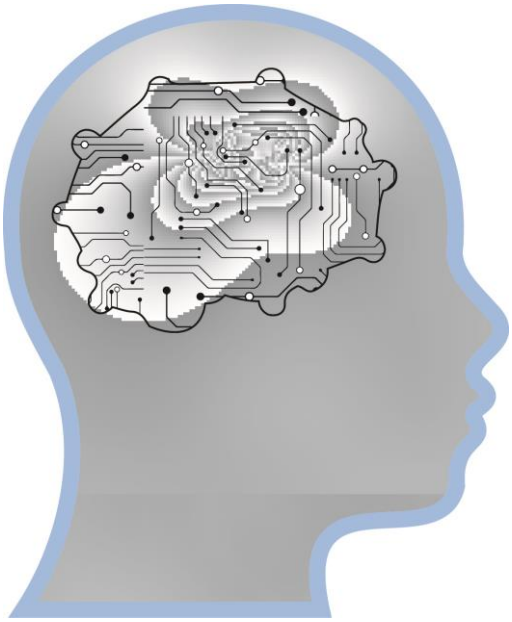


Deep learning application to detect Earth's surface deformation from earthquakes with SAR Interferometric techniques

Anahita BahreiniMotlagh

Master in Remote Sensing

Department of Geoscience, Environment and land use planning

2023

Supervisor

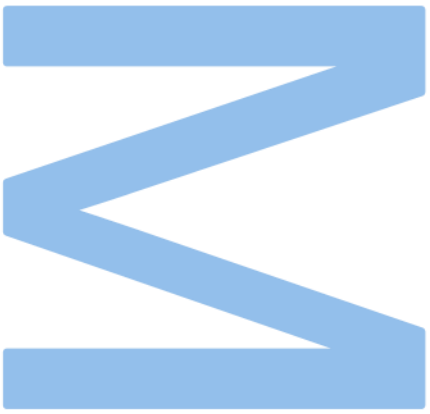
António Cunha, Auxiliar professor, University Trás-os-Montes e Alto Douro

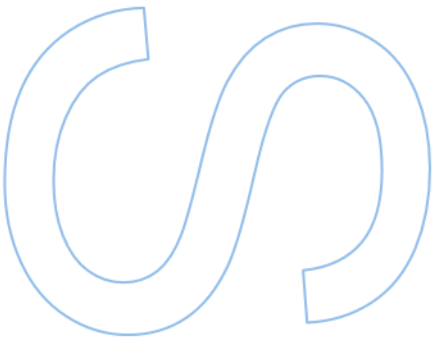
Co-supervisors

Joana Fernandes, Professor, School of Science from University of Porto

Joaquim João Sousa, professor, University Trás-os-Montes e Alto Douro


U.PORTO
FACULDADE DE CIÊNCIAS
UNIVERSIDADE DO PORTO





Acknowledgements

I would like to express my deepest gratitude and appreciation to the following individuals and institutions who have played a significant role in the completion of this thesis:

First and foremost, I am profoundly grateful to my supervisor, António Cunha, for his unwavering support, guidance, and expertise throughout this research journey. His invaluable insights, constructive feedback, and encouragement have been instrumental in shaping this work.

I am indebted to the members of my thesis committee, Joaquim Joao Moreira de Sousa and Maria Joana Afonso Pereira Fernandes, for their valuable input, suggestions, and critical evaluation of my research. Their expertise and scholarly perspectives have greatly enriched this study.

I extend my sincere appreciation to the staff and faculty at the University of Porto, Faculty of science, facilities, and the academic environment provided a conducive platform for conducting this research. The support from the Department of Geosciences, Environment and Spatial Planning has been instrumental in facilitating the progress of this thesis.

My heartfelt thanks go to my colleague, Bruno Miguel Ferreira da Silva, for his intellectual contributions, fruitful discussions, and camaraderie. His collaborative spirit and support have been invaluable in overcoming challenges and fostering a stimulating research environment.

I am grateful to the participants of this study, whose willingness to share their experiences and insights contributed immensely to the richness of this research. Their involvement and cooperation were integral to the success of this project.

Ebrahim Khevajevi Fard, my dear husband, thank you for all the support and for encouraging me whenever I was down.

Arina, my dear daughter, I would like to give you a special thank you for all the cooperation you had with me. You understood and supported me at such a young age. I hope to see your increasing success.

I would like to express my deep appreciation to my dear mother, brother and sister for their unwavering love, encouragement, and understanding throughout this demanding

endeavour. Their belief in my abilities and constant support provided the motivation and strength needed to persevere.

My dear father, you always encouraged and supported me to follow my dreams. I know that you are looking at me from heaven, I hope I was able to make you happy and proud.

Lastly, I would like to acknowledge the countless unnamed individuals who, in various ways, have contributed to the completion of this thesis. Their contributions, whether large or small, have not gone unnoticed and have played a part in shaping this work.

In conclusion, I am truly grateful to all those who have been part of this academic journey. Your support, encouragement, and contributions have been invaluable, and I am honored to have had the opportunity to work with and learn from each of you.

Resumo

O terremoto pode causar danos significativos a infraestruturas, como edifícios, pontes, barragens e oleodutos. O Radar Interferométrico de Abertura Sintética (InSAR) e as Redes Neurais Convolucionais (CNNs) são duas ferramentas poderosas no campo da Deteção Remota e têm aplicações significativas na detecção de terremotos.

Ao monitorizar o movimento terrestre com o InSAR e utilizar CNNs para análise, é possível identificar áreas vulneráveis e tomar medidas proactivas para reforçar a infra-estrutura, reduzindo danos e potenciais perdas económicas. O InSAR e as CNNs fornecem dados valiosos para avaliações de riscos sísmicos. A análise de dados históricos de terremotos e padrões de deformação do solo permite criar mapas de perigos, que são cruciais para o planeamento urbano, o mapeamento do uso do solo e o desenvolvimento de códigos de construção para reduzir os riscos relacionados aos terremotos.

O uso de InSAR e CNNs para detecção e estudo de terremotos é crucial para melhorar a capacidade de prever, preparar e responder a eventos sísmicos. Estas tecnologias não só ajudam a salvaguardar vidas e infraestruturas, mas também a promover a compreensão científica dos terramotos e dos seus impactos ambientais e sociais.

Este estudo apresenta a Deteção Remota como uma abordagem para monitorizar e estudar grandes áreas da superfície da Terra. Especificamente, destaca o uso de InSAR e Redes Neurais Convolucionais como método para detectar terremotos.

Visa aproveitar o Machine Learning, particularmente técnicas de Deep Learning, para a detecção automática de deformações sísmicas em interferogramas SAR. Foram realizadas experiências com o fim de avaliar o impacto do tamanho do conjunto de dados, da seleção da arquitetura CNN e do tratamento de conjuntos de dados desequilibrados na precisão da detecção de franjas sísmicas.

A missão do satélite Sentinel-1, equipado com um sensor Radar de Abertura Sintética (SAR), é uma valiosa fonte de dados para aplicações InSAR.

Existem duas abordagens para selecionar a arquitetura dos modelos CNN. A primeira abordagem consiste em treinar uma rede neural convolucional simples com um conjunto de dados desequilibrado, e a segunda abordagem consiste em usar uma arquitetura pré-treinada diferente para usar na aprendizagem por transferência. As arquiteturas pré-treinadas selecionadas são VGG19, ResNet50, ResNet101, ResNet152, InceptionV3, DeneNet201 e Xception.

