NOVA IMS Information Management School

MAA

Mestrado em Métodos Analíticos Avançados
Master Program in Advanced Analytics

Transmissibility Prediction

A Deep Learning Approach

Biazi Bayer

Dissertation presented as partial requirement for obtaining the Master's degree in Data Science and Advanced Analytics

NOVA Information Management School
Instituto Superior de Estatística e Gestão de Informação
Universidade Nova de Lisboa

NOVA Information Management School
Instituto Superior de Estatística e Gestão de Informação
Universidade Nova de Lisboa

TRANSMISSIBILITY PREDICTION

A DEEP LEARNING APPROACH

by

Biazi Bayer

Dissertation presented as partial requirement for obtaining the Master's degree in Data Science and Advanced Analytics

Advisor / Co Advisor: Mauro Castelli

Co-Advisor*:* Leonardo Vanneschi

January 2021

DEDICATION

I would like to dedicate this thesis to my wife, Katherine Bayer, for all support and incentive during all these ten years of marriage. She always believed me and saw potential where other people couldn't see.

ACKNOWLEDGEMENTS

I would like to deeply thank my thesis advisor, Dr Mauro Castelli. Thank you for being available for advising me. Also, this work would not be possible without the dataset provided by him.

I would like to thank my thesis co-advisor, Dr Leonardo Vanneschi. Thank you for the guidance and ideas for improving this work.

I would like to thank my friend and master's colleagues João Henrique Leal Arienti and Breno Lehmann for helping me and reviewing my work.

ABSTRACT

Transmissibility is a very important issue in the study of fractured rocks, as it is directly related to the efficiency of drilling and extracting oil wells, water reservoirs, and even gas exploration. In this piece of work, based on data from transmissibility simulations performed in oil fields in Norway, three different machine learning approaches were applied for predicting the transmissibility of fractured rock areas. First, the fracture diagram image was applied in two different Neural Networks architectures: GoogleNet and ResNet. Second, from the fracture diagram image, it was performed a decomposition of all fracture lines (scratches) on each image into X-axis and Y-axis and it was sent to the same two Neural Network architectures on the previous approach (GoogleNet and ResNet). And finally, it was performed a discretizing continuous variable, and applied on neural network ResNet, thus performing a multi-class classification for predictions instead of regression. Overall, this study provides contributions for transmissibility prediction on oil well fields. Creating options to the traditional technique of calculating transmissibility by computer simulation which is very costly and time-consuming.

KEYWORDS

Machine Learning; Neural Network; Deep Learning; Oil; Water; Rock; Fracture; Transmissibility

INDEX

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS AND ACRONYMS

AI Artificial Intelligence

ANN Artificial Neural Network

CNN Convolutional Neural Network

CV Computer Vision

FN False Negative

FP False Positive

ILSVRC ImageNet Large Scale Visual Recognition Challenge

ML Machine Learning

NN see ANN

ReLU Rectified Linear Unity

TN True Negative

TP True Positive

1. INTRODUCTION

1.1. MOTIVATION AND OBJECTIVES

Naturally fractured reservoirs may be characterized by hundreds up to hundreds of thousands of fractures, ranging from small to medium scales, which spread all the reservoirs. An explicit representation of all the fractures in real scenarios makes it soon unfeasible performing numerical simulations[1]. It requires lots of computing hours in "reservoir simulators" which means lots of money and too much time.

Reservoir simulators can manage up to 10^5 - 10^6 simulation cells. It depends on the hardware used, and the category of simulation will be executed (black oil or dead oil, for instance). Even when simulations reach over 10^6 cells, in production version simulations, they are still not performed well.

"Geological characterizations, by contrast, typically contain on the order of 10^7–10^8 cells. These models, which are referred to as fine grid models, geostatistical models, or simply geocellular models, represent a geological variation on very fine scales vertically, though their areal resolution is still relatively coarse[2]". So it is unfeasible to use reservoir simulator software for transmissibility calculations.

Seismic analysis images provide a wide range of information about fractures of rocks in that area. To each of those images, it is necessary to calculate the **transmissibility**, which means, how easy a fluid passes through the fractured rock area and in which direction it flows. There are two types of transmissibility: **Linear** and **Radial**. Today it is very expensive and slow to calculate those values by using simulations.

In the past few decades, deeper learning has enjoyed great success in a variety of multi-purpose application domains. This new area of learning has grown rapidly and has been applied to most traditional application domains, as well as some areas that present measurement opportunities.

[1] "Advances in computation of local problems for a flow-based ...." 30 out.. 2015, https://www.mate.polimi.it/biblioteca/add/qmox/55-2015.pdf. Acessed on 26 ago.. 2020.
[2] "Upscaling and Gridding of Fine Scale Geological Models for ...." http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.485.6291&rep=rep1&type=pdf. Acessed on 26 ago.. 2020.

Some different methods have been proposed based on different categories of learning, including supervision, semi-supervision. The experimental results show high performance in the use of deeper learning, in the areas of image processing, computer, speech recognition, automatic translation, art, the image in medicine, medical information processing, robotics and control, bioinformatics, natural language processing (NLP), cybersecurity and many others.

Since 2010 one field of Artificial Intelligence, **Computer Vision** (CV), has evolved a lot thanks to ImageNet Large Scale Visual Recognition Challenge (ILSVRC)[3]. Now Artificial Neural Networks have been used on large scale for solutions related to CV. From 2012 on, new algorithms using **Convolutional Neural Networks** (CNN) have been created over the years.

The goal of this research is to use Artificial Intelligence (more precisely CNN based CV) for creating a regression model to predict the transmissibility of fractured rock matrices. It could be a faster and cheaper way to predict transmissibility on fractured rock areas.

1.2. Contributions and Importance of the Topic

The contribution of this work can be extended to the following areas:

Geology - Same techniques used to find transmissibility is applied to all kind of fractured areas and water reservoir[4];

Oil Exploration - This is the main area affected by the results of this work.

Computer Vision - Techniques used here can help further research due to the approaches adopted.

Machine Learning - Insights about all algorithms tested during this piece can help to guide further works.

[3] "ImageNet Large Scale Visual Recognition ...." http://www.image-net.org/challenges/LSVRC/. Acessed on 24 ago.. 2020.
[4] "Hydraulic Conductivity/Transmissibility - Dielman - - Major ...." 15 abr.. 2005, https://onlinelibrary.wiley.com/doi/abs/10.1002/047147844X.gw481. Acessado em 26 ago.. 2020.

1.3. METHODOLOGY

The objective of this work is to test solutions to predict transmissibility from images of rock fractures. Therefore, it is a regression.

The solutions are:

- **Decomposition and grouping of vectors**: This is a way to create new features over the input. In this part, fractures are transformed into vectors and those vectors are decomposed in the X-axis and Y-axis and then is performed a summation of those values;

- **Bitmap**: This is a regular solution used in Computer Vision solutions. The images are converted into matrices of “zeroes” and “ones” before sending it to the neural networks;

- **Discretization of targets**: Instead of having one more regression, the idea is to try a multiclass classification. In this part, the target columns are divided into bins. Each bin is a category and can be used in classification algorithms.

Before starting the tests, it was necessary to prepare the data: missing values, outliers, and resource engineering, among other processes.

The solutions to be tested are all based on neural networks. Therefore, some of the main network architectures were tested so that it was possible to choose one: a ResNet.

The optimizer algorithm used was “**adam**”. Adam is a stochastic gradient descent method that computes individual adaptive learning rates for different parameters from estimates of first- and second-order moments of the gradients.

The metric to evaluate the models was **MSE** (Mean Square Error).

As one of the solutions is a classification, an assessment was based on the "loss" of each solution with some weighting regarding the classification, since it is pointing to a set of values (bin) instead of the exact value as the previous solutions. That is why this research has a quantitative character.

It is applied research, as the conduct of tests is always thought out and oriented to later use in a real environment. Data collection was done experimentally, creating neural networks, changing these, and verifying the difference between configurations.

All the programs used in the tests were made using a Python language to access the TensorFlow/Keras stack using Jupyter Notebooks.

1.4. STRUCTURE

We first present individually the main topics of our work: Transmissibility and Neural Networks (**Introduction, Transmissibility, and The State of the Art**), on chapters **1**, **2,** and **3**, respectively, to provide the non-specialist reader sufficient knowledge to comprehend the remainder of the text.

The practical development is then divided into two main topics, by their order of appearance on the work: The **Data Preparation** (**4**), where we explain the process and specificities of creating the dataset; the **Approaches** (**5**) used as paths to achieving the better results.

Finally, the discussion of the **Results** (**6**), the **Conclusions** (**7**), and the **Future Improvements** (**8**) present the main findings and future work.

1.5. RESEARCH QUESTIONS

This work was constructed to answer the following questions:

H1. Is it possible to predict transmissibility values based only on fractured rock images?

H2. Which technique presents the best results?

2. TRANSMISSIBILITY: AN INTRODUCTION

During oil wells drilling, one of the most important steps is getting the transmissibility and then calculate how difficult it will be to flow the oil out of the reservoir. **Transmissibility** is calculated based on **Porosity** (quantity of empty spaces inside the rock), **Permeability** (how these spaces are interconnected and how easy a fluid can pass through the rock), and **Fractures** (spaces among rock fragments).

2.1. POROSITY

The consideration that the rock reservoir was formed through sediment deposition in past geological times gives rise to three different types of empty spaces or pores. Some empty spaces that developed were interconnected with other empty spaces, forming a network, and some related to other empty spaces, although some pores were completely isolated or enclosed in other empty spaces due to cementation. Thus, almost all porous media have three basic types of pores, interconnected pores, snag pores, and isolated or closed pores.

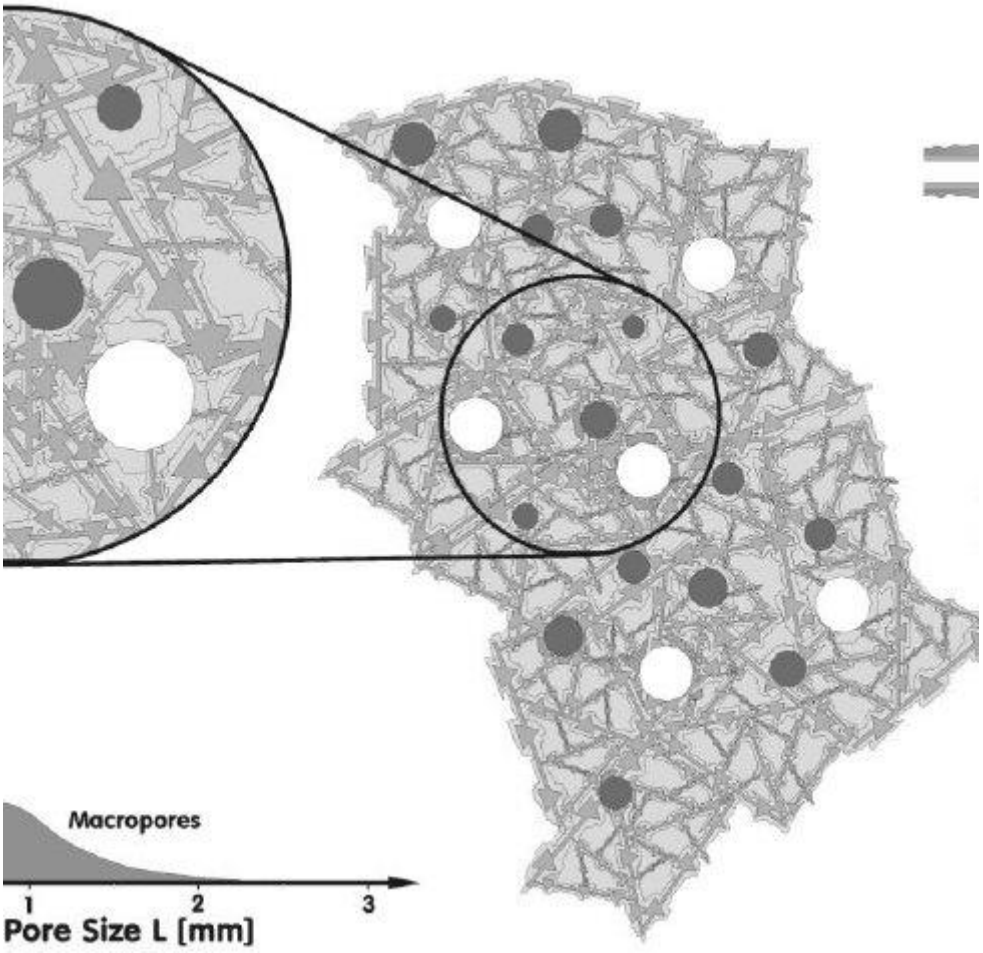


Figure 2.1-1: Porosity example.

