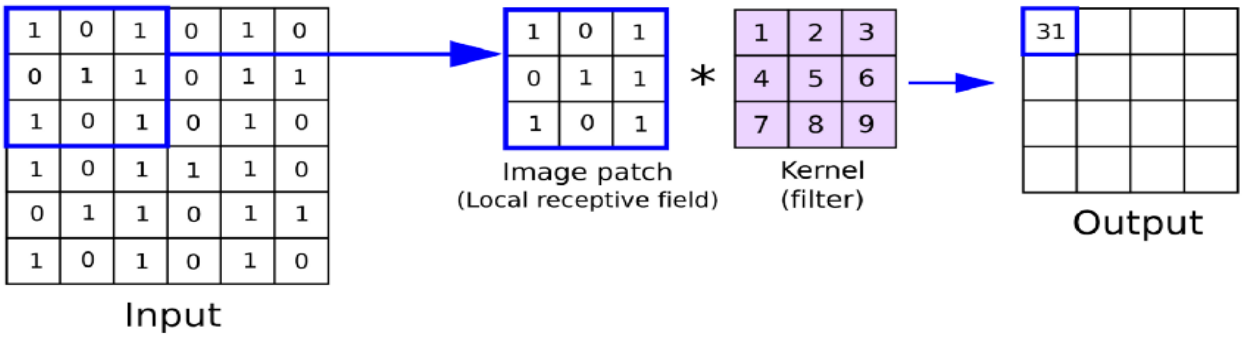


Figure 12, 3*3 kernel is used to extract features from the input data

A neural network's filter has a parameter called stride which controls the amount of movement in the input by the filter. The stride determines the spatial size of the output feature map, and therefore has a significant impact on the computational cost and the spatial resolution of the feature maps. Typically, the stride is equal to 1, which means that the kernel moves one pixel each time, covering all parts of the input image. However, setting the stride to higher values (e.g., 2 or 3) results in reducing of the spatial resolution of feature maps, as the kernel covers only every second or third pixel of the data. This reduction in spatial resolution is often used to decrease the computational cost of the convolution operation and to reduce overfitting, as it results in a reduction of parameters in the network (figure 15).

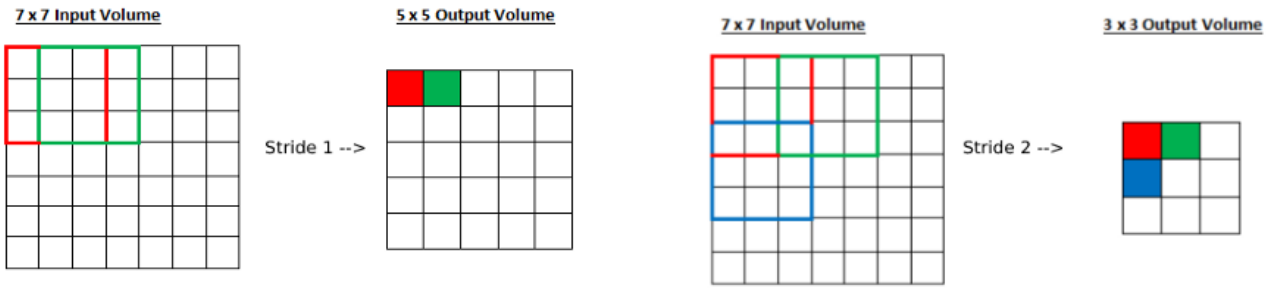


Figure 13, In first example kernel slides by 1 value of input and in the other by 2

In convolutional neural networks, padding is the number of pixels that are added to an input during the processing of the filter. As an example, if the padding value in a CNN is picked as zero, then every pixel value that are added will be equal to zero. As a result of using the filter to scan the image, the size of the image is getting reduced. In order to extract some low-level features, we must avoid this since we wish to preserve the initial size of the input. For this reason, padding is used and we will add some pixels outside of the image (figure 16).

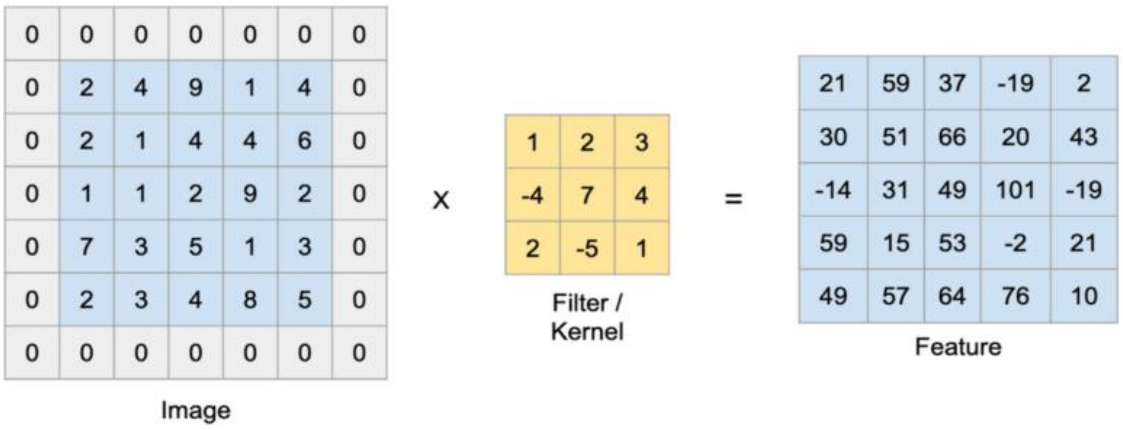


Figure 14, Padding

The process of flattening involves converting all of the final 2-dimensional arrays from pooled features maps into a single long linear vector. Using the flattened layer as

input, the fully connected dense layer can classify the image.

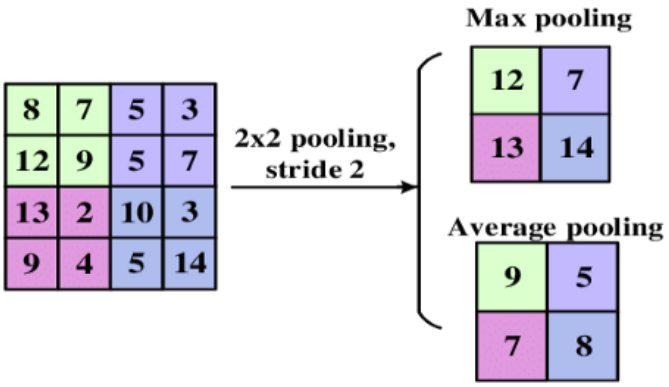


Figure 15, Different pooling techniques

When we create a neural network model, we first start to create layers and set their parameters, described before. Convolutional layer is first layer which is used find features from inputs. The next type of layer is fully connected layer. And finally, there is a pooling layer. It is primarily used to decrease computational costs by reducing the size of the convolved feature map. In figure 17, we can see how max and average pooling technique reduces the size of feature map. Another typical feature of CNN is Dropout layer. It is a mask that removes the impact of some neurons on the next layer and is leaving unmodified all others (figure 18).