Com base nos resultados alcançados, DenseNet201 surge como o modelo de melhor desempenho, demonstrando a mais alta exatidão, precisão equilibrada, recall, pontuação F1 e bom poder discriminatório.

Em conclusão, esta dissertação explora técnicas de Deep Learning para detecção de franjas sísmicas em interferogramas SAR e classificação de deformações sísmicas. O estudo identifica métodos de última geração para aplicar Deep Learning a dados InSAR e aborda desafios relacionados a conjuntos de dados anotados limitados e ao desequilíbrio dos conjuntos de dados. Embora exista espaço para melhorias, o estudo demonstra resultados promissores, com DenseNet201 mostrando desempenho superior. Estudos futuros deverão refinar ainda mais essas técnicas, com potencial para melhorias ainda mais significativas nos resultados. Além disso, esta dissertação responde a questões-chave de investigação, destacando o progresso substancial feito na classificação de terremotos com Deep Learning e o impacto da profundidade em arquiteturas CNN como ResNet nos resultados de classificação, enfatizando a superioridade de variantes mais profundas neste contexto específico.

Palavras-chave: Artificial Intelligence (AI), CNN Convolutional Neural Network, Deep learning, Earth’s surface deformation, Earthquake, Fringes, Interferograms, InSAR, Remote sensing, Transfer learning.

Abstract

Earthquake can cause significant damage to infrastructure such as buildings, bridges, dams, and pipelines. Interferometric Synthetic Aperture Radar (InSAR) and Convolutional Neural Networks (CNNs) are two powerful tools in the field of remote sensing and have significant applications in detecting earthquakes.

By monitoring ground motion with InSAR and using CNNs for analysis, it is possible to identify vulnerable areas and take proactive measures to reinforce infrastructure, reducing damage and potential economic losses. InSAR and CNNs provide valuable data for seismic hazard assessments. Analysis of historical earthquake data and ground deformation patterns, allows to create hazard maps, which are crucial for urban planning, land-use zoning, and building code development to reduce earthquake-related risks.

The use of InSAR and CNNs for earthquake detection and study is crucial for improving our ability to predict, prepare for, and respond to seismic events. These technologies not only help safeguard lives and infrastructure but also advance our scientific understanding of earthquakes and their environmental and societal impacts.

This study introduces remote sensing as an approach for monitoring and studying large areas of the Earth's surface. Specifically, it highlights the use of InSAR and Convolutional Neural Networks as a method to detect earthquakes.

It aims to leverage Machine Learning, particularly Deep Learning techniques, for the automatic detection of seismic deformation in SAR interferograms. We conducted experiments to evaluate the impact of dataset size, CNN architecture selection, and handling imbalanced datasets on the accuracy of seismic fringe detection.

The Sentinel-1 satellite mission, equipped with a Synthetic Aperture Radar (SAR) sensor, is a valuable source of data for InSAR applications.

We have two approaches for selecting CNN models architecture. First approach is training a simple Convolutional Neural Network with imbalanced dataset, and the second approach is using different pretrained architecture to use in transfer learning. The selected pretrained architectures are VGG19, ResNet50, ResNet101, ResNet152, InceptionV3, DeneNet201 and Xception.

Based on achieved results, DenseNet201 emerges as the best-performing model, demonstrating the highest accuracy, balanced precision, recall, F1-score, and good discriminatory power.

In conclusion, this dissertation explores Deep Learning techniques for earthquake fringe detection in SAR interferograms and seismic deformation classification. The research identifies state-of-the-art methods for applying Deep Learning to InSAR data and addresses challenges related to limited annotated datasets and dataset imbalance. While room for improvement exists, the study demonstrates promising outcomes, with DenseNet201 showing superior performance. Future research should further refine these techniques, with potential for even more significant improvements in results. Additionally, the dissertation answers key research questions, highlighting the substantial progress made in earthquake classification with Deep Learning and the impact of depth in CNN architectures like ResNet on classification results, emphasizing the superiority of deeper variants in this specific context.

Keywords: Artificial Intelligence (AI), CNN Convolutional Neural Network, Deep learning, Earth’s surface deformation, Earthquake, Fringes, Interferograms, InSAR, Remote sensing, Transfer learning.

Table of Contents

List of Tables

List of Figures

List of Abbreviations

AI	Artificial Intelligence
ANN	Artificial Neural Network
API	Application Programming Interface
AUC	Area under the ROC Curve
BAIR	Berkeley AI Research
CAMs	class activation maps
CNN	convolutional neural network
CNTK	Cognitive Toolkit
CT	Science and Technology
DL	Deep Learning
e.g.	Example
ESA	European Space Agency
EW	Extra Wide Swath
GPS	Global Positioning System
IGARSS	International Geoscience and Remote Sensing Symposium
ILSVRC	ImageNet Large Scale Visual Recognition Challenge
InSAR	Interferometric Synthetic Aperture Radar
IW	Interferometric Wide Swath
ML	Machine Learning
MLP	multi-layer perceptron
MWR	Microwave Radiation
NIR	Near Infrared Radiation
ONNX	Open Neural Network Exchange
RADAR	Radio Detection And Ranging
R-CNN	Regions with CNN Features

ReLU	Rectified Linear Unit
ResNet	Residual Network
RGB	Red Green Blue
RNNs	Recurrent Neural Networks
SAR	Synthetic Aperture Radar
SIG	Geographic Information System
VGG	Visual Geometry Group
VGG-19	Visual Geometry Group 19 Layer CNN
WV	Wave

1. Introduction

1.1. Context and Motivation

An earthquake is a natural disaster. It is the shaking of the surface of the Earth due to the sudden release of energy in the Earth's crust. Earthquakes occur along fault lines, cracks in Earth's crust where tectonic plates meet. They occur where tectonic plates meet, subducting, spreading, slipping, or colliding. As the plates grind together, they get stuck, and pressure builds up. Finally, the pressure between the plates is so great that they break loose. Depending on how much pressure has built up, the ground may tremble slightly or shake forcefully. Earthquakes near coastal areas or at sea may, in turn, cause tsunamis (massive tidal waves). Such hazards are severe threats to human life and property. If they occur in or near areas where people live, they can make buildings collapse, bridges sway, and roads buckle. For example, an earthquake occurred on 6 February 2023, In Turkey. There were 50,783 deaths, 297 missing and 107,204 injured across 11 of the 17 affected provinces of Turkey. At least 15.73 million people and 4 million buildings were affected. About 345,000 apartments were destroyed. So, the study and understanding of these natural disasters are very important and critical.

The National Earthquake Information Centre now locates about 20,000 earthquakes around the globe each year, or approximately 55 per day. But not all earthquakes are powerful enough to cause damage. In fact, earthquakes are happening all the time, on land and in the ocean. Most are so small that people don't even feel them.

To monitor the deformation of the earth’s surface, geotechnical instrumentation, GPS-based systems, and many other geodetic techniques are currently available. However, most of them are point-based measurement techniques and are too costly if a very large area needs to be monitored.

Remote sensing is another approach to studying the earth’s surface in large areas making it possible to measure dense points in a study area accurately, economically, conveniently, and efficiently.

Interferometric Synthetic Aperture Radar (InSAR) is an established method of Remote sensing to detect and measure earthquake surface displacements. InSAR uses radar images to measure ground deformation over time. It works by comparing radar images of the same area taken at different times and measuring changes in the phase of the radar waves as they bounce off the ground.