Pedersen & Christensen (2007) believe that rock porosity can generally be classified by way of origin, as original or induced. Original or induced porosity is also called primary or secondary porosity, respectively. The original porosity resembles a native porosity, that is, developed in the deposition of the material and, the induced porosity develops by some geological process after rock deposition. One of the main common examples of induced porosity is the development of fractures commonly found in limestones.

The main parameters that can influence porosity

Many factors affect the porosity of rocks, including grain size, grain shape, classification, clay content, compaction, and cementation. To understand the effect of grain size on porosity, an example is considered a system of rounded sediments, like that described by Graton & Fraser (1935) that are packed in a cubic arrangement that results in the porosity of 47.64%.

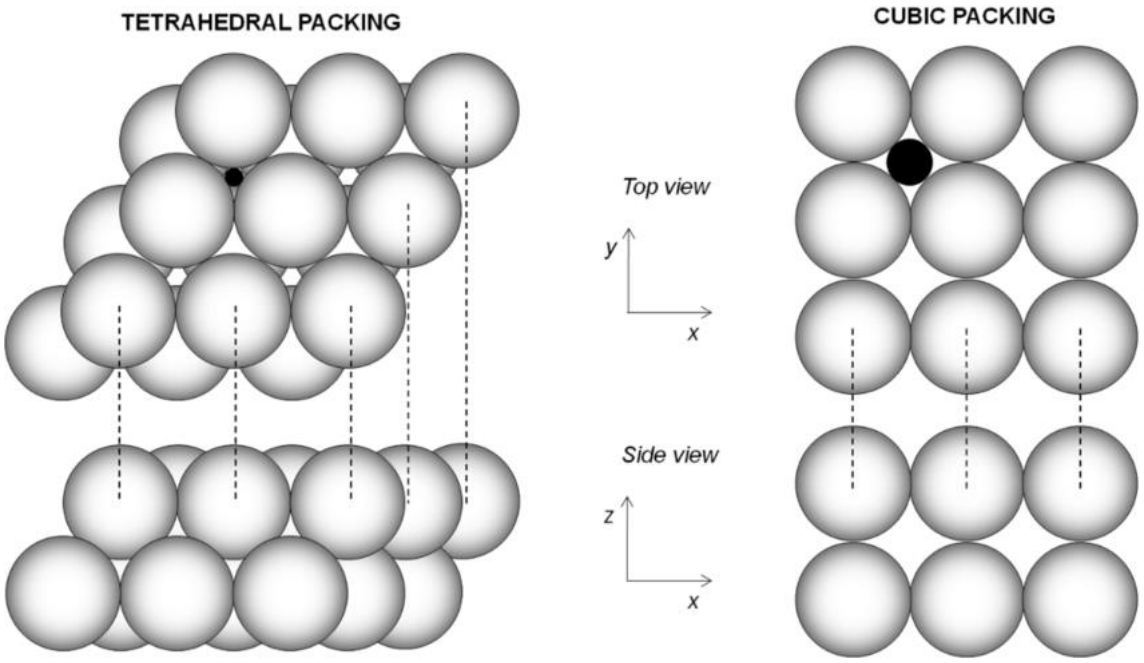


Figure 2.1-2: Tetrahedral and Cubic packings.

These theoretical porosities can be calculated using the geometry of the two different systems. This fact indicates that if a group of large diameters with uniform grains and a group of small diameters with uniform grains are packed in the same way at a fixed volume, then the two arrangements would have the same porosities. Therefore, the porosity of this theory system is independent of the grain size.

According to Zorlu et al. (2008) porosity is largely dependent on the packaging characteristics and the variation in grain size and shape. Well-classified spherical grains (a semblance of grain sizes) usually results in excellent reservoirs with greater porosity, because the grains leave large empty spaces when packed.

On the other hand, poor classification usually results in lower porosities, since smaller grains tend to occupy the empty spaces created among the largest.

Porosity is a measurement related to density, e.g., it gives us an idea about how much space there is inside the medium, for instance, a layer of rocks. It is defined as the ratio of the volume of the voids or pore space divided by the total volume.

$$n = \frac{V\ pore\ space}{V\ total}$$

Figure 2.1-3: Porosity Formula.

Porosity is highly variable through the different layers and types of rock. It depends on how dense the layers are.

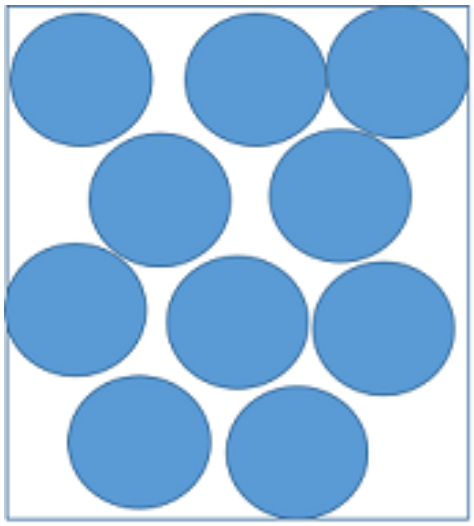


Figure 2.1-4: High porosity, large spaces.

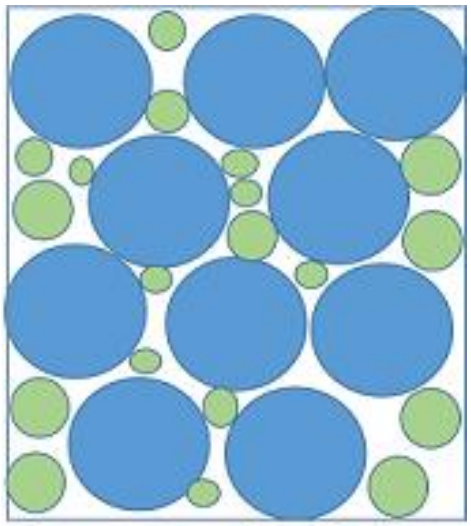


Figure 2.1-5: Low porosity, small spaces.

The pictures above can show the differences between two rock samples with the same structure (blue circles) but with porosities completely different. The second one, picture 2.1-5, show fewer empty spaces due to sediments (green circles).

2.2. PERMEABILITY

Permeability is known as how much the free/empty spaces in a porous medium are interconnected under the action of a pressure differential and its flow property.

Permeability is the most important physical property of a porous medium, while porosity is its geometric property. Permeability describes the conductivity of a porous medium concerning the fluid flow and describes the ease with which a fluid can move through the porous material. In this way, it is related to the connectivity of the empty spaces and the size of the pores.

For quality control, units of measurement may be irrelevant, but for scientific research, it is essential to be able to express results in absolute units. Thus, the permeability measures expressed in units of airflow or time cannot be compared with the dimensions of the pores. For this reason, it is necessary to calculate the permeability that results in appropriate units.

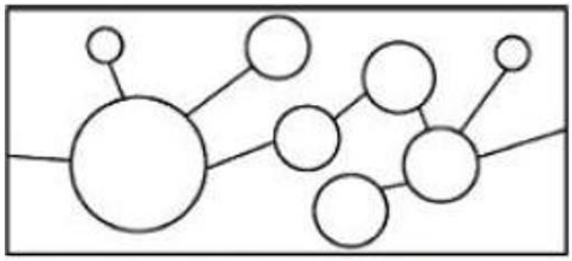


Figure 2.2-1: High porosity/low permeability.

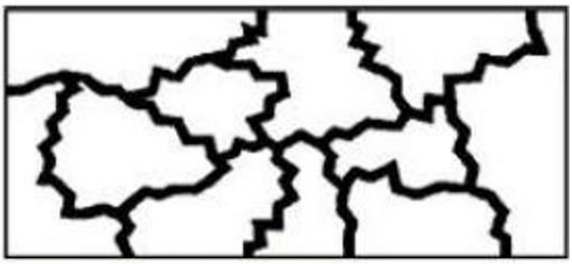


Figure 2.2-2: Porous/high permeability.

In picture 2.2-1, despite a high porosity, it presents low permeability because only a few empty spaces are interconnected. In picture 2.2-2, all pieces are interconnected, that is why it presents high permeability.

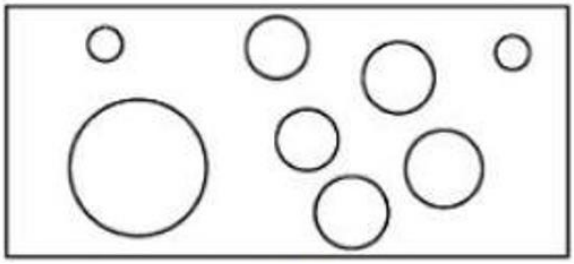


Figure 2.2-3: Porous/non-permeability

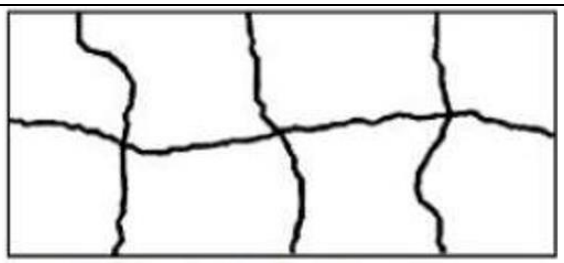


Figure 2.2-4: Low porosity/high permeability.

In picture 2.2-3, we have the same porosity as in picture 2.2-1, but the free spaces are not interconnected at all. That is why its permeability is zero. In the last picture, 2.2-4, despite low porosity it presents high permeability because all free spaces are interconnected.

2.3. FRACTURES

A rock fracture occurs when a rock is broken into two or more pieces. They affect the transmissibility because of how many fractures in the rock better will be the flux of a fluid to pass through it. It is so helpful for extracting oil that, most of the time, engineers force fracturing rock in some areas to improve transmissibility. That is why this piece of work is so important. So, depending on how these fractures are made, what are their shape, their size, and the number of rock fractures, it can provide different levels of transmissibility.

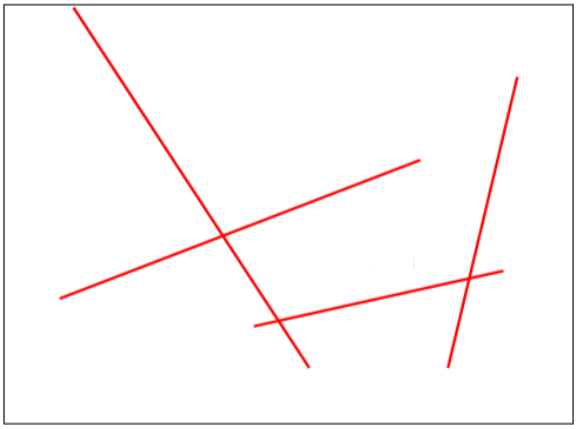


Figure 2.3-1: Fractures in a rock.

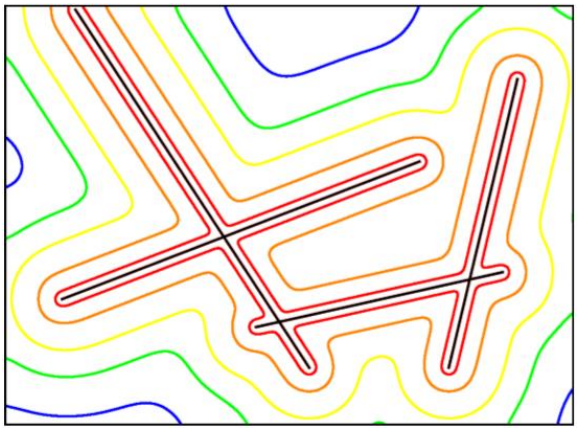


Figure 2.3-2: Isocountours of pressure.