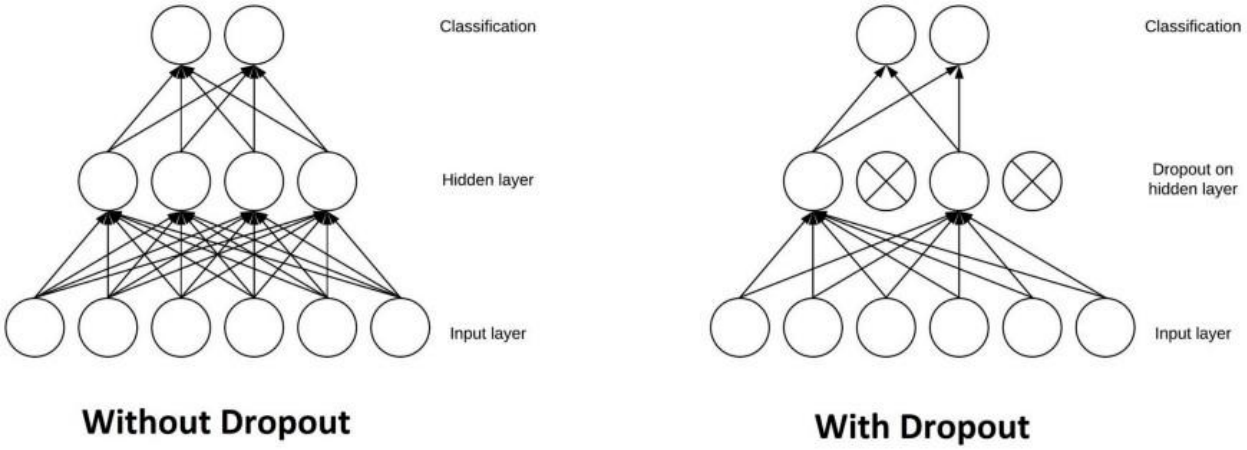


Figure 16, In the picture dropout rate is 0.5, meaning half of the neurons are dropped

In the end, we have activation functions that decide if neurons will be activated or not. Mainly sigmoid, tanh, Relu and softmax activations are used.

Here is a simple example of a CNN model:

```python
from keras.models import Sequential
from keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout

# Define the model
model = Sequential()

# Add the convolutional layer
model.add(Conv2D(32, kernel_size=(3, 3), activation='relu', input_shape=(28, 28, 1)))

# Add the pooling layer
model.add(MaxPooling2D(pool_size=(2, 2)))

# Flatten the feature maps
model.add(Flatten())

# Add the fully connected layer
model.add(Dense(128, activation='relu'))

# Add dropout to prevent overfitting
model.add(Dropout(0.5))

# Add the output layer
model.add(Dense(10, activation='softmax'))

# Compile the model
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
```

In this example, the first layer is a convolutional layer with 32 filters of size 3x3, ReLU activation, and an input shape of 28x28x1 (the last dimension corresponds to the number of color channels). The second layer is a max pooling layer with a pool size of 2x2. The feature maps are then flattened and passed through a fully connected layer with 128 neurons and ReLU activation. Dropout with a rate of 0.5 is applied to prevent overfitting. Finally, the output layer has 10 neurons with a softmax activation function, as this is a multi-class classification problem. The model is compiled with a categorical cross-entropy loss function and the Adam optimizer.

In general, when we talk about Convolution Neural Network (CNN), we are referring to a 2-dimensional CNN that is used to classify images. However, it is also important to point out that there are two other types of Convolution Neural Networks, which are 1D and 3D CNNs. The dimensions here show through how many dimension kernels slide in a model.

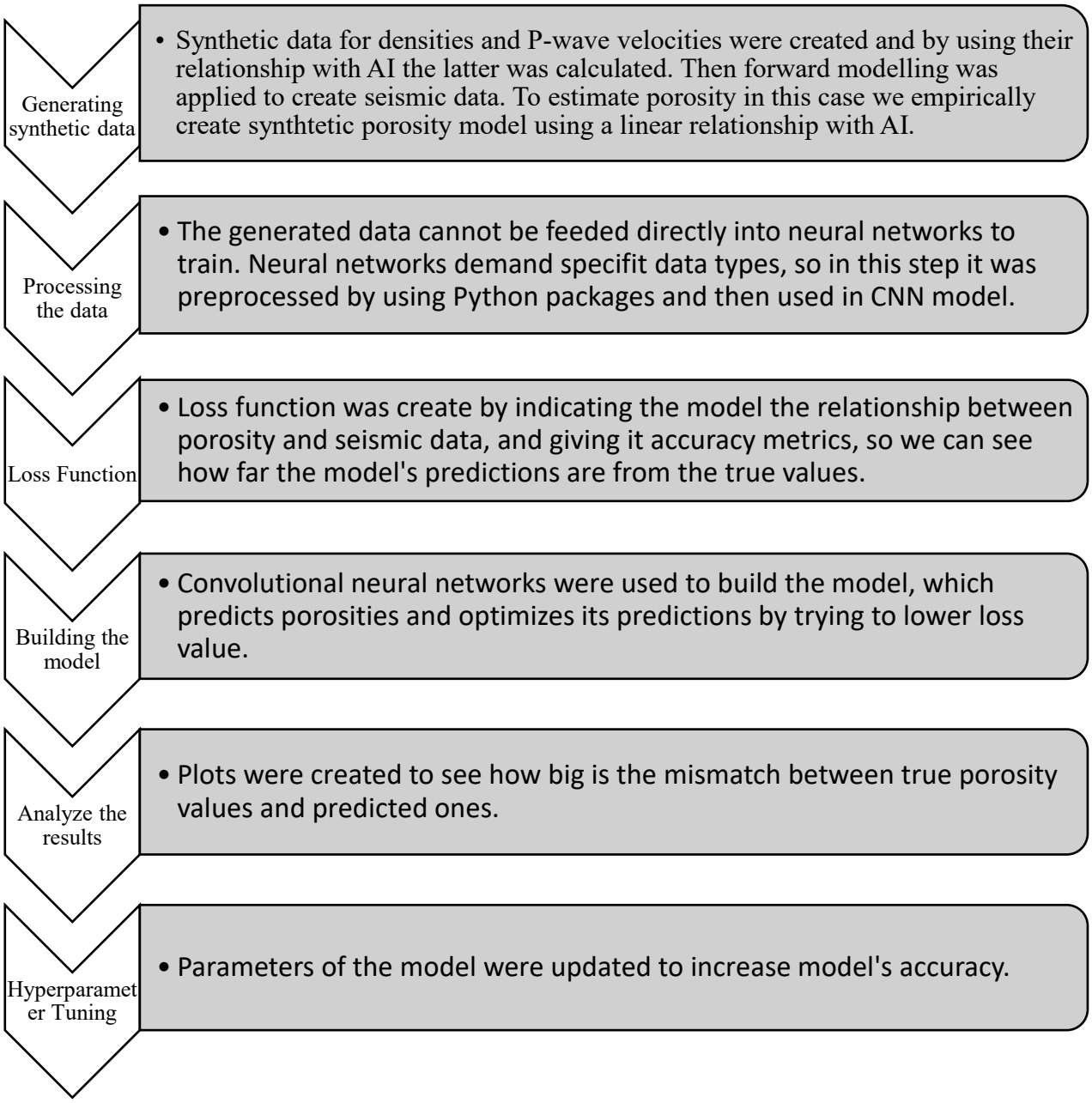
Originally introduced in Lenet 5 architecture, 2D CNN models are the standard Convolution Neural Network. It was the first CNN model and for many years this type of CNN models were used for mainly image classification, object detection and etc. Tesla's automotive cars exist due to this technology.

As we know in 1D Convolutional networks kernel slides through 1 dimension. The great use of it is time series data. In fact, in our model the data (seismic traces) is also representing time series. This is the reason for choosing this type of a model over other neural networks.

3D CNNs are used in videos and 3D images, like tomography or MRI scans.