Sentinel-1 is a European Space Agency (ESA) satellite mission that is equipped with a Synthetic Aperture Radar (SAR) sensor. The mission consists of two identical satellites

that orbit the Earth in a polar orbit, providing global coverage with a revisit time of six days. Sentinel-1 is designed to provide data for a range of applications, including environmental monitoring, disaster management, and security.

InSAR interferograms can be derived from the Sentinel-1 satellite mission by using the SAR sensor on board. Sentinel-1, SAR-image is particularly well-suited for InSAR applications because it provides high-quality radar images with a consistent acquisition geometry. This means that multiple images of the same area can be easily combined to create InSAR interferograms. InSAR interferograms derived from Sentinel-1 can be used for a variety of applications, including monitoring ground deformation caused by natural hazards such as earthquakes and volcanic eruptions, as well as by human activities such as mining and groundwater extraction. The data can also be used to monitor changes in land use and to detect infrastructure deformation, such as the subsidence of buildings and bridges.

One of the main advantages of InSAR for earthquake detection is its ability to provide continuous measurements over a large area. This makes it particularly useful for detecting slow-moving earthquakes, which may not be detected by traditional methods such as GPS and seismometers. Under ideal conditions, radar images acquired by satellites can provide more information about a deforming phenomenon than even the most intensive ground-based geodetic surveys (Sousa, 2021). In addition, InSAR can detect subtle changes in the Earth's surface that may not be visible to the naked eye.

Another advantage of InSAR is its ability to provide high-resolution maps of ground deformation. This can be particularly useful for identifying areas at risk of landslides and other types of ground failure following an earthquake. InSAR can be used to monitor the ongoing recovery of an area following an earthquake, providing valuable information for disaster response and recovery efforts. Monitoring the movement of the surface of the Earth on a large scale could give scientists clues to an imminent quake. Ongoing monitoring of global fault systems also creates massive data streams that are too much to interpret manually.

We created a deep learning model addressing both limitations. Deep learning is a subfield of machine learning that involves the creation of artificial neural networks with multiple layers. These networks can learn to recognize patterns and make predictions based on large amounts of data. Deep learning has been used in a variety of applications, including image and speech recognition, natural language processing, and autonomous vehicles. It has also shown promise in fields such as healthcare and finance.

Convolutional neural networks (CNNs) are a type of deep learning network that are particularly well-suited for image recognition tasks. They use a process called convolution to extract features from images, which are then used to make predictions. CNNs have been used in a wide range of applications, from identifying objects in photos to diagnosing medical conditions from X-rays and CT scans. They are also being used to develop self-driving cars and other autonomous systems.

Deep learning and Convolutional Neural Networks (CNNs) can be used on InSAR interferograms to improve their quality and facilitate interpretation.

InSAR interferograms are formed by combining two or more radar images of the same area taken at different times to measure ground deformation. The interferogram shows the phase differences between the images, which can be used to detect and measure ground displacement. However, the interferogram can also contain a significant amount of noise and other artifacts that can make interpretation difficult.

Deep learning and CNNs can be used to reduce noise and other artifacts in InSAR interferograms, making it easier to interpret the deformation patterns. For example, a CNN can be trained to identify and remove coherent noise sources, such as atmospheric turbulence or topographic effects, from the interferogram. This can improve the accuracy of the deformation measurements and make it easier to detect smaller-scale deformation patterns.

In addition, deep learning and CNNs can be used for the automatic interpretation and classification of different patterns of deformation on InSAR interferograms. By training a CNN on a dataset of labelled interferograms, the network can learn to recognize different types of deformation patterns, such as uplift, subsidence, and shear deformation. This can be particularly useful for monitoring natural hazards such as earthquakes, landslides, and volcanic eruptions, as well as for detecting human-induced activities such as mining and groundwater extraction.

Overall, deep learning and CNNs can be useful for InSAR interferograms by improving their quality and facilitating their interpretation. By reducing noise and other artifacts, these techniques can improve the accuracy of deformation measurements and make it easier to detect and interpret deformation patterns. Additionally, automatic interpretation and classification can enable more efficient and effective monitoring of natural hazards and human-induced activities.

1.2. Objectives

This work's main objective aims to explore Machine Learning techniques for the automatic detection of seismic deformation in SAR interferograms. To achieve this objective, we started by creating and annotating a dataset for training Deep Learning models. Then, we explored Deep learning classifiers to detect seismic fringes in interferograms patches.

During this work, we looked to answer the following questions:

- Although there are few published papers about detecting earthquake' fringes by Deep Learning, how efficiently is it possible to get better results with the increasing dataset?
- How does the depth or the number of layers in variants of a specific CNN architecture (e.g. ResNet) affect the result of earthquake classification
- How to deal with imbalanced Datasets to use in classification methods?

1.3. Structure of the Document

Five chapters are presented in this document. After this introductory chapter, where the motivation, main objectives and structure of the document are presented, chapter two a, Background, focuses on the concepts that support the work is presented. The chapter starts with some definitions of earthquake, Remote Sensing and InSAR techniques and ends with an explanation of artificial intelligence and subsets, Some CNN architectures and some techniques used in this work. Material and methods, which describe how the work is done step-by-step and are discussed in chapter three. The main Results, their discussion and interpretation, are presented in Chapter four. Finally, Chapter five, presents the main Conclusions by answering the research question stated in Chapter one.

2. Literature review

This work is based on 3 concepts. The first one is Earthquake which is the phenomenon that we are studying about detecting it. The second one is Remote Sensing and InSAR techniques that we create our dataset upon this method, And the last one is Deep Learning and CNN algorithms to use for detecting earthquake fringes from interferograms. In the following, we are going to introduce some fundamentals about each one and then go deeper into Machin Learning algorithms.

2.1. Earthquake

The deformation of the earth's surface is one of the prominent phenomena associated with many geological hazards, such as landslides, land subsidence, volcano eruption and earthquakes.

An earthquake is a natural disaster. It is the shaking of the surface of the Earth due to the sudden release of energy in the Earth's [Figure 1]. As a result, seismic waves (also known as S waves) are created [Figure 2]. Seismic waves radiate from the focus of an earthquake. These waves make a deformation on the earth's surface. The vibrations from an earthquake can lead to ground displacement and surface rupture. The surface rupture can cause other hazards, as well as damage to roads and buildings.

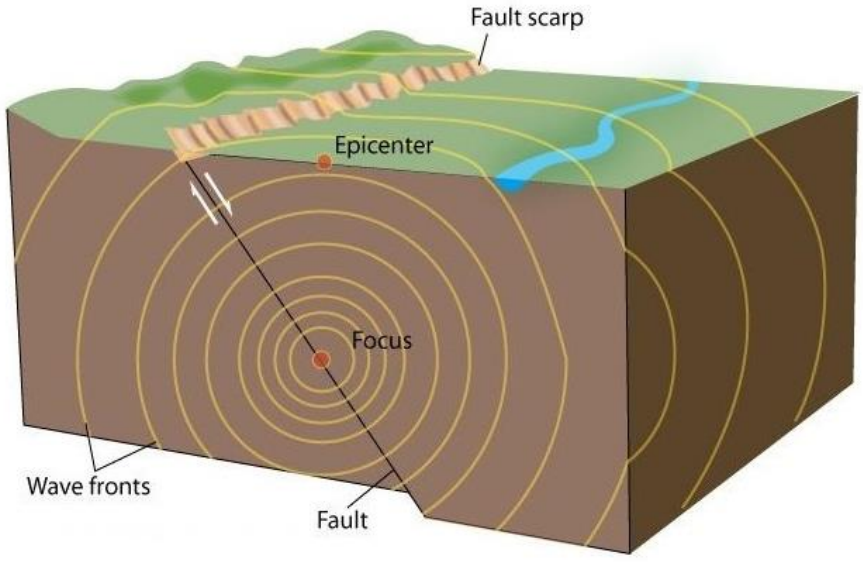


Figure 1 - Earthquake, Image from The University of Waikato Te Whare Wananga o Waikato