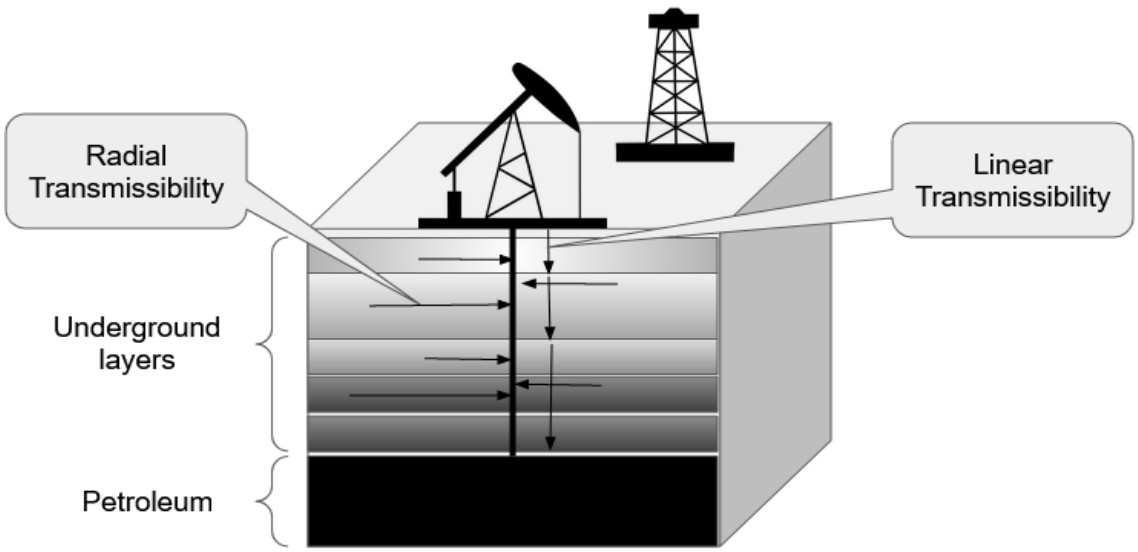
2.4. TRANSMISSIBILITY

Transmissibility is a measure that tells us how easily a fluid passes through a medium. It is calculated by the permeability times the thickness of the layer divided by the viscosity of the fluid.

$$\text{Transmissibility} = \frac{\text{Permeability} \times \text{thickness}}{\text{Fluidviscosity}}$$

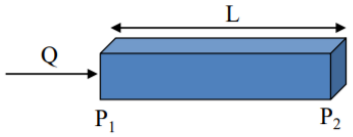
Figure 2.4-1: Transmissibility formula.

There are two types of transmissibility: **Linear** and **Radial**.

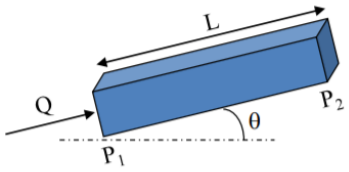


Linear transmissibility (also called Axial Transmissibility) refers to the ease of fluid flow in the one-dimension direction in wells for fluid exploration.

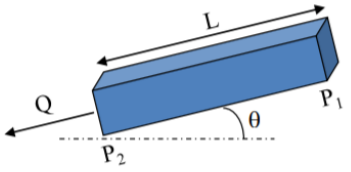
Linear Flow



$$Q = \frac{kA}{\mu}\left(\frac{P_1 - P_2}{L}\right)$$



$$Q = \frac{kA}{\mu}\left(\frac{P_1 - P_2}{L} - \rho g \sin\theta\right)$$

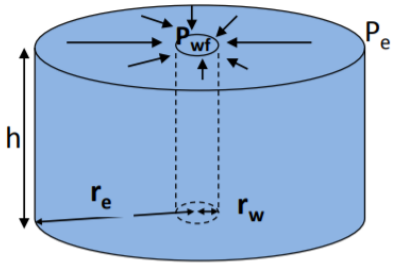


$$Q = \frac{kA}{\mu}\left(\frac{P_1 - P_2}{L} + \rho g \sin\theta\right)$$

Figure 2.4-2: Formulas for Linear Transmissibility.

Radial transmissibility refers to the ease of fluid flow in a plan section.

Radial Flow



$$Q = -\frac{kA}{\mu}\left(\frac{dP}{dR}\right)$$

$$Q = \frac{-k2\pi H}{\mu}R\left(\frac{dP}{dR}\right)$$

$$\frac{dR}{R} = \frac{-2\pi kH}{\mu Q}dP$$

$$\int_{r_e}^{r_w}\frac{dR}{R} = -\int_{p_e}^{p_{wf}}\frac{2\pi kH}{\mu Q}dP$$

$$\ln\frac{r_w}{r_e} = \frac{-2\pi kH}{\mu Q}\left(P_{wf} - P_e\right)$$

$$Q = \frac{2\pi kH\left(P_e - P_{wf}\right)}{\mu \ln\left(r_e/r_w\right)}$$

Figure 2.4-3: Formulas for Radial Transmissibility.

3. THE STATE OF THE ART

Before modelling a Neural Network (NN) from scratch it was necessary to understand the more known networks nowadays and decide which one we should use as a guide for our experiments.

3.1. ALEXNET

This algorithm started it all (Though some may say that Yann LeCun's paper in 1998 was the real pioneering publication). The paper titled "ImageNet Classification with Deep Convolutional Networks" [5] has been cited thousands of times since then and it is a paramount publication in the field. Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton created a "large, deep convolutional neural network" that was used to win the 2012 ILSVRC. 2012 marked the first year where a CNN was used to achieve a top 5 test error rate of 15.4%.

The second place achieved an error of 26.2%, so this is a great improvement. After that, Convolutional Neural Networks became almost a pattern in the competition. In that paper, the group detailed the architecture of the network (which was called AlexNet). The used layout was quite simple compared to the modern architectures of CNN. The network was made up of five Conv layers, max-pooling layers, dropout layers, and three fully connected layers.

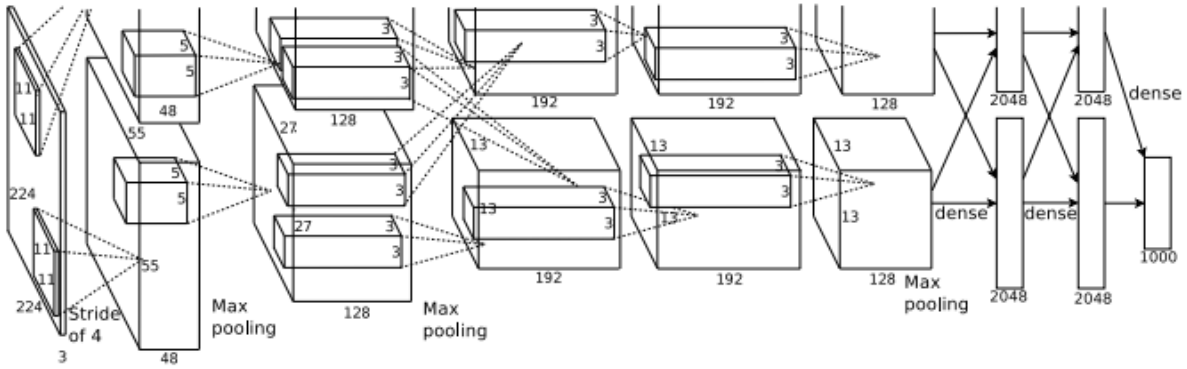


Figure 3.1-1: AlexNet architecture.

It is important to mention that AlexNet controls the complexity of the layer model completely connected by dropout and to further increase the data, the AlexNet training loop adds a large number of images such as inversion, clipping, and colour changes, making the model more robust.

The reasons why the AlexNet model is suitable for the analysis of forged images in its deep structure, its simple structure, fast training time, and less memory occupation. All these reasons lead Alex Net to be considered one of the best options in the process of detecting counterfeits. The Alex Net model can be learned automatically to identify and detect various types of image editing, and this aspect

[5] Krizhevsky, Alex & Sutskever, Ilya & Hinton, Geoffrey. (2012). ImageNet Classification with Deep Convolutional Neural Networks. Neural Information Processing Systems. 25. 10.1145/3065386.

eliminates the need for human intervention to define forensic detection capabilities. This model is used to streamline training and reduce overfitting.

In this regard, the study by Crawford et al. (2019) aimed to analyze the prediction of pore volume and transmissibility multipliers to simulate geomechanical effects in naturally fractured reservoirs. According to the authors, the implementation of stress sensitivity in the simulation of naturally fractured reservoir flow can be achieved through the use of multipliers that serve to modify the pore volume of the fracture network due to the change in reservoir pressure and the transmissibility of the fracture network due to changes in pore volume.

3.2. ZFNET

There was a large increase in the number of CNN models submitted to ILSVRC 2013. The best algorithm in that year was built by Matthew Ziegler and Rob Fergus. They named it "ZFNet" and this model got an 11.2% error rate. The architecture of ZF Net was more of a fine-tuning to the AlexNet structure, but still, they developed some interesting ideas about increasing performance. This was such a great paper that the authors spent a lot of time explaining better the intuition behind Convolutional Networks and showing how to visualize the filters and weights correctly.

In the paper which is titled “Visualizing and Understanding Convolutional Neural Networks” [6], Ziegler and Fergus begin by discussing the limited knowledge that researchers had on the inner mechanisms of these models, saying that without this insight, the “development of better models is reduced to trial and error”. While we do currently have a better understanding than 3-5 years ago, this remains an issue for a lot of researchers. The main contributions of this paper are details of a slightly modified AlexNet model and an interesting way of visualizing feature maps.

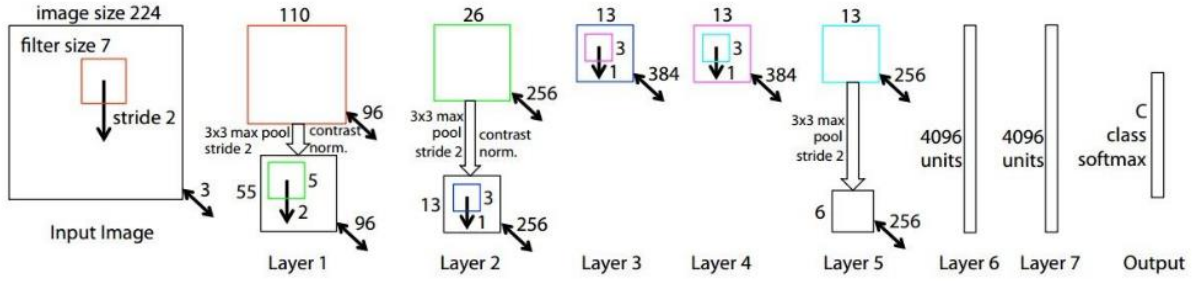


Figure 3.2-1: ZFNet architecture.

ZFNet has a similar architecture to AlexNet but differs in the size of the filter and the number of filters used. In the first convolution layer, AlexNet uses an 11 × 11 size filter with a pass of 4 and ZFNet uses 7 × 7 and uses 512,1024, and 512 filters in the third, fourth, and fifth convolution layer

[6] Zeiler M.D., Fergus R. (2014) Visualizing and Understanding Convolutional Networks. In: Fleet D., Pajdla T., Schiele B., Tuytelaars T. (eds) Computer Vision – ECCV 2014. ECCV 2014. Lecture Notes in Computer Science, vol 8689. Springer, Cham. https://doi.org/10.1007/978-3-319-10590-1_53

while AlexNet uses 384,384 and 256 filters, respectively. ZFNet was also designed to recognize objects on ImageNet (Fanany et al., 2017).

ZFNet has a similar architecture to AlexNet but differs in the size of the filter and the number of filters used. In the first convolution layer, AlexNet uses an 11 × 11 size filter with a pass of 4 and ZFNet uses 7 × 7 and uses 512,1024 and 512 filters in the third, fourth, and fifth convolution layer while AlexNet uses 384,384 and 256 filters, respectively. ZFNet was also designed to recognize objects on ImageNet (Fanany et al., 2017).

ZFNet is a trained network with a downward gradient of the backpropagation algorithm for 4 hours and 40 minutes, defining the mini lot size and the maximum season for 64 and 20 respectively. Likewise, the training accuracy rate of 99.65% and the test accuracy of 97.65% are achieved. The resources extracted from the last fully connected layer of ZFNet are sent to an ECOC classifier (Bai et al., 2014).

ZFNet like AlexNet consists of eight layers, the first five being the convolution layer, along with the three layers of maximum grouping and the next three layers fully connected. The main differences between AlexNet and ZFNet are in the first convolution layer of AlexNet, the kernel size is 11 and the step value is 4; in contrast, the kernel size is 7 and the “pass” value is 2 in the first ZFNet layer convolution.

Otherwise, from the second to the fifth AlexNet convolution, the stride is 1; however, the “pass” value of the second convolution layer is 2 in ZFNet, and the rest are like AlexNet.

3.3. GOOGLENET

When the network architectures started growing faster, Google decided to break this trend with the introduction of the Inception module. The GoogLeNet algorithm is a 22-layer CNN. It was enough for being the winner of ILSVRC 2014 with a top 5 error rate of 6.7%. This was one of the first CNN architectures that stood out from the general approach by combining Conv and pooling layers stacking them on top of each other in a sequential structure. The authors of the paper also emphasized that this new model places notable consideration on memory and power usage.