Chapter 4.2: Methodology

The main idea behind this model is to give seismic data, wavelet and prior model as an input and estimate porosities from it. The workflow in creating the model was the next:



The process of creating a model for estimating porosity from post-stack seismic data began with the creation of a synthetic seismic trace using equations in 3, 4 and 5. To do this, a set of prior and absolute P-wave velocities and densities were created as Numpy arrays

(figure 17(a)), and acoustic impedance (AI) was calculated using these values and the formula provided (figure 17(b)). Numpy has the function to create an array with random numbers given minimum and maximum range and desired mean value. So P-wave velocities and densities were generated by using this function. Average P-wave velocity was set to 4200 m/s and density to 2280 kg/m3. Reflectivities were then calculated from AI using relationships between them. The synthetic seismic trace was created by convolving the reflectivities with the wavelet (figure 17(b)) and adding noise (figure 17(c)). Wavelet was representing a general Ricker wavelet (figure 18(b)). This was created by using physics behind it (figure 17(a)). Time interval was chosen as 4 ms and total time as 250 ms. The resulting seismic data looks like in figure 18.

a)

```
# input data
wvlt_amp = np.load('wvlt_amp.npy')

Vp = np.load('Vp_t.npy')
Den = np.load('Den_t.npy')

Vp0 = np.load('Vp0_t.npy')
Den0 = np.load('Den0_t.npy')

#%% setting the parameters
dt = 0.004
nt = len(Vp_t)
#depth to time conversion
T = dt * np.linspace(0.0, nt-1, num = nt)
```

b)

```
AI = Vp * Den * 10 ** (-7)
AI0 = Vp0 * Den0 * 10 ** (-7)

ref = (AI[1:] -  AI[:-1]) / (AI[:-1] +  AI[1:])

S_t = np.matmul(wvlt_map, ref)

S_t = np.reshape(S_t,[-1,1])

S_t_plus = np.zeros((1,1))

S_t = np.vstack((S_t_plus, S_t))
```

c)

```
# add noise to seismic
Noise = 1
SNR = 30

if Noise == 1:
    for i in range(PP_t.shape[1]):
        SignalRMS = np.sqrt(np.mean(np.power(S_t[:, i], 2)))
        NoiseTD   = np.random.randn(len(S_t[:, i]),1)
        NoiseRMS  = np.sqrt(np.mean(np.power(NoiseTD, 2)));
        New = np.reshape(S_t[:, i], (-1, 1)) + (SignalRMS/NoiseRMS) * np.power(10, -SNR/20) * NoiseTD
        S_t[:, i] = New[:, 0]
```

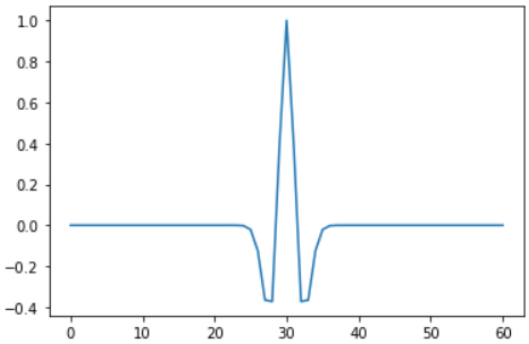
Figure 17, a) generated wavelet, velocities and denisites are imported as numpy arrays, b) AI, reflectivities and seismic trace are calculated using exact physics, c) noise is created and added to the seismic trace

```python
def ricker(f, length, dt):
    t = np.linspace(-length / 2, (length-dt) / 2, int(length / dt))
    y = (1.0 - 2.0 * (np.pi ** 2) * (f ** 2) * (t ** 2)) * np.exp(-(np.pi ** 2) * (f ** 2) * (t ** 2))
    return t, y

# Define the parameters for the wavelet
f = 40.0
length = 0.250
dt = 4.0 / 1000.0

# Generate the time and wavelet samples
t, y = ricker(f, length, dt)
```

a)



b)

Figure 18, a) Python script for creating zero phase 40 Hz Ricker wavelet, b) Resuling wavelet

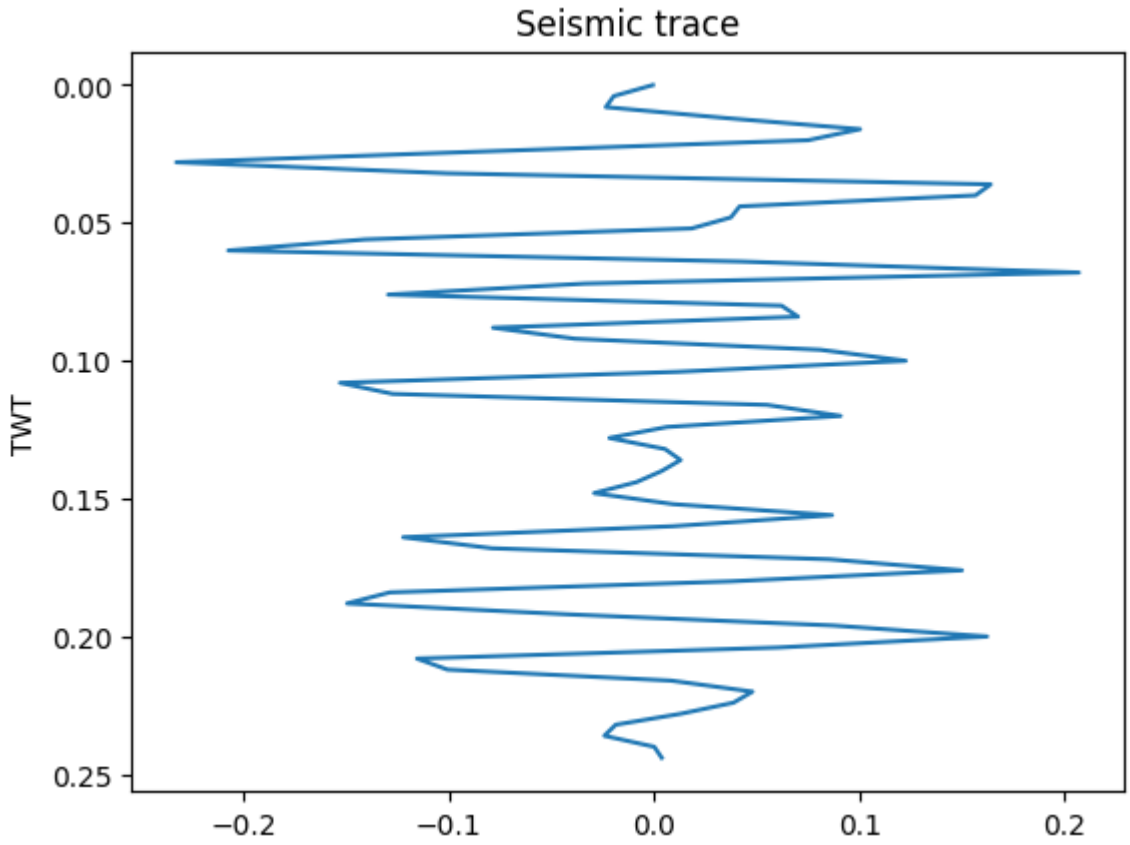


Figure 19, Synthetic seismic trace

Porosity may be determined by various methods. Some are based on core samples; others on well-log prior data and mathematical models. In deterministic methods the one used are techniques of porosity estimations from transit time analyses that make use of interval velocities gained from seismic traces (Alabi and Enikanselu, 2019). It is expected that acoustic impedance shows some correlation with porosity as proposed by Raymer et al. (1980) and the relationship between porosity and P-wave velocity is as follows:

$$V_p = ((1-\varphi)^2 * V_{ps}) + \varphi * V_{pf} \quad (6)$$

Where V_p is P-wave velocity and φ is porosity

The mass balance density and porosity equation is:

$$\varphi = (\rho_{log} - \rho_f)/(\rho_m - \rho_f) \quad (7)$$

To create a relationship between AI and porosities an empirical relationship was created for the synthetic model. In real case studies (figure 20) this relationship can be extracted by using deep learning models, instead of using linear regression models which would be based on assumptions and would not suit if the relationship is strongly non-linear. For simplifying the problem and to demonstrate the abilitty of deep learning models to extract multiple features at the sama time we assume, that there is known empirical linear relationship between AI and porosities in our synthetic seismic data.