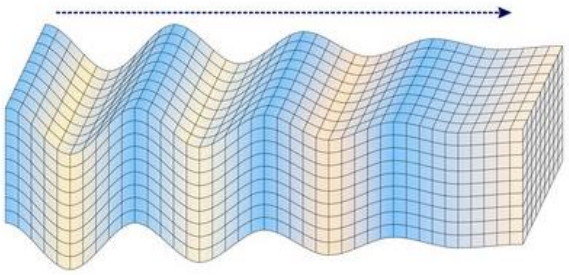


Figure 2 - Seismic Waves

2.2. Remote Sensing and InSAR techniques

Remote sensing is the process of detecting and monitoring the physical characteristics of an area by measuring its reflected and emitted radiation at a distance (typically from satellite or aircraft). Special cameras collect remotely sensed images, which help researchers "sense" things about the Earth. So, the components of remote sensing are Source of energy, Atmosphere, Earth, sensor and receiver, Ground station and Data Processing.

We can categorize remote sensing into Active remote sensing and Passive remote sensing. Active remote sensing instruments operate with their own source of energy (emission or light), while passive ones rely on the reflected one (Reflected Radiation from sun) [**Error! Reference source not found.**].

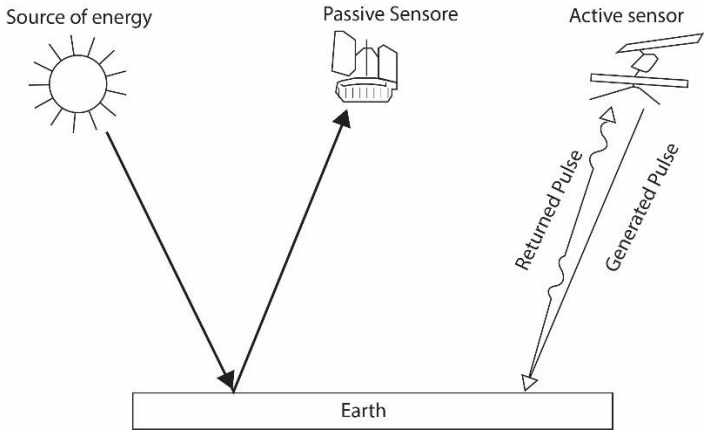
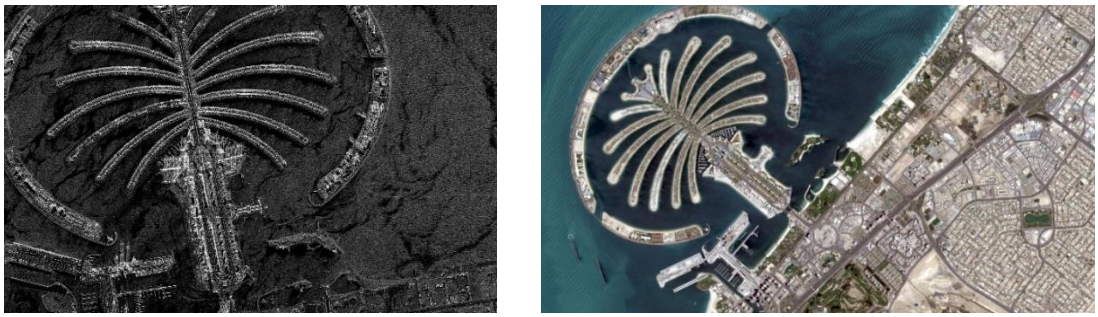


Figure 3 - Active and Passive sensors

Radars and lidars are the most epic examples of active remote sensing.

In this Figure you can see two satellite images from active sensor (SAR) and Passive sensor [**Error! Reference source not found.**].



(A) Active Sensor Image (microwave) (B) Passive Sensor Image(Optical)

Figure 4 - Satellite Images

Radiation also differs by wavelengths that fall into very short (Gamma, X-ray, Ultraviolet) short (visible, NIR, MIR) and long (microwave, Radio) [Figure 5].

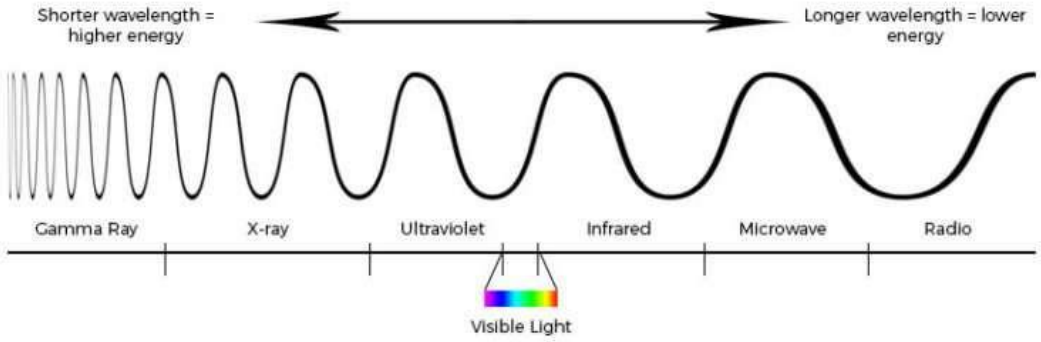


Figure 5 - Sun Radiation, Electromagnetic Spectrum, Image from Enigma, Posted on February 12, 2019

The visible spectrum is the portion of the electromagnetic spectrum that is visible to the human eye. Electromagnetic radiation in this range of wavelengths is called *visible light* or simply light. A typical human eye will respond to wavelengths from about 380 to about 750 nanometres (*Wikipedia*, n.d.) [Figure 6].