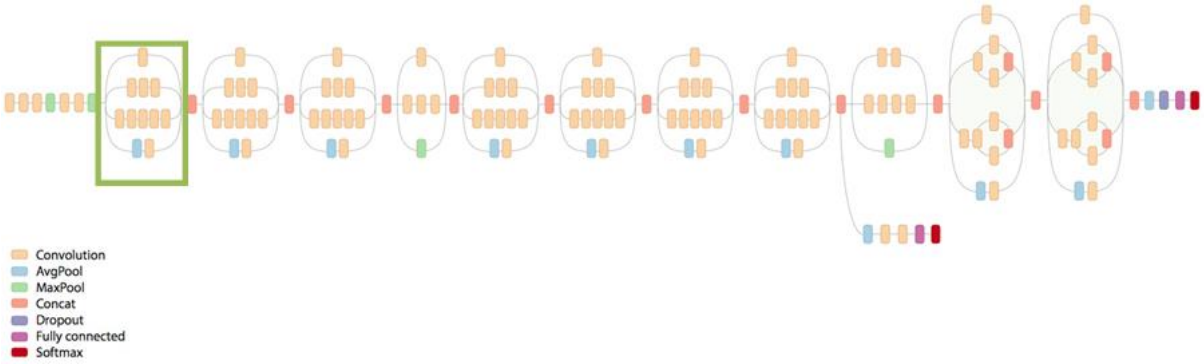


Figure 3.3-1: GoogLeNet architecture and inception module in green

This network architecture uses a CNN, and it was inspired by LeNet (1998) but implemented a new element which is called the “Inception module”. It used batch normalization, image distortions, and RMSprop. This module works based on a lot of exceedingly small convolutions to reduce the number of parameters.

Its architecture is composed of a 22-layer deep CNN but it reduced the number of parameters from 60 million in AlexNet to only 4 million.

In the picture below it is possible to understand better what happens inside of the “building block” Inception. The application of different filters provides diversity to the output layer. This is one of the improvements this algorithm brought.

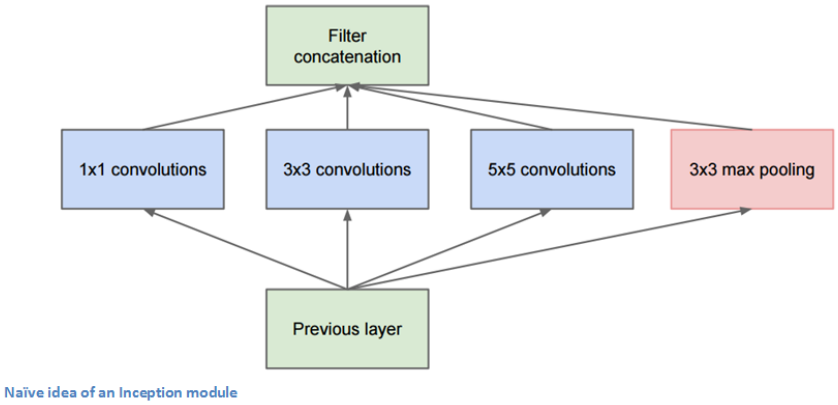


Figure 3.3-2: Inception Module.

3.4. RESNET

ResNet is a new 152-layer network architecture that sets new records in classification, detection, and localization through one incredible architecture. Aside from the new record in terms of the number of layers, ResNet won ILSVRC 2015 with an incredible error rate of 3.6% (Depending on their skill and expertise, humans generally hover around a 5-10% error rate. [7]

Residual Block: The idea behind a residual block is that the input x goes through "Conv-ReLU-Conv" series. This will give some F(x). That result is then added to the original input x.

It could be called:

$$H(x) = F(x) + x$$

Figure 3.4-1: Residual Block formula.

In traditional Convolutional Neural Networks, the H(x) would just be equal to F(x). So, instead of just computing that transformation (straight from x to F(x)), it is computing the term that was added, F(x), to the input, x. The mini-module shown below is computing a "delta" or a slight change to the original input x to get a slightly altered representation. The authors believe that "it is easier to optimize the residual mapping than to optimize the original, unreferenced mapping". Another reason why this residual block might be effective is that during the backward pass of backpropagation, the gradient will flow easily through the graph because we have additional operations, which distributes the gradient.

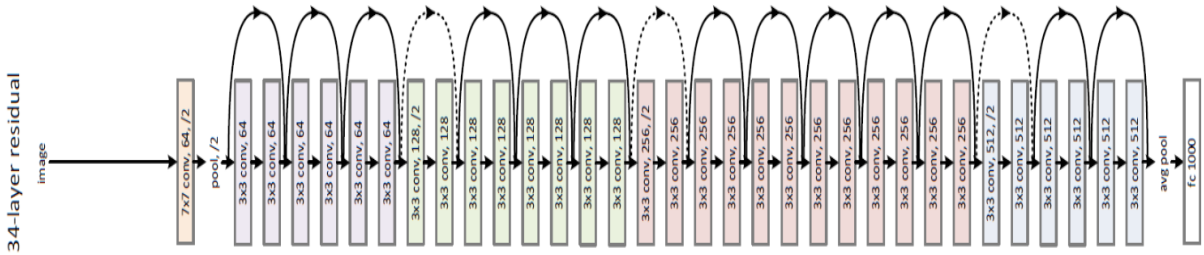


Figure 3.4-2: ResNet Architecture.

[7] https://towardsdatascience.com/review-resnet-winner-of-ilsvrc-2015-image-classification-localization-detection-e39402bfa5d8

Below is shown the performance of the best architecture of Convolutional Neural Networks in the ILSVRC competition between 2012 and 2016.

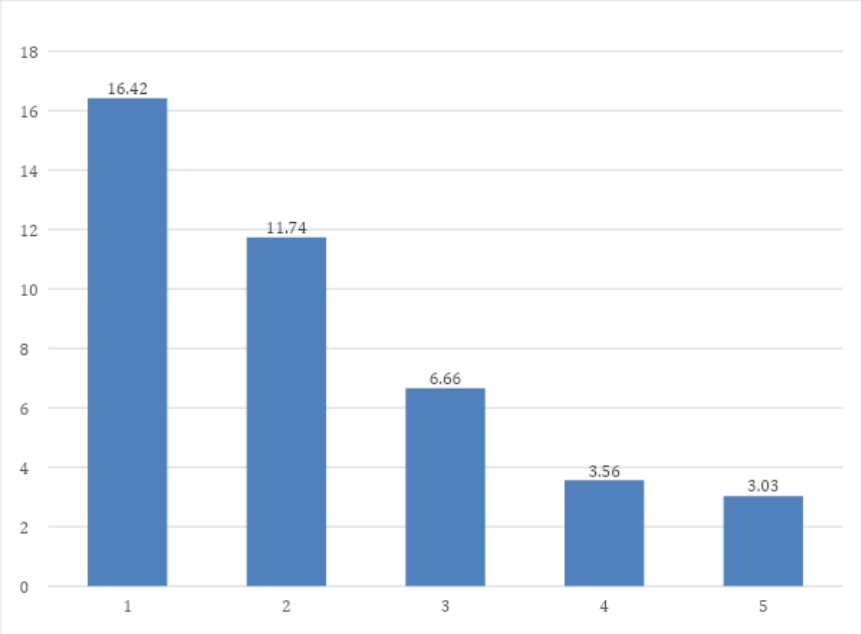


Figure 3.4-3: Performance of best algorithms error (%) by year in ILSVRC.

Number	**Neural Network**	Year
1	AlexNet	2012
2	ZFNet	2013
3	GoogLeNet	2014
4	ResNet	2015
5	ResNet (ResNeXt)	2016

Table 3.4-1: Best algorithm by year in ILSVRC.

According to the Kaggle website,[8] these are the best neural networks until the year 2016. Thus, ResNet and GoogLeNet were chosen to be applied to this research because of being well known and have easy access to their code.

There are new architectures of neural network pop out every year, like Trimps-Soushen(2016) or SENet (2017), and despite they have good improvements they were not used in this research. It could be done in future work.

[8] https://www.kaggle.com/getting-started/149448

4. DATA PREPARATION

The dataset provided was a set of **1,345** folders with files. Each folder had one fractured area image (with dimensions of 1350x1350 pixels) and one text file with two numbers related to transmissibility (Linear and radial) which was called targets x and y in this research.

4.1. DISTRIBUTION

Below are presented the charts of distribution and the quantiles for checking if the data on target X have gaussian distribution.

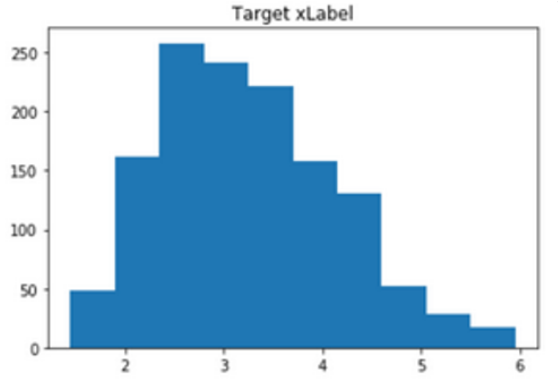


Figure 4.1-1: Target X distribution.

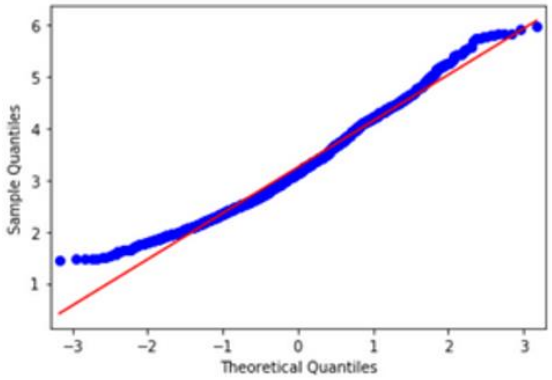


Figure 4.1-2: Target X quantiles.

And then is shown the results of the Shapiro-Wilk, D'Agostino-Pearson, and Anderson-Darling hypothesis tests.

Method	Statistics	p/cv	H0
Shapiro-Wilk	0.978	0.000	Rejected
D'Agostino-Pearson	49.762	0.000	Rejected
Anderson-Darling	7.155	0.785	Rejected

Table 4.1-1: Normality tests table for **X** target.

After executing all tests using target X, the H0 hypothesis for each method was rejected.

Below are presented the charts of distribution and the quantiles for checking if the data on target Y have gaussian distribution.

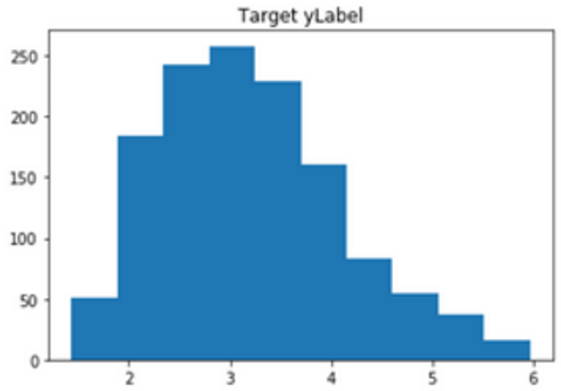


Figure 4.1-3: Target Y distribution.

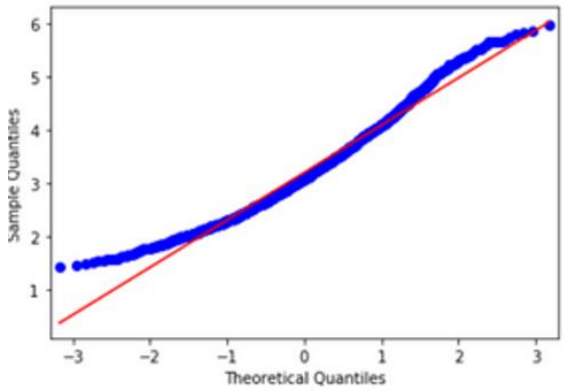


Figure 4.1-4: Target Y quantiles.

And then is shown the results of the Shapiro-Wilk, D'Agostino-Pearson, and Anderson-Darling hypothesis tests.

Method	Statistics	p/cv	H0
Shapiro-Wilk	0.972	0.000	Rejected
D'Agostino-Pearson	65.460	0.000	Rejected
Anderson-Darling	8.770	0.785	Rejected

Table 4.1-2: Normality tests table for **Y** target.

After executing all tests using target Y, the H0 hypothesis for each method was rejected.