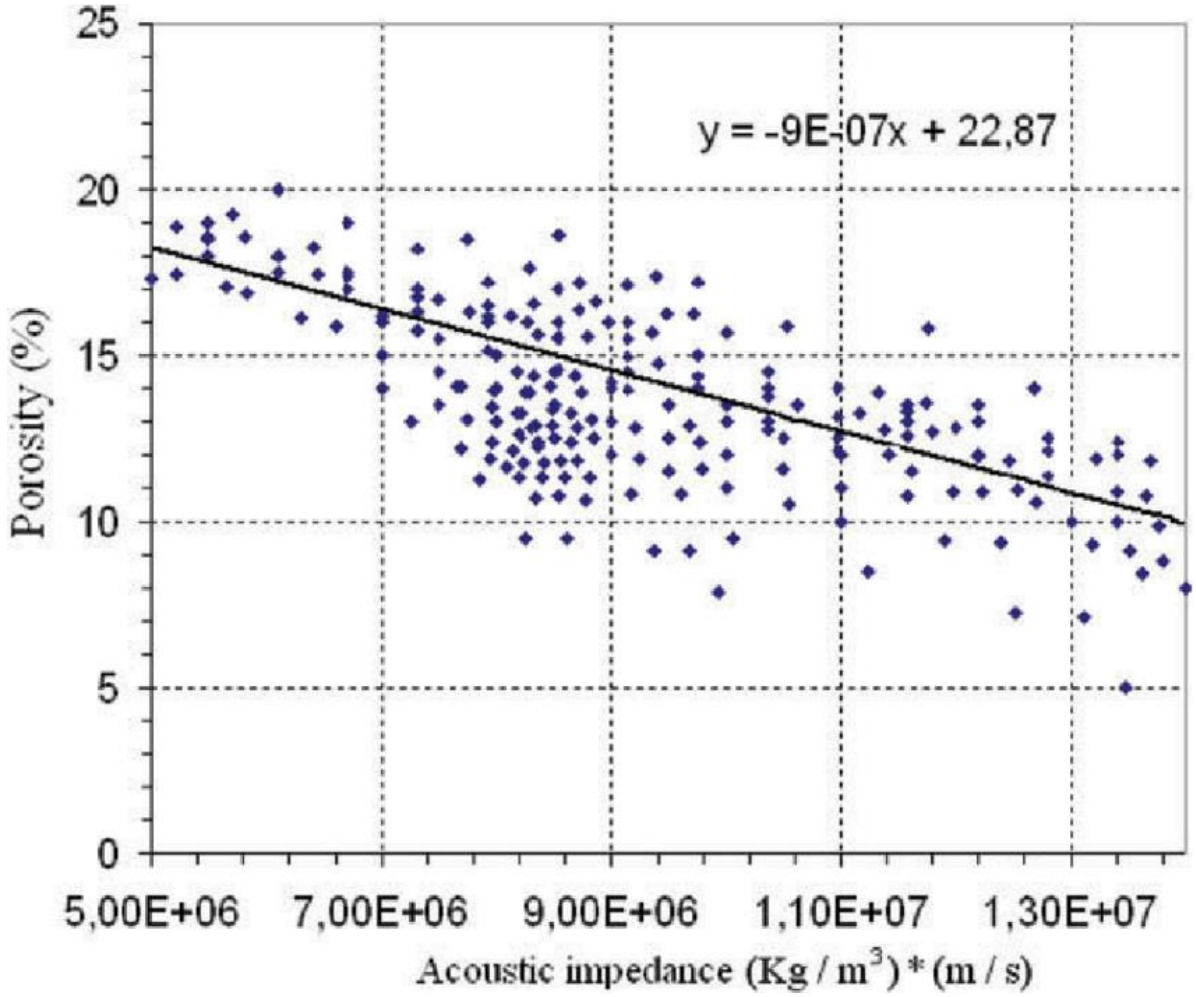


Figure 20, Linear regression model used for estimating the relationship between AI and porosity for Predicting Porosity of Carbonate Reservoir in a South Iranian Oil Field (Jalalalhosseini et al., 2015)

Data processing was held to make the inputs compatible with the model. To reshape the data tensorflow and numpy packages were used. The "expand dims" function was used to add an additional dimension to an array in order to increase its number of dimensions, which is necessary when working with Convolutional Neural Networks (CNNs).

The next part was creating a loss function. An objective here was to create an error metrics, which would show how much our predicted data is far from our original data (figure 21). After the model gives its first prediction on porosity values it then uses an empirical relationship between predicted porosities and acoustic impedance and predicts AI. Then it calculates reflectivities and convolves them with wavelet to create a predicted seismic trace. Finally, loss function compares predicted and actual seismic data and gets sum of mean squared errors between them, and also between prior porosity and predicted absolute porosity. The loss function was created in a way which would punish also big mismatches between predicted porosity and prior one. In this way better results were observed in prediction accuracy. And the loss function enables the model to be unsupervised and reduces the need to have abundant amount of labeled data. In machine learning there are supervised and unsupervised learning methods. In supervised we feed the model labeled data and then it compares its predictions with it. In unsupervised we don't give it labeled data and instead it tries to label it itself. In our case we don't give the model true porosity values to learn on it, but give only seismic data, wavelet and prior porosities.

```
def loss_fn(y_true, y_pred):
    Porosity_predict = Prior_porosity_tf + y_pred

    AI_predict = 5 * Porosity_predict + 0.3

    ref_predict = (AI_predict[1:] -  AI_predict[:-1]) / (AI_predict[:-1] +  AI_predict[1:])

    seismic_trace_predict = tf.matmul(wvlt_map_tf, ref_predict)

    seismic_trace_misfit = 100 * tf.reduce_mean(input_tensor=tf.square(seismic_trace_predict - y_true[1:]))
    porosity_misfit = 1 * tf.reduce_mean(input_tensor=tf.square(Porosity_predict - Prior_porosity_tf))
    return seismic_trace_misfit + porosity_misfit
```

Figure 21, Loss function

After the model was given an error metric on what it will optimize its predictions, the model was being builded. The general rule in model building is to start with the simple models and then add hyperparameters if the accuracy is not sufficient. To start with 1 1D Convolutional neural network was created with filters, kernel size equal to 1 and kernel and bias initializers was set 'Glorot uniform' and 'zeros'. Then the dropout layer was added followed by flattened layer and final dense layer. Generally the most useful activation function is Relu, so at the first attempt this function was used in all layers.

Here after we set number of filters, the model tries to extract features with slicing through input data by kernel size. Kernel size was set to 1, so it applies filters to each sample and extract features from it. Model initiates random weights, with a technnique specified with kernel initializer, then multiplies it by the filters and adds by bias, specified with bias initializer. The number of filters specifies how many neurons we want to create. If we set it to 12, then the model would create 12 neurons and do mathematical operations in (10) to predict a value and proceed to the next kernel. 1D convolutional neural networks suits our problem well because it lets filters slide in 1 dimenion, in our case through each seismic trace. Glorot uniform is standard kernel initializer and it is believed to be the most optimal for most of the models. Padding was set to valid, to prevent reduction of the input size. Filters at first attempt was chosen as 12 as it is recommended when creating the first simplest model.

$$y = \sum(weights * input + bias) \tag{8}$$

The training data for large neural networks can be overfit using relatively small datasets. When the model is validated on new data, e.g. a validation dataset, it performs worse due to learning the statistical noise in the training data. To reduce this problem dropout layer was used, as our dataset is small and prone to over generalizing. Rate, which defines which percent of neurons will be dropped was randomly picked as 0.25.

After a convolutional layer, batch normalization layer in order to do mathematical operations we give it activation function. General work principle of deep learning with use of activation functions is shown in figure 22.