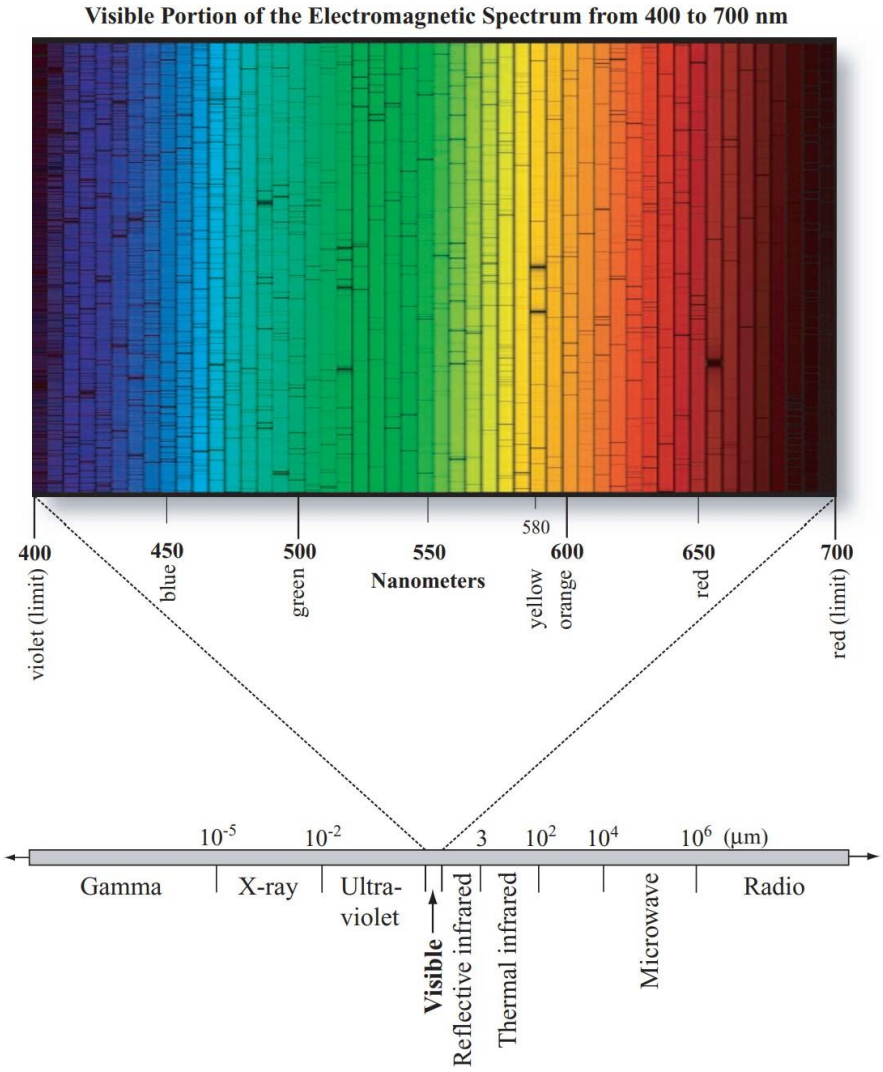


Figure 6 - visible electromagnetic spectrum, (Jensen, 2014)

In synthetic aperture radar (SAR) imaging, microwave pulses are transmitted by an antenna towards the earth's surface. The microwave energy scattered back to the spacecraft is measured. The SAR makes use of the radar principle to form an image by utilizing the time delay of the backscattered signals. When microwaves strike a surface, the proportion of energy scattered back to the sensor [Figure 7] depends on many factors:

- Physical factors such as the dielectric constant of the surface materials which also depend strongly on the moisture content.
- Geometric factors such as surface roughness, slopes, orientation of the objects relative to the radar beam direction.

- The types of landcover (soil, vegetation, or man-made objects).
- Microwave frequency, polarization, and incident angle [Figure 8].

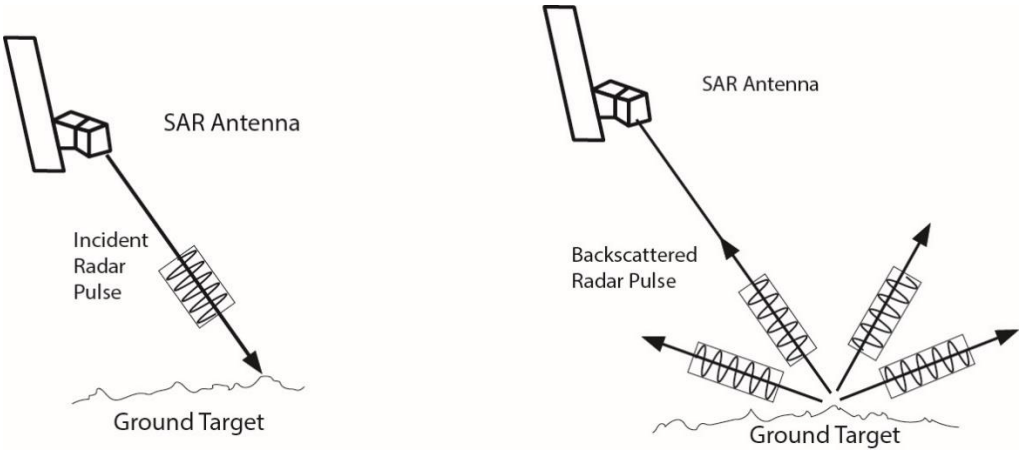


Figure 7 - Radar Antenna
Radar pulse is transmitted from the antenna to the ground (Left), Radar pulse is scattered by the ground targets back to the antenna (Right)

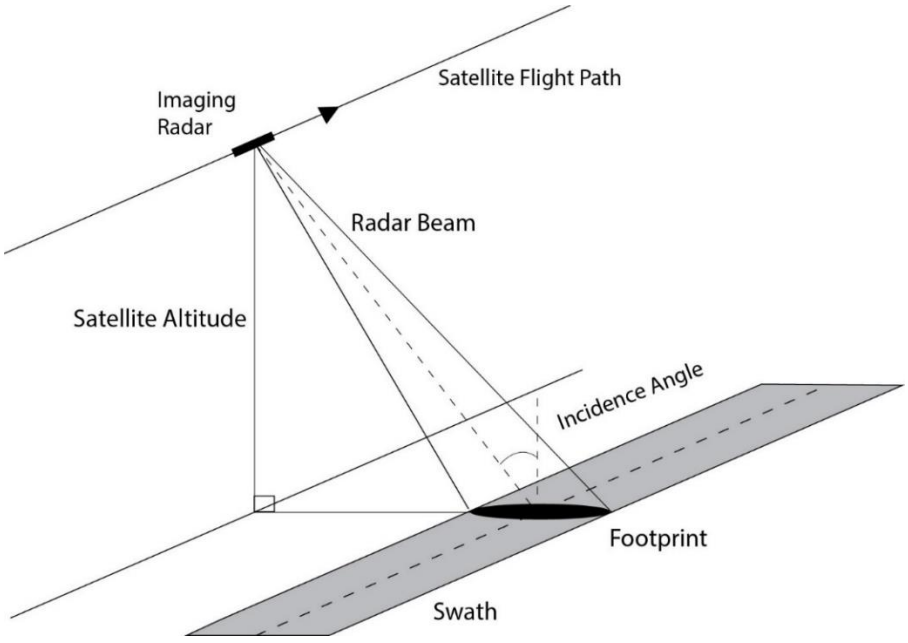


Figure 8 - Imaging geometry for a typical strip-mapping synthetic aperture radar imaging system

The ability of microwave to penetrate clouds, precipitation, or land surface cover depends on its frequency. Generally, the penetration power increases for longer wavelength (lower frequency).

The SAR backscattered intensity generally increases with the surface roughness. However, "roughness" is a relative quantity. Whether a surface is considered rough or not depends on the length scale of the measuring instrument [Figure 9]. If a meter-rule is used to measure surface roughness, then any surface fluctuation of the order of 1 cm or less will be considered smooth [Figure 9]. On the other hand, if a surface is examined under a microscope, then a fluctuation of the order of a fraction of a millimetre is considered very rough [Figure 9]. In SAR imaging, the reference length scale for surface roughness is the wavelength of the microwave. If the surface fluctuation is less than the microwave wavelength, then the surface is considered smooth. For example, little radiation is backscattered from a surface with a fluctuation of the order of 5 cm if a L-band (15 to 30 cm wavelength) SAR is used and the surface will appear dark [Figure 8]. However, the same surface will appear bright due to increased backscattering in a X-band (2.4 to 3.8 cm wavelength) SAR image[Figure 9].