Conclusion: Based on these tests, the samples of X and Y have **NOT Gaussian** distribution.

The target columns showed no gaussian behaviour. Despite Neural Network can work with non-gaussian distribution, it was tested to avoid operations that could only be applied on normal distribution datasets.

4.2. MISSING VALUES

Many existing, industrial and research data sets contain Missing Values. They are introduced due to various reasons, such as manual data entry procedures, equipment errors, and incorrect measurements. Hence, it is usual to find missing data in most of the information sources used. The detection of incomplete data is easy in most cases, looking for Null values in a data set. However, this is not always true, since Missing Values (MVs) can appear in the form of outliers or even wrong data (i.e. out of boundaries).

Missing values make it difficult for analysts to perform data analysis. Three types of problems are usually associated with missing values (J. Barnard, X.L. Meng - 1999):

- Loss of efficiency;
- Complications in handling and analyzing the data;
- Bias resulting from differences between missing and complete data.

During the first analysis, missing values were found: inconsistent folders, e.g., folders with no images, no text file, corrupted image, or text file. All of them, a total of nine observations, were removed and the total of observations went from 1,345 to **1,336**. Due to the nature of this project, there is no way to impute values.

4.3. OUTLIERS

A statistical outlier is any data point in a dataset that is beyond a pre-defined distribution range, usually representing a measurement error or abnormal data that should not be included. Outliers are defined in terms of being some distance away from the mean of the dataset's samples. The unit of measure for this distance is the standard deviation of the dataset, which is a measure of how similar the data samples are. Outliers can be visually determined based on a plotted graph of the data samples. [9]

While standard deviation and probability theory give a rough framework for spotting abnormalities, there is no firm mathematical definition of what constitutes an outlier.

Since the root of all deep learning training techniques in analyzing vast amounts of data to find some sort of mathematical pattern or relationship, outliers can produce all sorts of "ghosts" in a machine program if not weeded out early.

Often outliers are discarded because of their effect on the total distribution and statistical analysis of the dataset. This is certainly a good approach if the outliers are due to an error of some kind (measurement error, data corruption, etc.), however often the source of the outliers is unclear.

[9] https://deepai.org/machine-learning-glossary-and-terms/outlier visited in 27-11-2020.

There are many situations where occasional 'extreme' events cause an outlier that is outside the usual distribution of the dataset but is a valid measurement and not due to an error. In these situations, the choice of how to deal with the outliers is not necessarily clear and the choice has a significant impact on the results of any statistical analysis done on the dataset. The decision about how to deal with outliers depends on the goals and context of the research and should be detailed in any explanation about the methodology. [10]

Reference	Value
Q1	2.57046127735601230
Q3	3.88965201183290650
K	1.60000000000000000
IQR	1.31919073447689430
IQR*k	2.11070517516303100
Min. Limit	0.45975610219298124
Max. Limit	6.00035718699593800

Table 4.3-1: IQR and limits for target X.

For target "x" the IQR (interquartile range) was approximately **1.32** for this series, the constant multiplied was defined as **1.6** then the range was extended to **2.11**. Finally, all values above **6.00** and below **0.46** were considered outliers.

Reference	Value
Q1	2.55074467430058200
Q3	3.78837379384999370
K	1.80000000000000000
IQR	1.23762911954941180
IQR*k	2.22773241518894150
Min. Limit	0.32301225911164044
Max. Limit	6.01610620903893500

Table 4.3-2: IQR and limits for target Y.

For target "y" the IQR (interquartile range) was approximately **1.24** for this series, the constant multiplied was defined as **1.8** then the range was extended to **2.22**. Finally, all values above **6.01** and below **0.32** were considered outliers.

[10] https://deepai.org/machine-learning-glossary-and-terms/outlier visited in 27-11-2020.

After analyzing all those values, we identified **19** values that could be considered outliers. After this filtering, we got **1,317** observations in the final dataset. It is important to say that every time at least one of the targets was considered an outlier, the whole observation was removed.

4.4. IMAGE RESOLUTION

During the tests, a wide range of image resolution was used. The original size of the images is 1350x1350, but this size was slowing down the model training. Then it was tried 640x640, 480x480, 320x320, 256x256, and 128x128 resolutions. From all those, the best results were between 256x256 and 320x320.

When the existing Neural Networks (GoogLeNet and ResNet) were tested it was noticed that the input layer expected 224x244 images. Then it was tested and showed a greater performance to train models with no loss of precision.

Thus, all images had their resolution downscaled to 224x224 to better fit the Input Layer in neural networks. It was important because instead of changing all known neural networks' input layer, we had only one point of changing.

4.5. IMAGE COLOR CHANNELS

Colour images have height, width, and colour channel dimensions. An image can be stored as a three-dimensional array in memory. Typically, the image format has one dimension for rows (height), one for columns (width), and one for channels. If the image is black and white (grayscale), the channel dimension may not be explicitly present, e.g. there is one unsigned integer pixel value for each (row, column) coordinate in the image. Coloured images typically have three channels. A given pixel value at the (row, column) coordinate will have red, green, and blue components. Deep learning neural networks require that image data be provided as three-dimensional arrays (Brownlee, J. 2019).

Grayscale images are loaded as a two-dimensional array. Before they can be used for modelling, you may have to add an explicit channel dimension to the image. This does not add new data; instead, it changes the array data structure to have an additional third axis with one dimension to hold the grayscale pixel values (Brownlee, J. 2019).

The Keras deep learning library is agnostic to how you wish to represent images in either channel first or last format, but the preference must be specified and adhered to when using the library. Keras wraps several mathematical libraries, and each has a preferred channel ordering.

The three main libraries that Keras may wrap and their preferred channel ordering are listed below:

- TensorFlow: Channels last order.
- Theano: Channels first order.
- CNTK: Channels last order.

By default, Keras is configured to use TensorFlow and the channel order is, also by default, channels last. You can use either channel ordering with any library and the Keras library. Some libraries claim that the preferred channel ordering can result in a large difference in performance (Brownlee, J. 2019).

For this experimentation, all images were changed from RGB channels to Grayscale and, after that, from Grayscale to BW (black and white). These changes bring more contrast to images and allow a better fit for Neural Networks filters.

4.6. IMAGE INTO MATRIX

Images were in JPEG format and could be loaded directly using the Image class. This returns an Image object that contains the pixel data for the image as well as details about the image. After the image is transformed into a matrix of pixels, it is possible to access the properties of the image as well as functions that allow you to manipulate the pixels and format of the image.

The format property on the image reports the image format (it was used JPEG), the mode will report the pixel channel format (in this case, RGB), and the size will report the dimensions of the image in pixels (e.g. 1350 $\times$ 1350, or smaller later).

This step converted each image into a matrix of “0” and “1”, where zero represents a black pixel and 1 represents a white pixel. It was sent to the input layer of the Neural Networks.

4.7. DATASET SPLIT

We can split our loaded dataset into separate train and test datasets that we can use to train and evaluate models for this problem. After loading the prepared dataset, then splits it into train and test sets and we can see the shape of the prepared datasets. We can see that we have more examples in the training dataset than the test dataset.

Proportion	Train	Test
60/40	790	597
70/30	921	396
80/20	1053	264
90/10	1185	132

Table 4.7-1: Dataset split proportions used.

The best results were obtained using 70/30 proportion, e.g., 70% for train and 30% for test. Greater datasets caused overfitting and smaller presented too few observations to the Neural Network to be trained. So, the 70/30 proportion was the more balanced choice.

4.8. DATA AUGMENTATION FOR OVERSAMPLE

The performance of deep learning neural networks often improves with the amount of data available. Data augmentation is a technique to artificially create new training data from existing training data. This is done by applying domain-specific techniques to examples from the training data that create new and different training examples. Image data augmentation is perhaps the most well-known type of data augmentation and involves creating transformed versions of images in the training dataset that belong to the same class as the original image. Transforms include a range of operations from the field of image manipulation, such as shifts, flips, zooms, and much more (Brownlee, J. 2019).

The intent is to expand the training dataset with new plausible examples. This means, variations of the training set images that are likely to be seen by the model. For example, a horizontal flip of a picture of a cat may make sense, because the photo could have been taken from the left or right. A vertical flip of the photo of a cat does not make sense and would probably not be appropriate given that the model is very unlikely to see a photo of an upside-down cat. As such, it is clear that the choice of the specific data augmentation techniques used for a training dataset must be chosen carefully and within the context of the training dataset and knowledge of the problem domain. Besides, it can be useful to experiment with data augmentation methods in isolation and concert to see if they result in a measurable improvement to model performance, perhaps with a small prototype dataset, model, and training run (Brownlee, J. 2019).

Modern deep learning algorithms, such as the convolutional neural network, or CNN, can learn features that are invariant to their location in the image. Nevertheless, augmentation can further aid in this transform-invariant approach to learning and can aid the model in learning features that are also invariant to transforms such as left-to-right to top-to-bottom ordering, light levels in photographs, and more. Image data augmentation is typically only applied to the training dataset, and not to the validation or test dataset. This is different

from data preparation such as image resizing and pixel scaling; they must be performed consistently across all datasets that interact with the model.

However, It was not possible to multiply existing images by rotating or flipping because it would affect the actual targets. A rotated image would have different target values because the radial and linear flux would be changed, and there is no way to calculate these new values.

4.9. PIXEL VALUES NORMALIZATION

For most image data, the pixel values are integers with values between 0 and 255. Neural networks process inputs using small weight values, and inputs with large integer values can disrupt or slow down the learning process. As such it is good practice to normalize the pixel values so that each pixel value has a value between 0 and 1 (Brownlee, J. 2019).
It was valid for those images because they are black and white pictures and we can have pixel values in the range 0-1 then the images can be viewed normally. This could be achieved by dividing all pixel values by the largest pixel value; that is 255. This was performed across all channels, regardless of the actual range of pixel values that are present in the image.

4.10. TARGET ISOLATION

One of the main steps during the data preparation is to identify and separate independent variables from the dependent ones. In this case, it was not difficult because the dependent ones came in separate text files. Each image was in a separate folder with a text file containing two values. These two values are linear transmissibility and radial transmissibility. Then, this is a regression problem with two targets.

The objective of this step was to collect all those values from their text files and put them into two vectors: xLabel and yLabel. One for Linear Transmissibility and the other for Radial Transmissibility. The vectors were created to store these Real numbers with double precision.

The positions of those values were kept according to the position of the images previously load, then each target is related to its image by the index of their arrays.

4.11. Binning (Discretization)

Values for the variable are grouped into discrete bins and each bin is assigned a unique number (in this case, a real number) such that the ordinal relationship between the bins is preserved. The use of bins is often referred to as binning or *k*-bins, where *k* refers to the number of groups to which a numeric variable is mapped. The mapping provides a high order ranking of values that can smooth out the relationships between observations. The transformation can be applied to each numeric input variable in the training dataset and then provided as input to a machine learning model to learn a predictive modelling task (Brownlee, J. 2020).

Different methods for grouping the values into k discrete bins can be used; common techniques include:

- Uniform: Each bin has the same width in the span of possible values for the variable.
- Quantile: Each bin has the same number of values, split based on percentiles.
- Clustered: Clusters are identified, and examples are assigned to each group.

In this case, it was used the Quantile technique, where 20 bins were created for each target. It is a good hypothesis using classification instead of regression for solving this problem. Instead of continuous values, we performed a discretization to create classes of values and try a classification instead of a regression solution. So, a fork project was created to investigate and measure this possibility.

4.12. Vector Decomposition

In this stage, this technique was tried to identify each line in the pictures and transform them into vectors. Using the image below as an example, the idea is to calculate the projections of these four lines on the orthogonal axis.

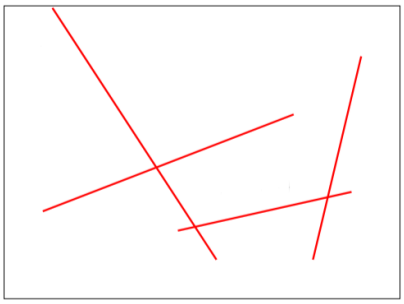


Figure 4.12-1: Example of Fractures on rocks.

After this step, the decomposition of all lines into X and Y values, it was calculated the sum of all X values and another sum of all Y values for each image. These new values were used as targets in a new experiment of Machine Learning. The process will be better explained in the next chapter.

5. APPROACHES

5.1. USING THE SUM OF DECOMPOSED VECTORS

The idea behind this approach was to decompose all lines on each image into the axis, X and Y, and sum up all of them. Doing this, images are transformed into two summed values for X (radial transmissibility) and Y (radial transmissibility).