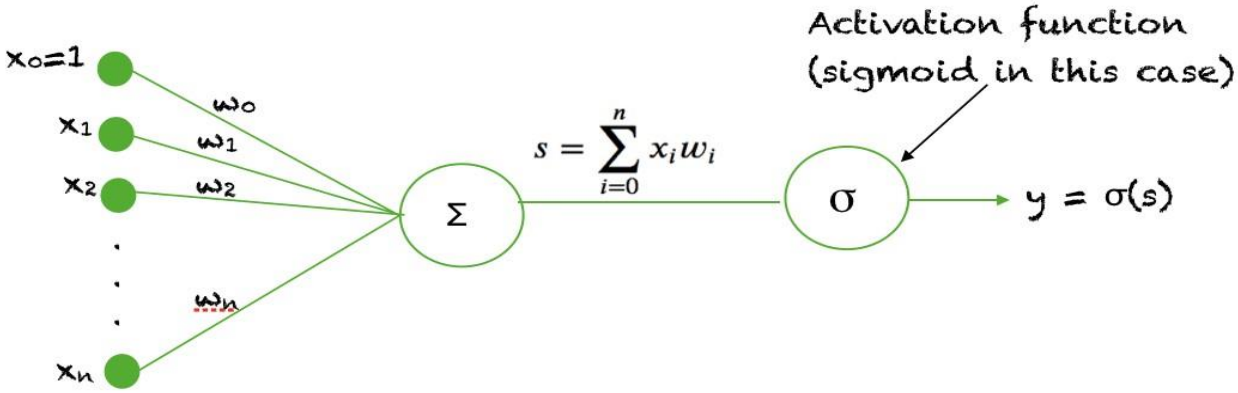


Figure 22, An activation function in a neural network

The last types of layers used in the simple model were a Dense and Flattened layer. Dense layers in neural networks are deeply connected to their preceding layers, which means their neurons are connected to all neurons in the preceding layers. The input data in our model is 2D data and when we use Convolutional neural networks, they do operations in 2D matrices. However, in our case the output data should be 1D porosity data, so we flatten evaluations from CNN operators and then using Dense layers try to get exact shape of outputs.

The simple model written is shown below.

```
model = keras.Sequential()
model.add(layers.Conv1D(filters = num_filters, kernel_size=1, strides=1, padding = 'valid', kernel_initializer = tf.compat.v1.ker
model.add(layers.BatchNormalization())
model.add(layers.Activation('relu'))
model.add(layers.Dropout(rate = 0.25))
model.add(layers.Flatten())
model.add(layers.Dense(units = 1, kernel_initializer = tf.compat.v1.keras.initializers.VarianceScaling(scale=1.0, mode="fan_avg"
model.add(layers.Activation('relu'))
```

Keras has very powerful API. It reduces the need to write low level python scripts manually to create all the sequences to make estimations and optimize it. Here first sequential model was created. It lets us to add layers on top of other layers.

After the model created it is time to evaluate it. In comparison with other deep learning libraries like Pytorch it is way easier to do. First, we need to compile a model, where we give it loss function and the way to optimize it. After we fit the model to our input and output data and check its accuracy. Epochs show how many times we want to iterate the model to train over the data. More epoch can show greater results, however it will also longer time to run. Batch size is picked as length of each seismic trace. One more synthetic data was created to validate our model. Model doesn't train on validation set, it just uses trained model on the dataset, so if we get high accuracy on bot validation and training examples then the model works well and is not overfit or underfit.

```
model.compile(loss=loss_fn, optimizer='Adam',metrics=[keras.metrics.MeanAbsoluteError()])
```

```
model.fit(X_train, Y_train, epochs=1000, batch_size=62,shuffle=False,validation_data=valid)
```

The results of the first model were not great. It showed big losses, however the model could learn from each iteration and showing potential to work. In figure 23, predictions from 10, 100 and 1000 iterations for both trained and validated data is shown. With Relu activation function and Adam optimizer the model adjusted weights and activated them in a way to minimize loss through approaching it to prior porosities. This is due to loss function depending on also the difference between absolute and prior porosities. In later iterations it tries to minimize also the mismatch between our seismic trace and the one created by the model. In the next chapter hyperparameter tuning will be discussed and the final results will be given.

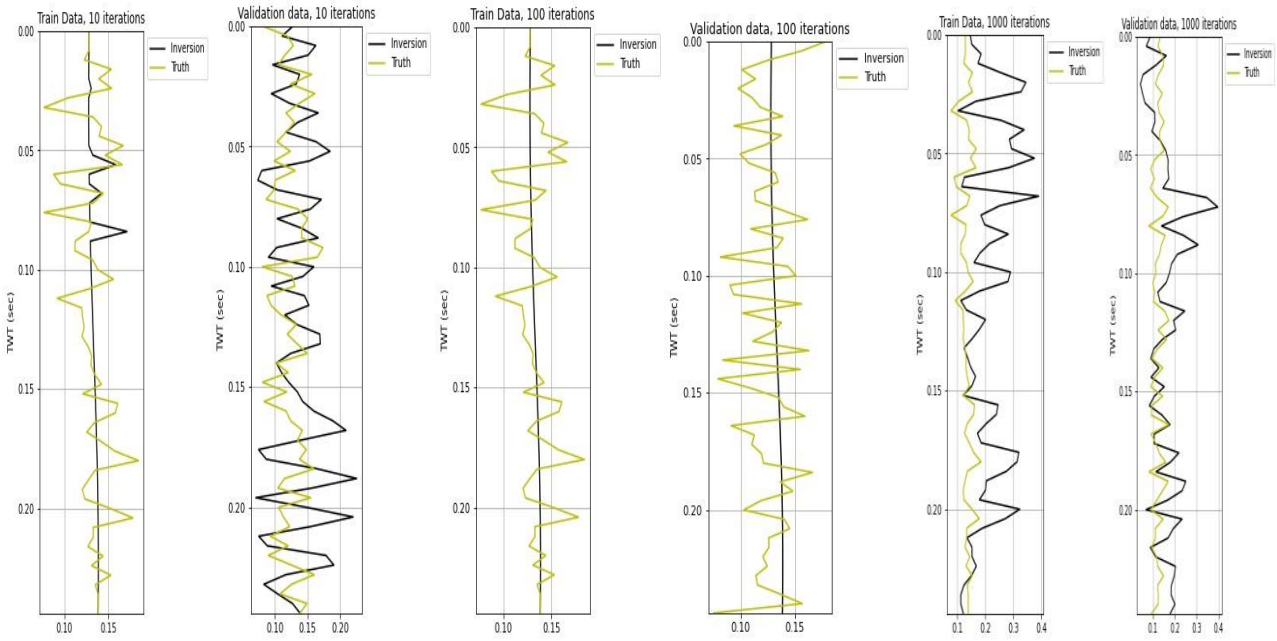


Figure 23, Mismatch between predictions from the model in 10, 100 and 1000 iterations and real values

Chapter 5: Results and Discussions

When a simple model, doesn't produce predictions with acceptable accuracy, hyperparameter tunings are needed to be done. The first step in tuning the model was playing around with activation functions. When the activation function was changed from rectified linear unit to hyperbolic and then to the sigmoid function very big changes in accuracy were observed.

Rectified linear unit function (Relu) returns values greater than zero as they are and as zeros if they are negative. This activation function is the most commonly used in convolutional neural networks and it suits most problems well. However, one of the main drawbacks of this activation function is its incompatibility with batch normalization. When batch normalization was removed the model gave a good prediction with Relu on training data (figure 24), however, to reduce the computational time of the model batch normalization is needed in the model. The input data used in the work is small, however if much bigger data is given, a seismic trace thousand times longer than it, then batch normalization would have huge impact.