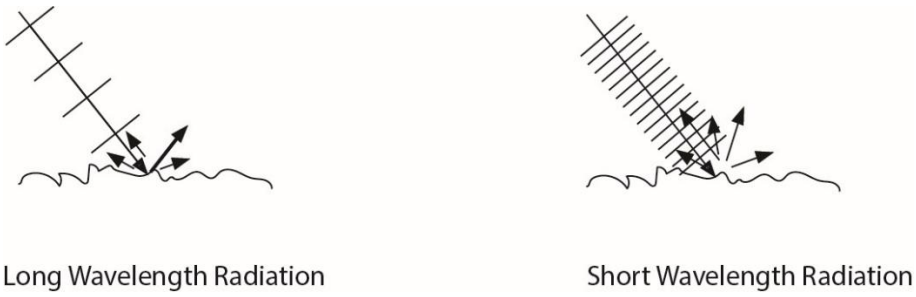


Figure 9 - Long Wavelength vs Short Wavelength Radiation on the same surface.

In the left figure, the land surface appears smooth to a long wavelength radar. Little radiation is backscattered from the surface. In the right figure, the same land surface appears rough to a short wavelength radar. The surface appears bright in the radar image due to increased backscattering from the surface.

Both the ERS, RADARSAT SARs and Sentinel-1 use the C band microwave while the JERS SAR uses the L band. The C band is useful for imaging ocean and ice features. However, it also finds numerous land applications. The L band has a longer wavelength and is more penetrating than the C band. Hence, it is more useful in forest and vegetation study as it can penetrate deeper into the vegetation canopy. In [Figure 10] we can see theoretical response of a pine forest stand to X-, C-, and L-band microwave energy(Jensen, 2014). The shorter the wavelength, the greater the contribution from surface scattering. The longer the wavelength, the greater the penetration into the material and the greater the volume scattering [Figure 10].

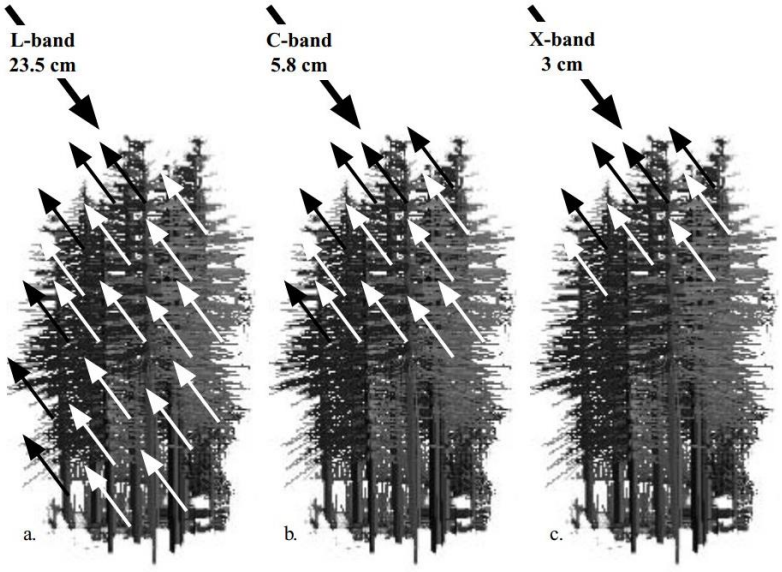


Figure 10 - X-, C-, and L-band microwave energy (Jensen, 2014)

2.2.1. Satellites and Sensors

Copernicus is the European Earth observation program, which was based on a partnership established between the European Union, the European space agency, and the various member states. The objective of this mission is to provide services that allow quick and timely access to accurate and reliable data on the environment, civil protection, and security (*Copernicus*, n.d.). This program integrates three components, one of which is the space component, aimed at observing the Earth through remote sensors, that is, satellites. The Sentinel mission that is part of this program has several satellites, the ones used in this work being the Sentinel-1A and Sentinel-1B.

2.2.2 Sentinel-1

Sentinel-1 is a satellite mission developed by the European Space Agency (ESA) as part of the Copernicus program. It is a constellation of synthetic aperture radar (SAR) satellites designed for Earth observation and monitoring applications. Sentinel-1A was released on April 3, 2014, and Sentinel-1B on April 25, 2016, both of which are still fully operational, sending thousands of data per day.

Here are some key features and information about the Sentinel-1 mission:

- SAR Imaging: Sentinel-1 carries a C-band SAR instrument that operates in the microwave frequency range. SAR imaging allows for all-weather, day-and-night observations of the Earth's surface and have the benefit of penetrating clouds, dust, haze, making it particularly valuable for monitoring and mapping applications to track ground deformation continuously(JC Curlander, 1991).
- Dual Polarization: Sentinel-1 is equipped with dual polarization capability, which means it can transmit and receive radar signals in both horizontal and vertical polarizations. This enables the characterization of the scattering properties of the Earth's surface, providing valuable information about the type and condition of the observed features.
- Global Coverage and Revisit Time: The Sentinel-1 mission is designed to provide global coverage with a short revisit time. The constellation consists of two identical satellites, Sentinel-1A and Sentinel-1B, positioned in the same orbital plane but with a 180-degree phase shift. This configuration allows for frequent revisits to a specific area, providing regular and consistent monitoring capabilities.
- Imaging Modes: Sentinel-1 offers several imaging modes to accommodate different application needs. The main modes include Interferometric Wide Swath (IW), Extra Wide Swath (EW), and Wave (WV). These modes provide different spatial resolutions, coverage areas, and imaging capabilities to cater to a wide range of user requirements.
- Applications: Sentinel-1 data is used for various applications, including land and ocean monitoring, disaster management, ice and snow observations, agriculture monitoring, forest mapping, and urban planning. The data is particularly useful for mapping terrain elevation, monitoring surface deformation, detecting changes in land cover, and observing natural hazards.
- Open Data: One of the key aspects of the Copernicus program is the open and free access to the Sentinel data. Sentinel-1 data is freely available to users worldwide, allowing researchers, scientists, and the general public to access and utilize the data for various applications and research purposes.

2.2.3. Introduction to In-SAR Technology

In-SAR, or Interferometric Synthetic Aperture Radar, is a remote sensing technique that uses radar images to measure changes in the Earth's surface over time. This technology was first developed in the 1970s and has since been used for a variety of applications, including monitoring volcanoes, tracking land subsidence, and measuring glacier movement.

The basic principle behind In-SAR is to compare two or more radar images of the same area taken at different times.

When the radar waves reflect off the Earth's surface, they are affected by any changes in the distance between the satellite and the ground. By comparing the phase of the radar waves in the two images, InSAR can calculate the amount of ground deformation that has occurred.

To create a precise map of ground deformation, InSAR requires many radar images taken over a period. By combining these images, InSAR can create an interferogram, which shows the differences in ground displacement between each pair of images. These interferograms can then be combined to create a detailed map of ground deformation over a wide area.

Satellite images allow you to view Earth from a broader perspective. One of the most important aspects of using satellite images is that you can also browse past images of certain locations. This means that you can track how the area changed over time and predict how it will change in the future. Comparing radar images over time, we can detect ground surface movement.

In recent years, InSAR has become an increasingly popular tool for detecting earthquakes, as it offers several advantages over traditional methods such as GPS and seismometers. Unlike these methods, InSAR can provide continuous measurements over large areas, making it particularly useful for monitoring slow-moving earthquakes and detecting subtle changes in the Earth's surface.

2.2.4. How SAR and In-SAR Work

SAR (Synthetic Aperture Radar) sensors work by transmitting microwave signals towards the Earth's surface and measuring the reflections that bounce back. The signals are transmitted in pulses, and the radar antenna receives the signals that bounce back from the surface. The time delay between the transmitted and received signals is used to calculate the distance from the radar to the surface. This information is used to create a high-resolution image of the surface.