The picture below shows how the decomposition works. It was used the OpenCV library function called “HoughLinesP” (Probabilistic Hough Line Transform), which is more precise (Knut, 2011) than the original “HoughLines” (Standard Hough Transform) because it can identify image gaps and reconstruct the entire original lines.

The outcome of the **HoughLinesP** function is a list of “lines”, e.g., each element in the list has four values: x1, y1, x2, and y2, where (x1, y1) are the **initial coordinates** and (x2, y2) are the final coordinates of each line found on the input matrix which represents an image.

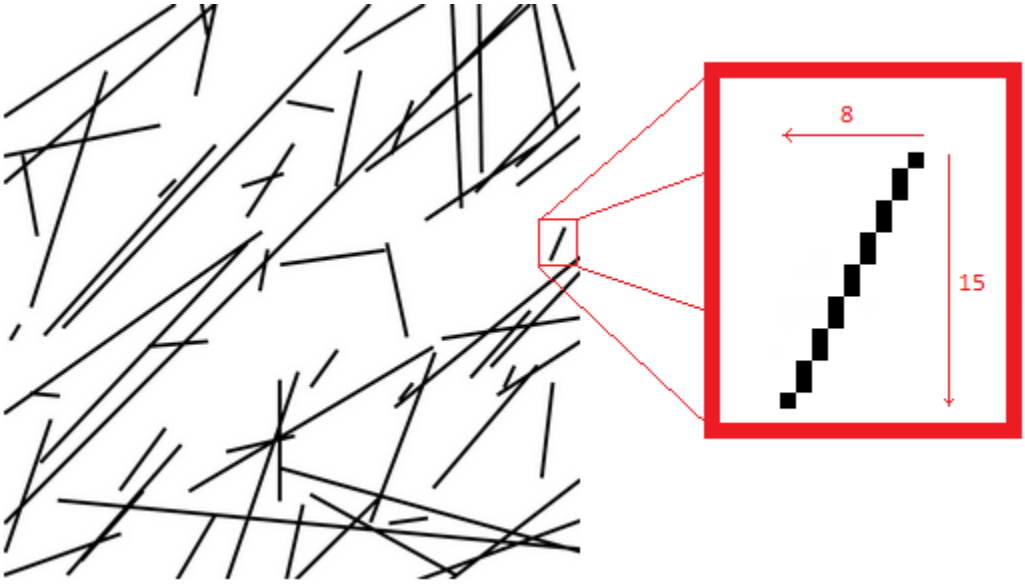


Figure 5.1-1: Vector decomposition of the fractured area picture.

With these new features, there is no need to use a very complex model specialized in extracting image features. Thus, it was created a simple model (Ellman, 2013) which had only two input values.

The model was created after some experimentation with the number of hidden layers, nodes per layer, and dropout layers.

Before getting the final shape, it was tested with many layers and nodes configurations. Here they are:

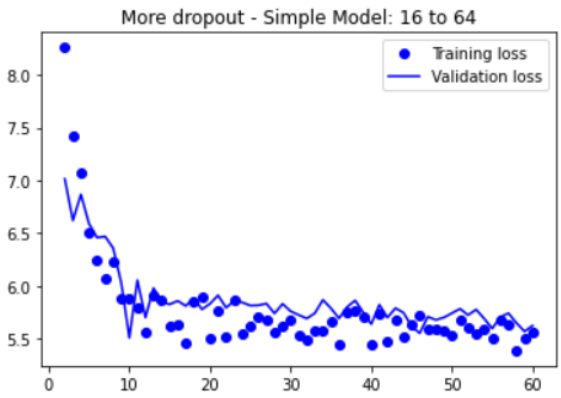


Figure 5.1-2: Three layers (16-64 nodes).

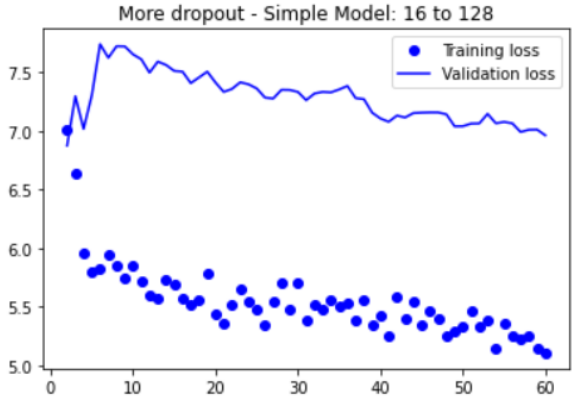


Figure 5.1-3: Four layers (16-128 nodes).

Below five layers, the neural networks (16 to 64 or 16 to 128) don't seem to be enough for this problem, they have shown no good precision (see figures above).

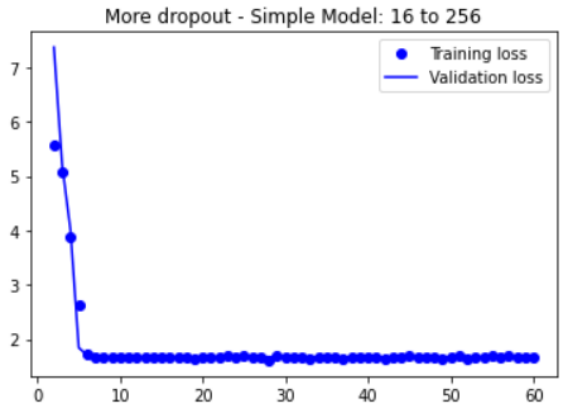


Figure 5.1-4: Five layers (16-256 nodes).

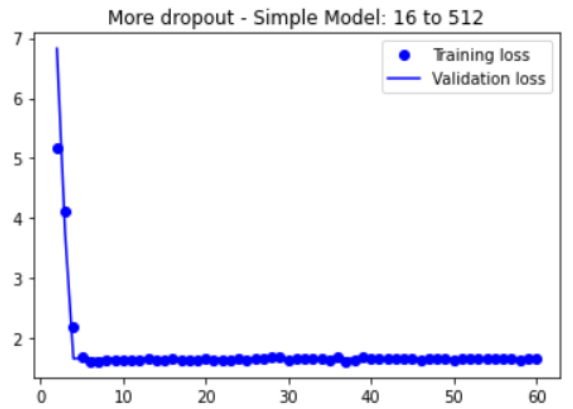


Figure 5.1-5: Six layers (16-512 nodes).

The middle-sized networks, with five and six layers, showed good efficiency. Aside from that, it is possible to see a clear convergence between validation and training losses, and it takes only a few seconds for training the models.

It was used different numbers of epochs: 10, 20, 40, and 60. In the figures above it is possible to check that the convergence occurs even before 10 epochs.

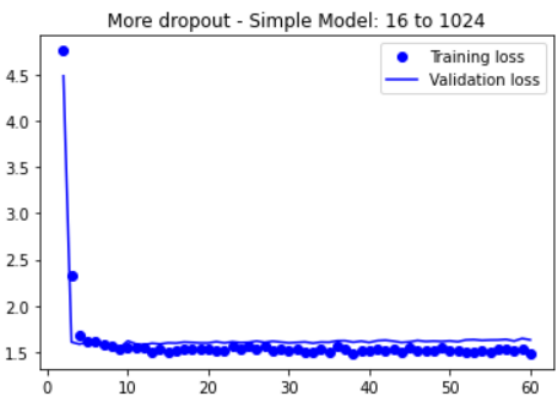


Figure 5.1-6: Seven layers (16-1024 nodes).

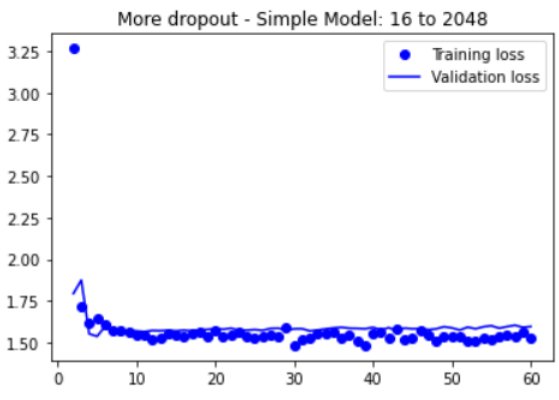


Figure 5.1-7: Eight layers (16-2048 nodes).

From seven layers on, the models started losing efficiency. They showed a divergence between training and validation losses.

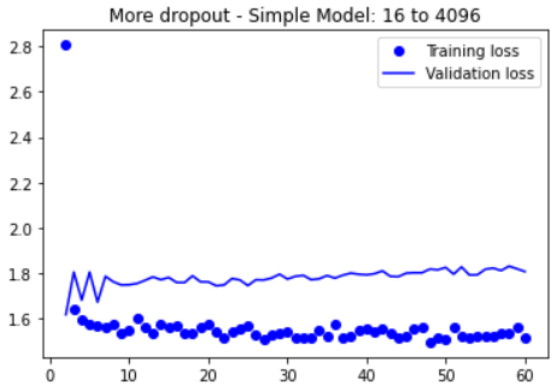


Figure 5.1-8: Nine layers (16-4096 nodes).

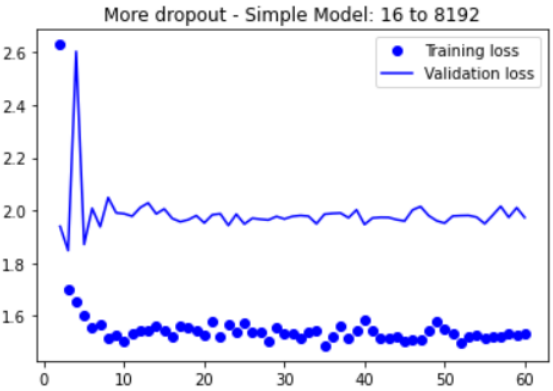


Figure 5.1-9: Ten layers (16-8192 nodes).

Networks with nine or ten layers (4096 up to 8192 nodes) seem to be too much. Despite they have better precision in comparison to the small ones, they showed a greater divergence between training and validation losses.

In the end, the very best structures were the middle-sized networks with five/six layers networks. For this research, the six-layer (16-512) was chosen because there is a clear convergence between validation and training losses, and it takes only a few seconds for training the models.

Below you can see the table with the architecture of the chosen model.

Layer (type)	Output Shape	Param #	Connected to
Input (InputLayer)	[(None, 2)]	0	
Dense_16 (Dense)	[(None, 16)]	48	Input[0][0]
Dense_32 (Dense)	[(None, 32)]	544	Dense_16[0][0]
Dropout_1 (Dropout)	[(None, 32)]	0	Dense_32[0][0]
Dense_64 (Dense)	[(None, 64)]	2112	Dropout_1[0][0]
Dropout_2 (Dropout)	[(None, 64)]	0	Dense_64[0][0]
Dense_128 (Dense)	[(None, 128)]	8320	Dropout_2[0][0]
Dropout_3 (Dropout)	[(None, 128)]	0	Dense_128[0][0]
Dense_256 (Dense)	[(None, 256)]	33024	Dropout_3[0][0]
Dropout_4 (Dropout)	[(None, 256)]	0	Dense_256[0][0]
Dense_512 (Dense)	[(None, 512)]	131584	Dropout_4[0][0]
Dropout_5 (Dropout)	[(None, 512)]	0	Dense_512[0][0]
Out_X (Dense)	[(None, 1)]	513	Dropout_5[0][0]
Out_Y (Dense)	[(None, 1)]	513	Dropout_[0][0]
Total params: 176,658 Trainable params: 176,658 Non-trainable params: 0			

Table 5.1-1: Baseline model for Decomposed Vectors solution.

The baseline model was created with:

- One input layer (for two features)
- Eleven hidden layers
- Two output layers (double targets).

The first two hidden layers are not too wide (Dense_16 and Dense_32), therefore there is only one dropout (Dropout_1) for them. From Dense_64 on, each remaining dense layer has its dropout layer for regularization and to avoid overfitting. The dense layers have 16, 32, 64, 128, 256, and 512 nodes in this order. In the end, there are two outputs since the problem requires two prediction values: Radial and Linear transmissibility.

Starting with **random weights**, before 10 epochs the **validation loss** stays stabilized around **1.70**. Thus, 40 was chosen as the maximum number of epochs.

After weights being set previously, the validation loss could only reach **1.39** with 60 epochs.