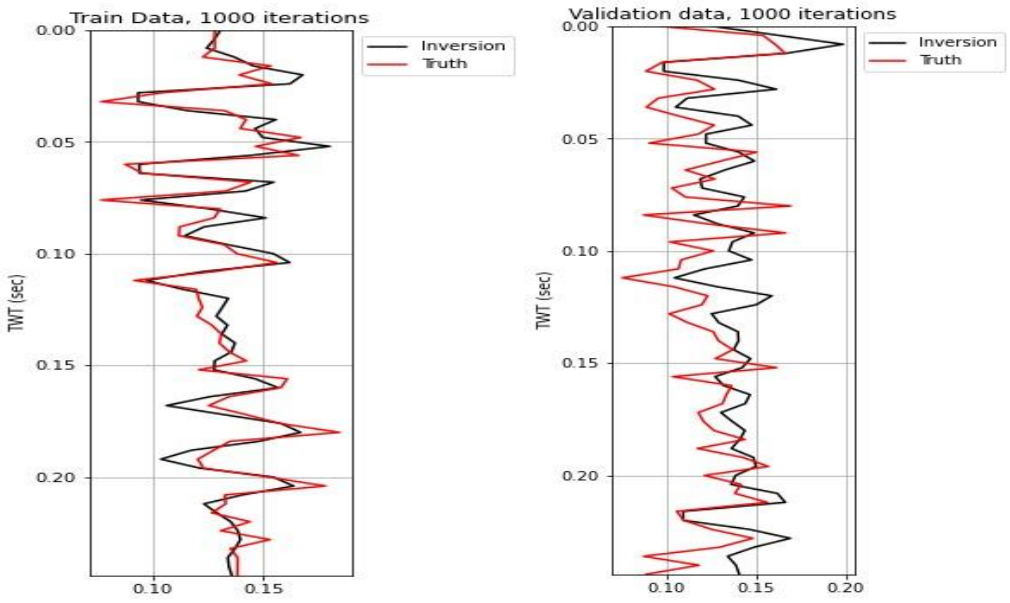


Figure 24, Accuracy of the model's prediction with Relu activation, without batch normalization

In the second attempt hyperbolic activation function was used. This activation function uses the next equation for activating the layer's inputs:

$$\sigma(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \tag{7}$$

After the hyperbolic function activates the inputs from layers the results are always between -1 and 1. This suits our problem well, as relative porosity can take values in the same range. This makes the learning process of neural networks easier and much faster. For this reason, this activation function was kept to activate the final dense layer to give sensible prediction from the earliest epochs. When used in hidden layers the predictions were the poorest compared to other used activation functions (figure 25a). The reason for this was that the layers took too much negative numbers, even though minimum value it gives as an output was negative 1 it is still too much for our desired output. In very big iterations it was capable of learning and giving good predictions, however speed of training the model is important. When used with Relu activation and without batch normalization it gave better results than with solely Relu activation (figure 25b).

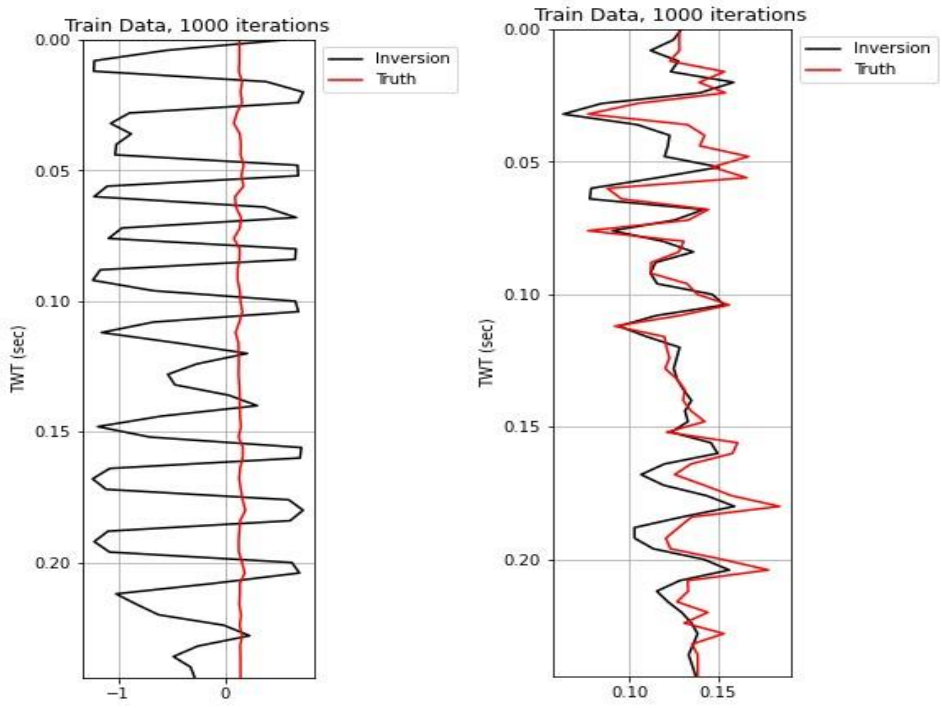


Figure 25, model's predictions in 1000 iterations with a) only tanh activation, b) relu + tanh

In the attempt sigmoid functions were tried. Sigmoid function uses next equation to activate layers:

$$\sigma(x) = \frac{1}{1+ e^{-x}} \quad (8)$$

Sigmoid function, also called as logistic function, is mainly used in classification problems. It takes values between 0 and 1 as probabilistic values. If it outputs 0.8 then it is sure by 80% that the output is true. However, it suits our problem well, as it outputs values only in range between 0 and 1. Compared to tanh, where it gave negative numbers in hidden layers and made the learning process slower, sigmoid function could only give positive numbers, as rectified linear activation does, but in sensible for our desired output range. It was not used in final layer as relative porosity can be negative. It gave best and fastest predictions among other activation functions with batch normalization.

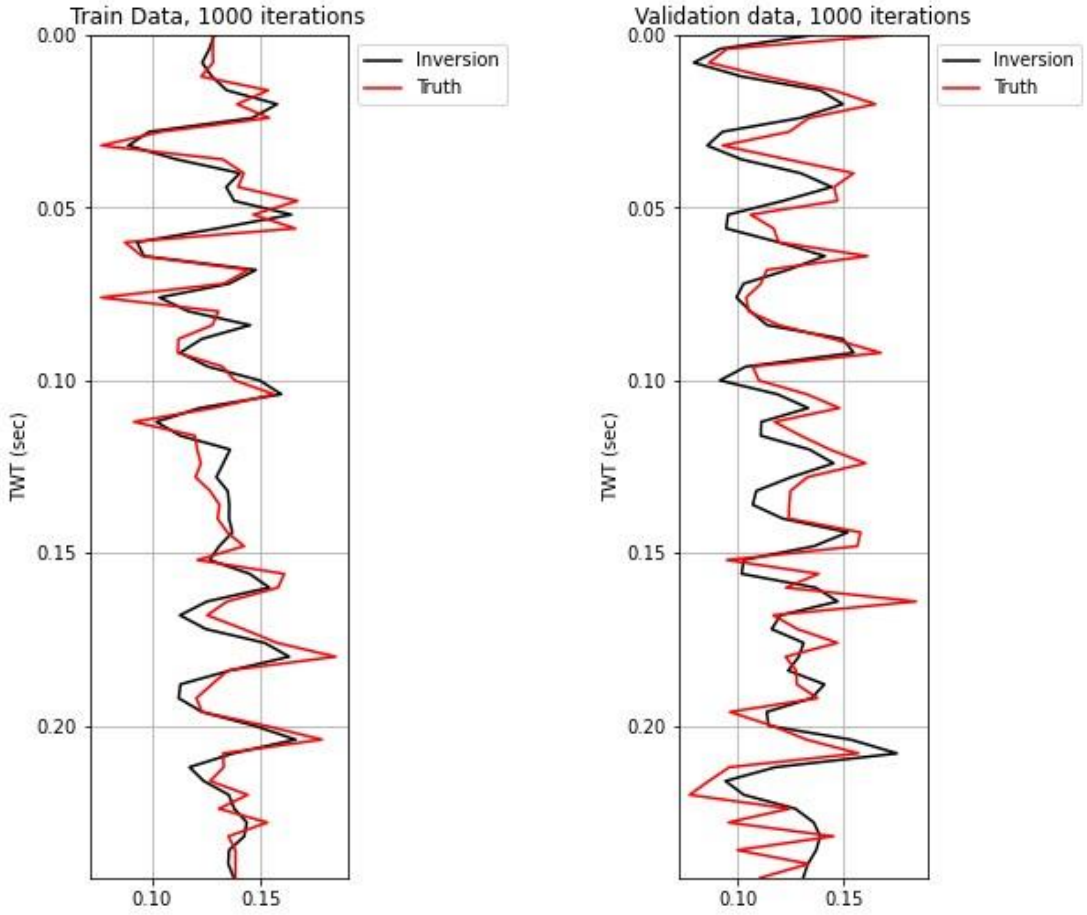


Figure 26, Model's prediction in 1000th iteration with sigmoid activation function