The basic principle of SAR can be described by the following mathematical equations:

- The range equation:

$$R = \frac{c\Delta t}{2} \qquad (1)$$

where R is the distance from the radar to the surface, c is the speed of light,

and Δt is the time delay between the transmitted and received signals.

The Doppler effect, or Doppler shift, is the apparent change in frequency of a wave in relation to an observer moving relative to the wave source (*Wikipedia*, n.d.) [Figure 11].

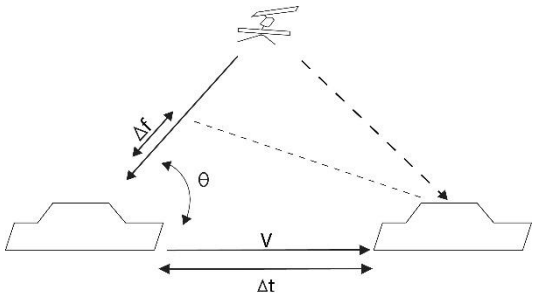


Figure 11 - Doppler effect

The Doppler equation:

- $\Delta f = \frac{2v}{\lambda \cos(\theta)}$ (2)

Where,

- Δf is the Doppler frequency shift,
- v is the velocity of the surface,
- λ is the wavelength of the radar signal,
- and θ is the angle between the radar and the surface.

The range and Doppler equations can be used to create a high-resolution image of the surface. The radar emits microwave signals towards the surface and measures the time delay and frequency shift of the reflected signals. The information is processed to create an image of the surface.

In summary, SAR sensors work by transmitting and receiving microwave signals from an antenna and then processing the signals to create an image of the target surface. The signals are described by mathematical equations, such as the range equation [$R = \frac{c\Delta t}{2}$ (1] and the Doppler equation [$\Delta f = \frac{2v}{\lambda \cos(\theta)}$ (2]. The raw SAR data is a complex-valued function that can be processed using various algorithms to form an image.

InSAR (Interferometric Synthetic Aperture Radar) is a remote sensing technique that uses pairs of SAR images to measure the deformation of the Earth's surface. In-SAR

works by measuring the phase difference between two or more radar images of the same area. When radar waves are emitted from a satellite or aircraft, they bounce off the Earth's surface and return to the sensor. The phase of the returning wave is affected by any changes in the distance between the sensor and the ground surface.

By comparing the phase of the returning radar waves from two or more images, In-SAR can detect changes in the Earth's surface as small as a few millimetres. This makes it an incredibly precise tool for monitoring geological processes and natural hazards.

As we said, InSAR works by comparing the phase difference between the two SAR images acquired at different times and then processing the phase information to create a deformation map. The phase difference between the images is used to calculate the distance that the ground has moved. The resulting image is called an interferogram. Here are some key terms and basic principle of InSAR that can be described by the following mathematical equations:

- Phase: The phase of a radar signal is a measure of the position of the wave relative to a reference point. In InSAR, the phase difference between two radar images is used to calculate the distance that the ground has moved. The phase is usually measured in radians, (0-2 π).
- Interferogram: An interferogram is an image produced by subtracting the phase of one radar image from another. It shows the changes in distance between the ground and the satellite over time. The color scale of an interferogram is typically displayed in units of wavelength, which corresponds to the range of the radar signal.
- Baseline: The baseline is the distance between the satellite positions when the two radar images were acquired. The baseline affects the sensitivity of InSAR to deformation and the spatial resolution of the resulting interferogram. The formula for baseline in the direction of the radar look vector is:
- $$B = \left(\frac{\mathrm{d}}{\sin(\theta)}\right)\left(\frac{1}{\cos(\alpha)}\right) \quad (3)$$

where d is the distance between the satellite and the ground, θ is the angle between the radar look vector and the ground, and α is the angle between the satellite orbit and the ground.

- InSAR phase formula: The formula for the InSAR phase difference between two radar images is:
- $$\Delta\Phi = \frac{4\pi}{\lambda\,(\mathrm{R}^1 - \mathrm{R}^2)} \quad (4)$$

where λ is the radar wavelength, R_1 and R_2 are the distances between the satellite and the ground at the times the radar images were acquired.

The interferometric phase equation:

- $\Delta\Phi = \frac{4\pi}{\lambda\,(\Delta r - \Delta r_0)}$ (5)

where ΔΦ is the interferometric phase difference,

λ is the wavelength of the radar signal,

Δr is the change in range between the two SAR images,

and Δr_0 is the geometric range difference.(Brengman & Barnhart, 2021)

The interferometric phase is proportional to range variations.(Paul A Rosen, 2009)

The deformation equation:

$$\Delta h = \frac{\lambda\Delta\Phi}{4\pi}\, sin(\theta) \quad (6)$$

where Δh is the deformation in the line-of-sight direction,

θ is the incidence angle of the radar signal,

and sin(θ) is the projection factor.

The coherence equation:

$$\gamma = \frac{|<S1S2*|}{|S1||S2|} \quad (7)$$

where γ is the coherence between the two SAR images,

S1 and S2 are the complex-valued SAR images,

And <S1S2*| is the complex conjugate dot product between the two images.

The interferometric phase difference and coherence information can be used to create a deformation map that shows the displacement of the Earth's surface. The deformation map is typically displayed in a color-coded format, where different colors represent different amounts of deformation.(A. Richard Thompson, n.d.)

which is equal to half of the microwave wavelength. [Figure 12]

InSAR uses two or more Synthetic Aperture Radar (SAR) images of the same area taken from the same position (or, for topographic applications, slightly different positions) and finds the difference in phase between them, producing an image known as an interferogram [Figure 12].

satellites use different wavelengths, and that control the amount of ground deformation represented per coloured fringe. Each fringe represents a displacement in half of the wavelength magnitude. In this paper we are using Sentinel-1 images which operate on C-band with ~5.5465763 cm wavelength.

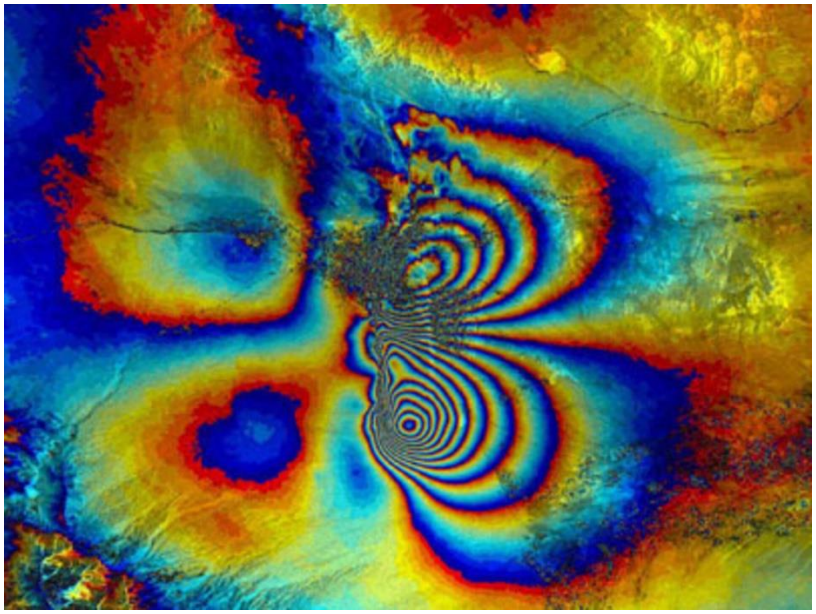


Figure 12 - Interferogram of Bam earthquake

2.3. Artificial Intelligence and subsets

When studying Machine Learning you will come across many different terms such as artificial intelligence, machine learning, neural network, and deep learning. [Figure 13]

Artificial Intelligence (AI) is an umbrella discipline that covers everything related to making machines smarter.