5.2. USING BINNING

Discrete values have important roles in data mining and knowledge discovery. They are about intervals of numbers that are more concise to represent and specify, easier to use, and comprehend as they are closer to a knowledge-level representation than continuous values. Many studies show induction tasks can benefit from discretization: rules with discrete values are normally shorter and more understandable and discretization can lead to improved predictive accuracy. Furthermore, many induction algorithms found in the literature require discrete features (Liu, Huan, et al., 2002).

So, **Binning** is the simplest method to discretize a continuous-valued attribute by creating a specified number of bins. The bins can be created by **equal width** and **equal frequency**.

In both methods, **arity k** is used to determine the number of bins. Each bin is associated with a distinct discrete value. In equal-width, the continuous range of a feature is evenly divided into intervals that have equal width and each interval represents a bin. In equal frequency, an equal number of continuous values are placed in each bin. The arity k was defined as **20** for this purpose.

The two methods are very simple but are sensitive for a given k. For equal frequency, for instance, many occurrences of a continuous value could cause the occurrences to be assigned into different bins. One improvement can be after continuous values are assigned into bins, boundaries of every pair of neighbouring bins are adjusted so that duplicate values should belong to one bin only (Liu, Huan, et al., 2002).

The **stopping criterion** is an important element in this process, however, as the number of bins is fixed (20) there was no need for any other stopping criterion.

There are numerous discretization methods available in the literature. This piece of work analyzed the two Split Unsupervised Binning types: Equal **Width** and Equal **Frequency**.

Another problem is outliers that take extreme values. One solution can be to remove the outliers using a threshold. And that was done executing the steps 4.1 - 4.7 plus 4.10 and 4.11, which means all data preparation, the separation of targets X and Y in different vectors (arrays), and the targets.

The goal of this path was discretizing the continuous target variables into classes and, since it was decided not to use K-NN clustering for this, the remaining options are two paths:

- Uniform bins - Keep balanced ranges (focus on width);
- Quantiles - Keep balanced classes (focus on frequency).

5.2.1. Balanced Ranges (Uniform Bins)

Uniform Binning is a method where all bins have equal size ranges. Then, during the creation of the bins, the ranges were kept at constant sizes and the differences between classes were huge, especially those at the very end and the very beginning of the distribution graph.

Thus, looking at the distribution of the target X below, the result is small classes for values under 2.0 or greater than 4.5, and big classes for values between 2.0 and 4.5. The target Y presents a similar behaviour: small classes under 2.0 or 4.0, and big classes between 2.0 and 4.0.

When the data have great differences among classes, e.g., unbalanced classes, it becomes a great issue in a multiclass classification problem because the model predicts with more precision those which have more observations.

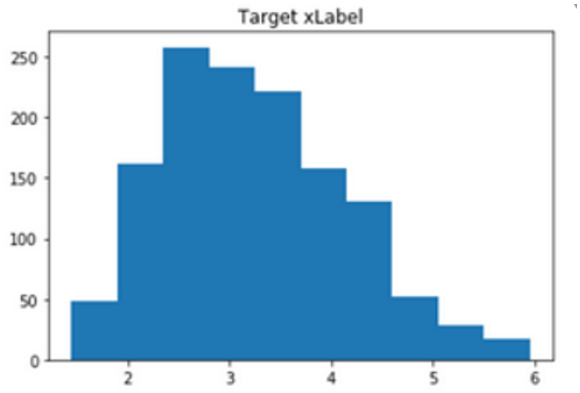


Figure 5.2-1: Target X distribution.

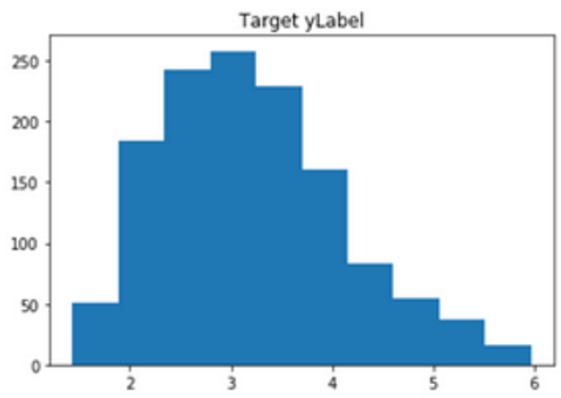


Figure 5.2-2: Target Y distribution.

Looking at the pictures above, it is noted that the classes at the chart ends have a representation much lower than the others. To avoid getting unbalanced classes it was necessary to choose between two strategies: oversampling or undersampling.

- Undersampling would affect the precision of the classification because there are not enough observations for the smaller classes.
- Oversampling is not possible because this type of dataset does not allow the creation of new observations.

Then, Uniform Binning was not an option for this study.

5.2.2. Balanced Classes (Quantiles)

Use Quantiles for binning is a method where all bins have an equal (or almost equal) number of elements. Then, during the creation of the bins, the number of elements was kept constant and the differences between classes were almost null.

Looking at the distribution of the target X below, the result is an even distribution, except for some classes on the left side. The target Y distribution chart is not so different. The classes are even distributed except for few classes on centre-left positions.

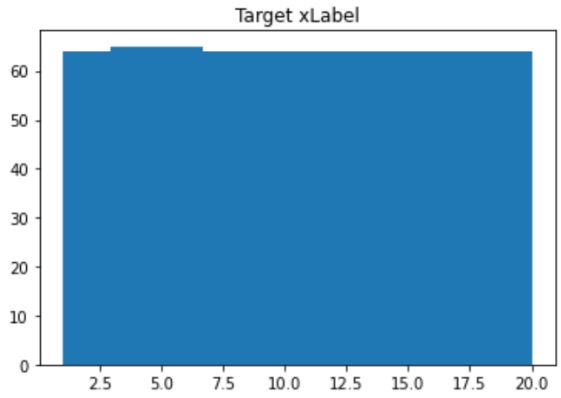


Figure 5.2-3: Target X by Classes.

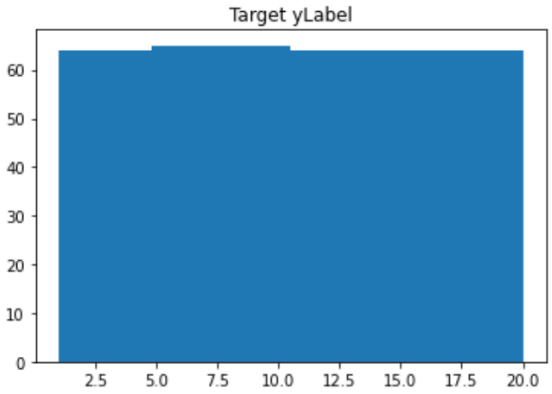


Figure 5.2-4: Target Y by Classes.

As said previously, the classification of data with imbalanced class distribution has encountered a significant drawback of the performance attainable by most standard classifier learning algorithms which assume a relatively balanced class distribution and equal misclassification costs (Sun & Wong, 2011).

To avoid this problem, the Quantiles was the discretizing method used, despite having different ranges of values through the classes and, of course, different error margins, here we have quite even distribution for all classes.

The table below shows not only the ranges but also the size of each class range.

Class #	X Ranges	Class X Size	Y Ranges	Class Y Size
1	(1.2, 2.0]	□□□□□□□□□	(1.2, 2.0]	□□□□□□□□□
2	(2.0, 2.2]	□□□	(2.0, 2.1]	□□
3	(2.2, 2.3]	□□	(2.1, 2.3]	□□□
4	(2.3, 2.5]	□□□	(2.3, 2.4]	□□
5	(2.5, 2.6]	□□	(2.4, 2.5]	□□
6	(2.6, 2.7]	□□	(2.5, 2.6]	□□
7	(2.7, 2.8]	□□	(2.6, 2.8]	□□□
8	(2.8, 2.9]	□□	(2.8, 2.9]	□□
9	(2.9, 3.0]	□□	(2.9, 3.0]	□□
10	(3.0, 3.1]	□□	(3.0, 3.1]	□□
11	(3.1, 3.3]	□□□	(3.1, 3.2]	□□
12	(3.3, 3.4]	□□	(3.2, 3.3]	□□
13	(3.4, 3.5]	□□	(3.3, 3.5]	□□□
14	(3.5, 3.7]	□□□	(3.5, 3.6]	□□
15	(3.7, 3.9]	□□□	(3.6, 3.8]	□□□
16	(3.9, 4.1]	□□□	(3.8, 3.9]	□□
17	(4.1, 4.2]	□□	(3.9, 4.1]	□□□
18	(4.2, 4.4]	□□□	(4.1, 4.4]	□□□□
19	(4.4, 4.8]	□□□□□	(4.4, 4.9]	□□□□□□
20	(4.8, 6.0]	□□□□□□□□□□□□□	(4.9, 6.0]	□□□□□□□□□□□□

Table 5.2-1: Comparison classes of targets X and Y after binning.

In the table above the columns “Size” (Class X Size and Class Y Size) are very important to understand how precise can the model be to find the correct value behind the bins. These columns show small squares that represent the size of each bin. Then the precision is inversely proportional to the number of squares, e.g., the size of each bin. Thus, in both variables (X and Y) the classes with lower precision are number 1 and 20. It is acceptable to mention also class number 19, in both variables, as a low precision class. All the others keep 2 or 3 squares but the number 18 of variable Y. The table below shows all classes according to their precision type.

Precision Type	# of Units	Classes Target X	Classes Target Y
Good	2-3	2-18	2-17
Fair	4-6	19	18, 19
Bad	10+	1, 20	1, 20

Table 5.2-2: Precision of classes for targets X and Y.

Accuracy is a statistical measure generally used for classification problems. It is calculated by summing up True Positive and True Negative and dividing by the total number of elements. Below is shown its formula:

$$Accuracy = \frac{(TP + TN)}{(TP + TN + FP + FN)}$$

Figure 5.2-5: Formula of Accuracy.

Element	Meaning
TP	True Positive
TN	True Negative
FP	False Positive
FN	False Negative

Table 5.2-3: Legend for Accuracy formula.

The goal of performing the binning was to change the data to solve this problem using a classification algorithm. For this step, it was necessary to choose a measure to evaluate the model. Then Accuracy measure was the chosen one.

Looking at the chart below, we see the accuracy of both targets, X and Y, converging after 30 epochs and reaching their best values at epoch 35. At that point, the accuracy is around 98%.

It is not possible to compare Accuracy, used here in this classification, with the loss used to evaluate the other solutions based on regression. That's why the loss was also taken into account. It will be analyzed in the next section for comparison purposes.

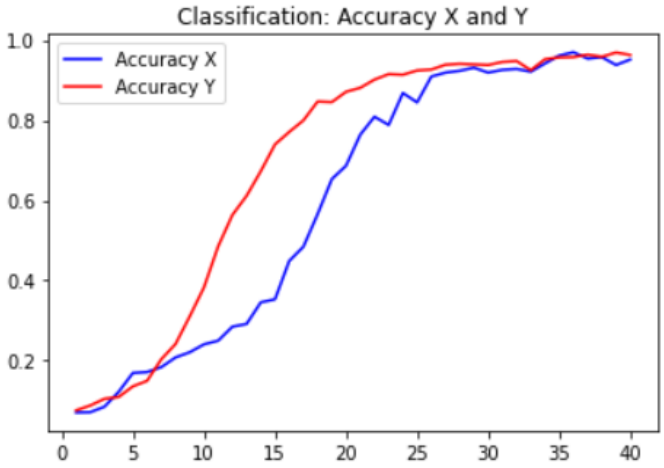


Figure 5.2.2-5: Chart of Accuracy for Targets X and Y.

5.3. USING MAP OF BITS

This path was followed by executing steps 4.1 - 4.7 plus 4.10, which means all data preparation and the separation of targets X and Y in different vectors (arrays). The idea was to use the vector of image matrices as input for a neural network algorithm to create a predictive model.

The ResNet architecture was chosen because it seems to be the best architecture for image recognition nowadays [Ilya et al., 2013]. It was slightly changed to provide two outputs (radial and linear transmissibility). Below is a chart showing the training and validation losses.

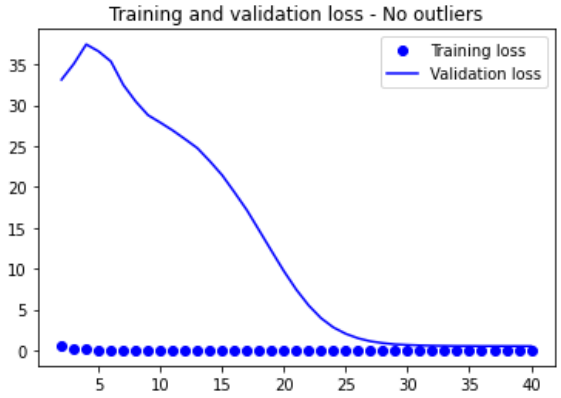


Figure 5.3-1: ResNet: batch=32 and epochs=40.