After the most optimal activation functions was found, the next step in hyperparameter tuning was playing around the numbers of layers and their parameters. The data used for training is not big, so very deep network is not needed

with many hidden layers. Thus, just a few additional layers were added to improve the model.

At first, 1 more convolutional layer was added, with the same parameters as in first and one more dense layer after flattened layer. After flattening all the 2D data collapses into 1D and we get many parameters and feeding them to one dense layer makes it hard to give right output. When 1 additional dense layer is added it gives the network to think one more time before giving the final output.

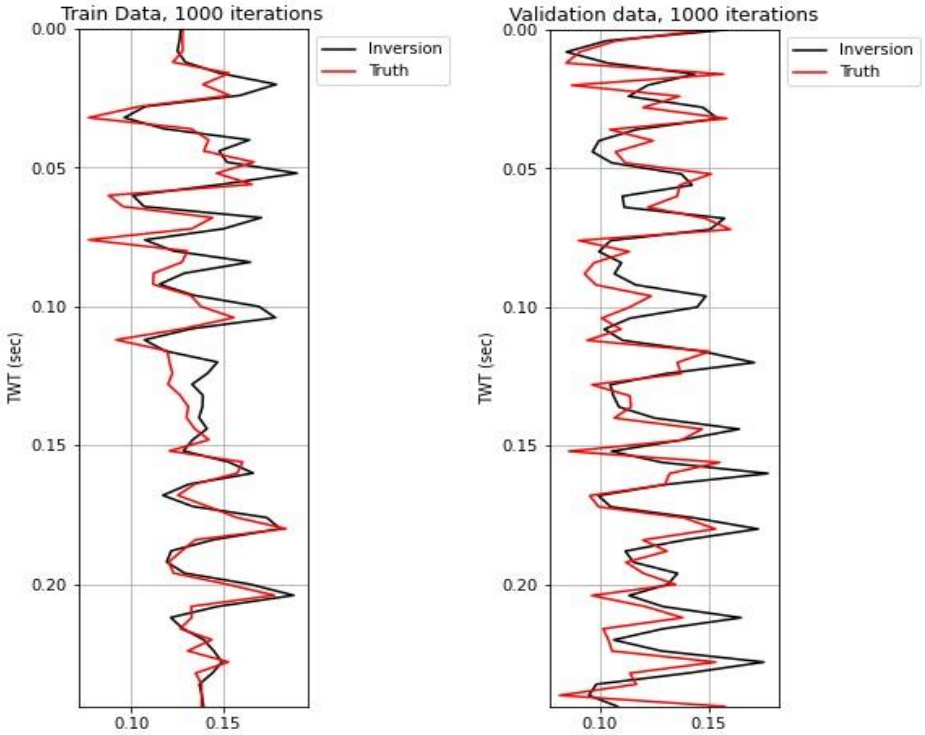


Figure 27, Predictions with additional layers

In the last step activation function of second convolutional layer was changed to Relu and the results were better and the number of iterations needed to train the model was less. If in other models 1000 iterations needed in the new model could give better predictions in twice less iterations, therefore decreasing the time needed twice.

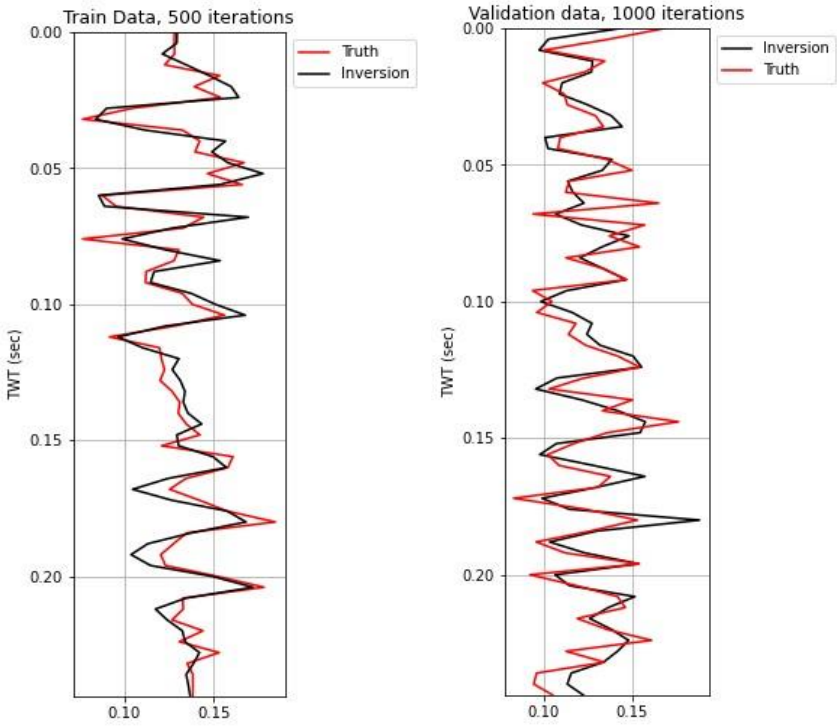


Figure 28, The model's accuracy when it consisted of 2 convolutional layers with sigmoid and Relu functions, and 2 dense layers with sigmoid and tanh activations.

The final model is shown below.

```
model = keras.Sequential()
model.add(layers.Conv1D(filters = 12, kernel_size=1, strides=1, padding = 'valid', kernel_initializer = 'glorot_uniform'))
model.add(layers.BatchNormalization())
model.add(layers.Activation('sigmoid'))
model.add(layers.Conv1D(filters = 24, kernel_size=1, strides=1, padding = 'valid', kernel_initializer = 'glorot_uniform'))
model.add(layers.Activation('relu'))
model.add(layers.Dropout(rate = 0.25))
model.add(layers.Flatten())
model.add(layers.Dense(units = 60, kernel_initializer = 'glorot_uniform', bias_initializer = 'zeros'))
model.add(layers.BatchNormalization())
model.add(layers.Activation('sigmoid'))
model.add(layers.Dense(units = 1, kernel_initializer = 'glorot_uniform', bias_initializer = 'zeros'))
model.add(layers.Activation('tanh'))
```

In the work the data was created synthetically by using physics laws, however, the model can be run with real data too. It needs seismic trace, wavelet, and prior model as inputs. With a low noise-to-sound ratio, the model can give exceptional results. Good preprocessing is needed; however, it can work with a little noisy data too. Another requirement for the porosity estimation is giving an empirical relationship between acoustic impedance and porosity. With enough data

available an empirical relationship can be found with machine learning algorithms or deep learning methods.

With bigger input data a deeper network will be needed to be created. Some more convolutional and dense layers should be added. The number of filters in each layer should be increased too. However, the general architecture of the model will be close to the one in the work. Keras enables the user to automate the process of tuning and the number of filters, kernels, and initialization methods can be checked in one code. It is recommended to use the sigmoid function at the first layer and the hyperbolic tangent function at the last. Batch normalization and flattening layers are necessary too. It is also possible to create custom activation functions for new models.