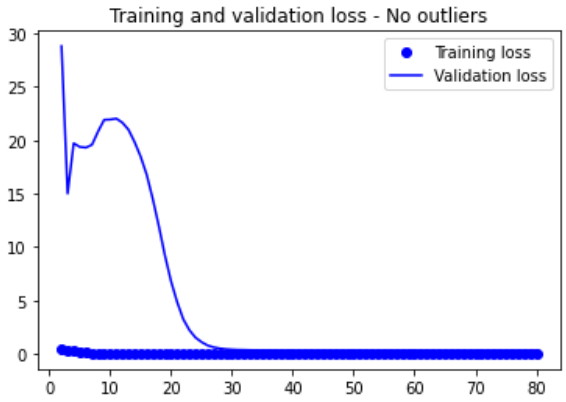


Figure 5.3-2: ResNet: batch=32 and epochs=80.

Starting with **random weights**, after 30 epochs the **validation loss** stays stabilized around **0.32**. Thus, 40 was chosen as the maximum number of epochs.

After weights being set previously, the validation loss could reach **0.08** with 80 epochs.

6. RESULTS

6.1. DECOMPOSED VECTORS

Weights	# of Epochs	Loss
Random	10	1.72
Random	40	1.70
Initialized	60	1.39

Table 6.1-1: Results for the decomposed vectors approach.

As can be seen from the Loss column from the table above, the approach Decomposed Vectors have lower values of loss overall, demonstrating that it performed worse than the others. In the best case the loss was 1.39 and it still too far from the best approach.

6.2. MAP OF BITS

Weights	# of Epochs	Loss
Random	30	0.32
Random	80	0.31
Initialized	80	0.08

Table 6.2-1: Results for the map of bits approach.

The "Map of Bits" approach shows better results than the previous one. In the best case showed 0.08 loss and it is too much near to the best value obtained.

6.3. BINNING

Weights	# of Epochs	Accuracy	Loss
Random	30	0.9377	0.1807
Random	40	0.9689	0.0961
Initialized	80	0.9807	0.0723

Table 6.3-1: Results for the binning approach.

This approach, "Binning", showed the best loss value: 0.07. It is important to mention that it is a classification problem while the previous approaches are regression problems. This means the accuracy of 98% (the loss column was added for comparison purposes) refers to a class of values, not a precise value.

7. CONCLUSIONS

The purpose of this work was to present and test some alternatives to the use of computer simulators to predict the values of transmissibility based on images of fractured areas. Throughout the development of the work, we saw some alternative algorithms being implemented and tested.

The first, and most elegant, was called "**Decomposed Vectors**".

The objective was to identify all lines of the image, then transform them into vectors where they would be facing downwards, due to the forces generated by gravity and towards the centre of the well, where the flow converges.

Then, decompose all the vectors on the orthogonal axes and then add all these projected values on each axis into a single final value, per axis. With this information, the idea was to submit these values and their targets to a neuronal network and create a model that could predict the targets: axial transmissibility and linear transmissibility.

Here there are already points to be explored such as the process of identifying the lines and the algorithms that could be used to predict the values. Using Standard Hough Lines and Probabilistic Hough Lines it was possible to identify lines, however, there are ways to increase the accuracy of this process through image treatment techniques such as changes in contrast and edge detection, among others.

This solution had poor results. At least lower than expected, as it had a **"loss"** around **1.39** in its best scenario. Perhaps partly due to the way Hough Lines identified the line set or the type of algorithm to which the data was submitted. This is something that can be explored better in the future and it looks like a very promising idea. Therefore, in the tested scenario and with the techniques used, this did not prove to be the best solution.

The second, due to the discretization of continuous values, was called "**Binning**".

This is another way of thinking about a solution to the proposed problem. By discretizing these values and creating categories, it is possible to explore an alternative completely different from the original: instead of regression, we now have a multi-class classification.

That categorization of values was conceived in two ways: one focusing on the width of the intervals and the other on the distribution of values in these intervals. As already discussed, there are clear advantages in keeping the number of elements constant in the intervals to avoid unbalanced categories, which would imply an anomalous classification by the model.

Thus, the classification achieved an excellent "**accuracy**", around **97%** on average (98% at its best) and an average **"loss"** of **0.96**. At first glance, it looks like an excellent result. However, we need to remember that it is about getting a range of values right and not the exact value. As the intervals vary from 0.1 (in the central part) to 0.8 (at the ends), we would need to give additional weight to the "loss" by multiplying by at least 2. This would bring the "loss" to values even higher than the first tested solution (1.39).

Therefore, although there are other points for improvement in this technique, we understand that this would not be the best solution either.

The third, which is a simple conversion of images into matrices, was called "**Bitmap**".

It is a direct transformation of the original image into a matrix, where each pixel of the image has been transformed into a numerical value (0 and 1) of the matrix. This matrix is then used to train a neuronal network so that the regression is made based on the "targets" columns referring to the radial and linear transmissibility.

This study was the last to be developed and, therefore, was able to count on several previous experiences and discoveries. Perhaps this was the main factor in achieving good results. At that time it was already confirmed that the ResNet architecture was the best among all those tested. Therefore, less time was spent on choosing architecture.

Unlike previous solutions, in which more advanced steps are needed, this solution is much simpler and leaves more time to experiment with other parameter settings such as learning rate and batch size, for example. Thus, with this setup, it was possible to reach a "loss" of **0.32** with random weights and **0.08** with weights pre-loaded with specific values. These are much better values than the first solutions and, therefore, the best solution was chosen.

7.1. Answers for the Research Questions

H1. Is it possible to predict transmissibility values based only on fractured rock images?

YES, after testing some techniques and algorithms it was possible to predict transmissibility with acceptable precision. All solutions: Decomposed Vector, Binning, and Map of Bits can offer good ways to predict transmissibility.

H2. Which technique presents the best results?

The Map of Bits. After testing elegant solutions like Decomposed Vectors and even a classification based on binning, the simplest solution of those showed the best results

8. FUTURE IMPROVEMENTS

8.1. GENETIC PROGRAMMING IMPROVING NEURAL NETWORKS

Evolutionary computing has traditionally been applied to the design of neural network architectures. There are two types of encoding schemes for network representation: direct and indirect encoding.

Direct coding represents the number and connectivity of neurons directly as the genotype, while indirect coding represents a generation rule for network architectures.

Although almost all traditional approaches optimize the number and connectivity of low-level neurons, modern neural network architectures for deep learning have many units and several types of units, for example, convolution, grouping, and normalization. Optimizing so many parameters in a reasonable amount of computational time can be difficult. Therefore, the use of highly functional modules as a minimum unit is promising.

The are many studies related to hyperparameters improvements using Genetic Programming with very good achievements.

The most appropriate way to choose parameter values for genetics is to consult previous studies with a description of a similar problem and adopt these values.

8.2. NUMBER OF OBSERVATIONS

The initial dataset had only 1345 observations. It is a little bit small for Computer Vision since most CV models are created using tens or hundreds of thousand images for training.

Unlike the traditional Machine Learning models where you achieve a performance plateau after a certain number of observations, Deep Learning models show improvements as the number of observations is increased.

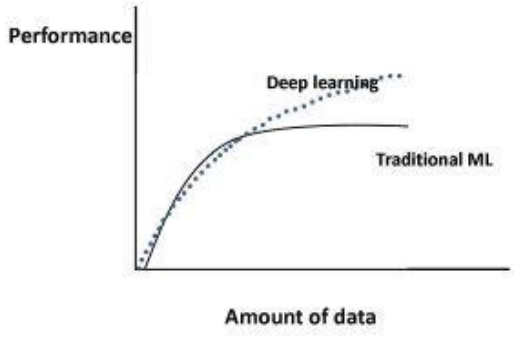


Figure 8.2-1 – Deep learning vs. Traditional ML performance comparison.

9. BIBLIOGRAPHY

Abidoye, A., B. & Chan, A., P., C. (2017). Valuers' receptiveness to the application of artificial intelligence in property valuation», *Pac. Rim Prop. Res. J.,* vol. 23, n. 2, pp. 175–193, Mai.

Bai, Jinfeng and Chen, Zhineng and Feng, Bailan and Xu, Bo. (2014) "*Image character recognition using deep convolutional neural network learned from different languages*. IEEE International Conference on Image Processing (ICIP):25602564.

J. Barnard, X.L. Meng. Applications of multiple imputation in medical studies: From AIDS to NHANES. Stat. Methods Med. Res., 8 (1) (1999) 17-36, DOI: 10.1191/096228099666230705

Bliesner, W.C., "A Study of the Porous Structure of Fibrous Sheets Using Permeability Techniques," Doctor's Dissertation, Appleton, Wis., The Institute of Paper Chemistry, June 1963. p 10.

Bostrom, n & Yudkowsky, e. (2014). The ethics of artificial intelligence, em The Cambridge Handbook of Artificial Intelligence, K. Frankish e W. M. E. Ramsey, Eds. *Cambridge University Press*, 2014, pp. 316–334.

Brownlee, Jason (2019). *Deep Learning for Computer Vision*. Edition v1.3 - Pages 83-122.

Brownlee, Jason (2020). *How to Use Discretization Transforms for Machine Learning*. From https://machinelearningmastery.com/discretization-transforms-for-machine-learning/ accessed on October 15, 2020.

Crawford, B.R., Homburg, J.M., and Fernandez-Ibanez, F, Myers, R.D., and Gao, B. (2019). *Predicting Pore Volume and Transmissibility Multipliers For Simulating Geomechanical Effects in Naturally Fractured Reservoirs*. American risk management Association p. 18-429

Ding, Y.; Urgelli, D. Upscaling of transmissibility for field-scale flow simulation in heterogeneous media. In Proceedings of the Proceedings of the SPE Symposium on Reservoir Simulation, Dallas, TX, USA, 8–11 June 1997

Ellerman, David 2013. *Quantum mechanics over sets*. arXiv:1310.8221 [quant-ph)

Fanany, Mohamad Ivan, and others. (2017) *"Handwriting recognition on form document using convolutional neural network and support vector machines (CNN-SVM).* 5th International Conference on Information and Communication Technology (ICoIC7), IEEE:16.

Graton, L., C. & Fraser, H., J. Geology 43, 785 (1935). Google ScholarCrossref; 10. G. Hägg, Zeits. f. Physik. Chemie 12, 33 (1930–31). Google Scholar; 11.

Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. (2013). *On the importance of initialization and momentum in deep learning.* In Proceedings of the 30th international conference on machine learning (ICML-13), pp. 1139–1147

Knuth, Donald E. (2011). *The Art of Computer Programming:* Vol. 4A Combinatorial Algorithms Part 1. Boston: Pearson Education.

Liu, Huan & Hussain, Farhad & Tan, Chew Lim & Dash, Manoranjan. (2002). Discretization: An Enabling Technique. Data Min. Knowl. Discov.. 6. 393-423. 10.1023/A:1016304305535.

Luque Aranda, A., 2010. Andalusian marbles: durability criteria applied in its use as ornamental stone (PhD Thesis), University of Granada (215 pp.)

Mokadam, R. G., J. Appl. Mech. 28:208 - 12 (1961).

Piteira, M., Costa, C., J. & Aparicio, M. (2017). CANOE e Fluxo: Determinantes na adoção de curso online gamificado, *Revista Ibérica de Sistemas e Tecnologias da Informação* (RISTI), pp. 34–53,

Pedersen, K. and Christensen, P., Phase Behavior of Petroleum Reservoir Fluids, Taylor & Francis Group, Boca Raton, FL, 2007.

Ridgway, C.J., Schoelkopf, J., and Gane, P.A.C., Nordic Pulp Paper Res. J. 18 (4): 377 - 381(2003)

Rodríguez-Gordillo, J., Sáez-Pérez, M.P., 2006. Effects of thermal changes on Macael marble: experimental study. Constr. Build. Mater. 20, 355–365

Yanmin Sun & Wong, Andrew & Kamel, Mohamed S. (2011). Classification of imbalanced data: a review. International Journal of Pattern Recognition and Artificial Intelligence. 23. 10.1142/S0218001409007326.

Zorlu, K., Gokceoglu, G., Ocakoglu, F., Nefeslioglu, H.A., Acikalin, S., 2008. Prediction of uniaxial compressive strength of sandstones using petrography-based models. Eng. Geol. 96, 141–158