Keras enables creation of custom models, activation functions, loss function and optimizers. When the general workflow of deep learning is understood, such as forward propagation and backward propagation, how neural networks are adjusting weights and biases for input values, how they then are activated and trained in hidden layers and give outputs in final layer, how they then are changed by back propagation with loss function and optimizers, a model can be adjusted for almost every problem. If the range for porosity is known it is possible to create an activation function that would give outputs in this range and make the training faster.

For the exact problem in the thesis convolutional neural networks are working great, however, other modern architectures can be used too. 1D convolutional neural networks are great for capturing sequential data. As mentioned before, kernels slide through each value of seismic trace and with given wavelet it is capable of extracting reflectivities and then AI with porosities. This would not work with general artificial neural networks. However, another architecture that works great with sequential data is recurrent neural networks. RNNs are commonly used in applications such as Siri, voice search, and Google Translate to solve ordinal or temporal problems. These algorithms are also used for natural language processing (NLP), speech recognition, and image captioning. As they keep track of data from previous inputs, they can influence the input and output of the present. Recurrent

neural networks produce outputs based on the prior elements within a sequence, instead of assuming inputs and outputs are independent. This feature would be useful for porosity estimation. Its main drawback is its computational time being longer than conventional CNN models.

Chapter 6: Conclusions

To achieve good generalization performance, DL, being a data-driven approach, requires a large number of labeled training sets. It can be challenging for exploration geophysics, because unlabeled data are abundant, whereas labeled data are scarce. As guided in thesis unsupervised physics guided deep learning models can use the advantage of having abundant unlabeled seismic data and cover the lack of labeled data. Having little amount of labeled well-log data and unlabeled seismic data are enough to effectively estimate porosities, as shown in the thesis.

Fast and precise solutions to seismic inversion problems are important for geophysical exploration tasks, which at the moment solely rely on model-based numerical solutions which require powerful computational resources, human intervention, and planning which may lead to months of work. The time needed for training the model to estimate porosities from seismic data was around 20 seconds by running it in moderate CPU. TensorFlow has the feature to run the model in GPUs. With powerful graphic cards, the model can be trained in a few milliseconds. And with evolving packages for deep learning, it is foreseen decrease the computation time even more in future. Using CNNs to estimate porosities can reduce significant amount of time compared to traditional methods.

The reduced computational time, need for human intervention and the need for huge amount of labeled data, as well as great accuracy are huge advantages of using unsupervised deep learning models in porosity estimation and can be as effective in other seismic inversion problems. Currently very deep models and modern architectures are used for several seismic problems: autoencoders, generative adversarial networks, u-net architectures, recurrent neural networks (Zhu et al., 2021). All of them uses unsupervised and physics guided models, but with different architectures. DL is proving to be an effective tool for seismic inversion as seismic data volumes increase exponentially and DL is proving to be successful in solving inverse problems. Deep learning algorithms and architectures has great capability of approximation and this makes it great tool not just for geophysics, but also in other sciences as well.

References

A. Adler, T. Poggio. 2021. "Deep Learning for Seismic Inverse Problems: Toward the Acceleration of Geophysical Analysis Workflows." *EEE Signal Processing Magazine* 38: 89-119. doi:10.1109/MSP.2020.3037429.

A.E. Charola, Corine Wegener, Robert J. Koestler, Robert J. Koestler. 2014. "Unexpected-Earthquake 2011: Lessons to Be Learned." doi:10.5479/si.19492367.4.

Alabi, A., Enikanselu, P.A. 2019. "Integrating seismic acoustic impedance inversion and attributes for reservoir analysis over 'DJ' Field, Niger Delta." *J Petrol Explor Prod Technol* 9: 2487–2496. https://doi.org/10.1007/s13202-019-0720-z.

Andrei S. Dukhin, Philip J. Goetz. 2010. *Studies in Interface Science.* https://www.sciencedirect.com/science/article/pii/S138373031023003X.

Artur Luczak, Bruce L. McNaughton, Yoshimasa Kubo. 2022. *Neurons learn by predicting future activity.* doi:10.1038/s42256-021-00430-y.

Bacon, M. 2005. *SEISMIC SURVEYS.* Elsevier. doi:https://doi.org/10.1016/B0-12-369396-9/00110-6.

Barkaszi, Mary Jo. 2019. "Seismic survey mitigation measures and protected species observer reports: synthesis report." https://www.researchgate.net/publication/335925587_Seismic_survey_mitigation_measures_and_protected_species_observer_reports_synthesis_report.

Farfour, M., Yoon, W. and Kim, J. 2015. *Seismic attributes and acoustic impedance inversion in interpretation of complex hydrocarbon reservoirs.*

Gerard, Schuster. 2017. *Seismic Inversion.* Vol. Volume 20 of Investigations in Geophysics. SEG Books.

Jalalalhosseini, Seyed Mostafa. 2015. "The Technique of Seismic Inversion and Use of a Relation Between Inversion Results and Porosity Log for Predicting Porosity of Carbonate Reservoir in a South Iranian Oil Field." *Energy Sources, Part A: Recovery, Utilization and Environmental Effects.* doi:10.1080/15567036.2011.580326.

Kamyab S, Azimifar Z, Sabzi R, Fieguth P. 2022. *Deep learning methods for inverse problems.* doi:https://doi.org/10.7717/peerj-cs.951.

Loewenthal, D., L. Lu, R. Roberson, and J. Sherwood. 1976. *The wave equation applied to migration: Geophysical Prospecting.* doi: https://doi.org/10.1111/j.1365-2478.1976.tb00934.x.

M. Mohammadi, H. Yousefi, and M. Bagheri. 2019. "Time-series classification using convolutional neural networks: A review." *Applied Soft Computing* 80: 854-867.

Maurya, S.P., and Sarkar, P. 2016. *Comparison of Post stack Seismic Inversion Methods:A case study from Blackfoot Field.* https://www.ijser.org/onlineResearchPaperViewer.aspx?Comparison-of-Post-stack-Seismic-Inversion-Methods-A-case-study-from-Blackfoot-Field,-Canada.pdf.

Patel, Sanskruti & Patel, Atul. 2020. *Object Detection with Convolutional Neural Networks.* doi:10.1007/978-981-15-7106-0_52.

Paul C. H. Veeken, M. Da Silva. 2004. *Seismic Inversion Methods and some of their constraints.* doi:10.3997/1365-2397.2004011.

Robinson, E. A. 1983. *Seismic Velocity Analysis and the Convolutional Model.* doi:doi:10.1017/S0016756800030867.

Russell, Brian H. 1988. *Introduction to Seismic Inversion Methods: Society of Exploration Geophysicists.* doi:10.1190/1.9781560802303.

Sheriff, Robert E., and Lloyd P. Geldart. 1982. *Exploration Seismology.* Cambridge: Cambridge UP.

Tankönytár. 2014.

team, Codecademy. n.d. "Deep Learning Workflow." https://codecademy.com/article/deep-learning-workflow.

Veeken, Paul & Silva, M. 2004. *Seismic Inversion Methods and some of their constraints.* doi:10.3997/1365-2397.2004011. .

Yilmaz, Öz. 2001. *SEISMIC DATA ANALYSIS.* doi:http://dx.doi.org/10.1190/1.9781560801580.

Zhang, Rui & Sen, Mrinal & Phan, Son & Srinivasan, Sanjay. 2012. "Stochastic and deterministic seismic inversion methods for thin-bed resolution." *Journal of Geophysics and Engineering.* doi:10.1088/1742-2132/9/5/611.

Zhu, Xiao Xiang. 2021. "Deep Learning Meets SAR." https://arxiv.org/pdf/2006.10027.pdf.