Machine Learning (ML) refers to an AI system that can self-learn based on the algorithm. Systems that get smarter and smarter over time without human intervention is ML.

Artificial Neural Network (ANN) is a method in artificial intelligence that teaches computers to process data in a way that is inspired by the structure and function of biological neural networks in the human brain. It creates an adaptive system that

computers use to learn from their mistakes and improve continuously. Thus, artificial neural networks attempt to solve complicated problems, like summarizing documents or recognizing faces, with greater accuracy. A neural network takes in inputs, which are then processed in hidden layers using weights that are adjusted during training. Then the model spits out a prediction. The weights are adjusted to find patterns to make better predictions.

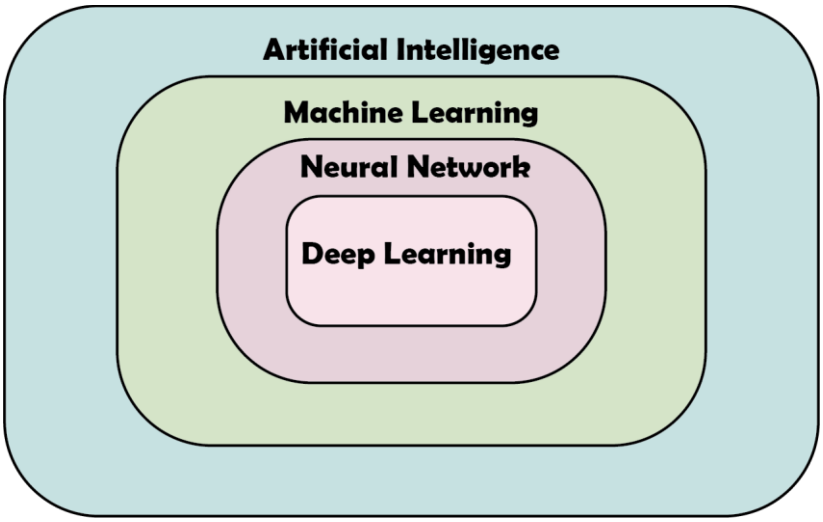


Figure 13 - Artificial Intelligence and subsets

Artificial Neural Network (ANN) consists of interconnected artificial neurons (also called nodes or units) organized in layers [Figure 18]. The nodes receive input signals, perform mathematical operations on them, and produce an output signal that is passed on to the next layer.

The mathematics underlying artificial neural networks involves two main components: the activation function and the learning algorithm.

1. Activation Function:

The activation function determines the output of a node given the weighted sum of its inputs. It introduces non-linearities into the network, enabling it to learn complex patterns and relationships. Common activation functions include:

- The sigmoid function, also known as the logistic function, is a popular activation function used in neural networks. It maps the input value to a range between 0 and 1, which makes it suitable for binary classification problems. The formula for the sigmoid function is equation $\sigma(x) = \frac{1}{1+e^{-x}}$ (8. It has a characteristic S-

shaped curve and is commonly used in the hidden layers of a neural network [Figure 14].

- The hyperbolic tangent (tanh) function is another activation function that is like the sigmoid function. It maps the input value to a range between -1 and 1. The formula for the tanh function is equation $tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$ (9. Like the sigmoid function, it is also commonly used in the hidden layers of neural networks [Figure 15].

- ReLU is a widely used activation function that has gained popularity in recent years. It replaces negative input values with zero and keeps positive input values unchanged. The formula for ReLU is equation $f(x) = max\ (0, x)$ (10. ReLU is computationally efficient and helps address the vanishing gradient problem. It is widely used in deep neural networks and has been shown to provide good performance in various applications [Figure 16].

- Leaky ReLU is a variant of the ReLU activation function that addresses the issue of "dying" ReLU units, where certain neurons can become inactive and not contribute to learning. Leaky ReLU allows a small negative slope for negative input values, which helps prevent these neurons from dying. The formula for leaky ReLU is $\mathrm{f(x) = max\ (0.01x, x)}$ (11, where a is a small constant (e.g., 0.01). It retains the benefits of ReLU while mitigating the dying ReLU problem [Figure 17].

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

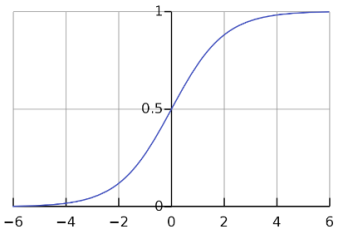


Figure 14 - Sigmoid function

$$tanh(x) = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}} \quad (9)$$

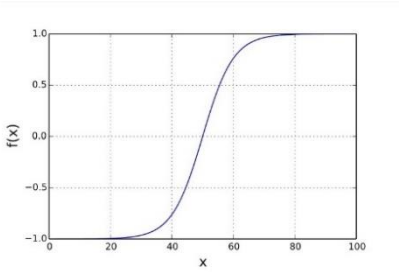


Figure 15 - Hyperbolic tangent function, tanh(x)

$$f(x) = max\,(0, x) \quad (10)$$

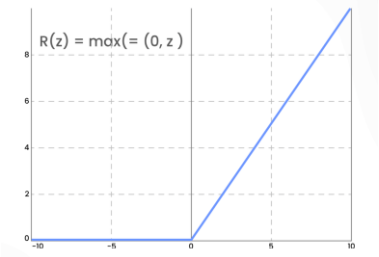


Figure 16 - Relu function

$$\mathrm{f(x) = max\,(0.01x, x)} \quad (11)$$

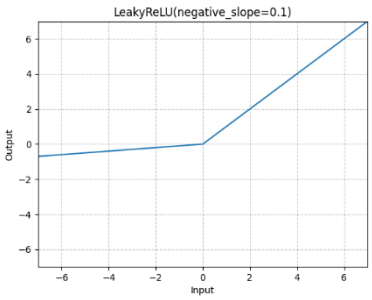


Figure 17 - Leaky Relu function

These activation functions serve different purposes and can be used in various scenarios depending on the nature of the problem and the characteristics of the data. It's important to experiment and choose the most suitable activation function for a specific neural network architecture and task at hand.

2. Learning Algorithm:

The learning algorithm is responsible for adjusting the weights and biases of the network based on training data. The most widely used algorithm for training neural networks is called backpropagation, which consists of two main phases:

- Forward Propagation: During this phase, the input data is fed through the network, and the output is computed layer by layer using the activation functions and the current weights and biases.
- Backward Propagation: In this phase, the error between the network's output and the desired output is calculated. This error is then propagated backward through the network, and the weights and biases are adjusted to minimize the error. This adjustment is typically done using an optimization algorithm such as gradient descent or one of its variants.

The key idea behind backpropagation is to iteratively update the weights and biases of the network by computing the gradient of the error function with respect to these parameters and moving in the direction of the steepest descent.

Overall, the mathematics of artificial neural networks involve linear algebra (matrix operations for computing weighted sums), calculus (for computing gradients and updating weights), and optimization techniques (for finding the optimal set of weights that minimize the error). The complexity and depth of the mathematics depend on the architecture and training algorithm used for a specific neural network model.

In the following, we will see how the architecture of a simple neural network is [Figure 18]:

Let's consider a simple neural network architecture with one input layer, one hidden layer, and one output layer. This architecture is often referred to as a feedforward neural network or a multi-layer perceptron (MLP).

Here's a graphical representation of the neural network [Figure18]:

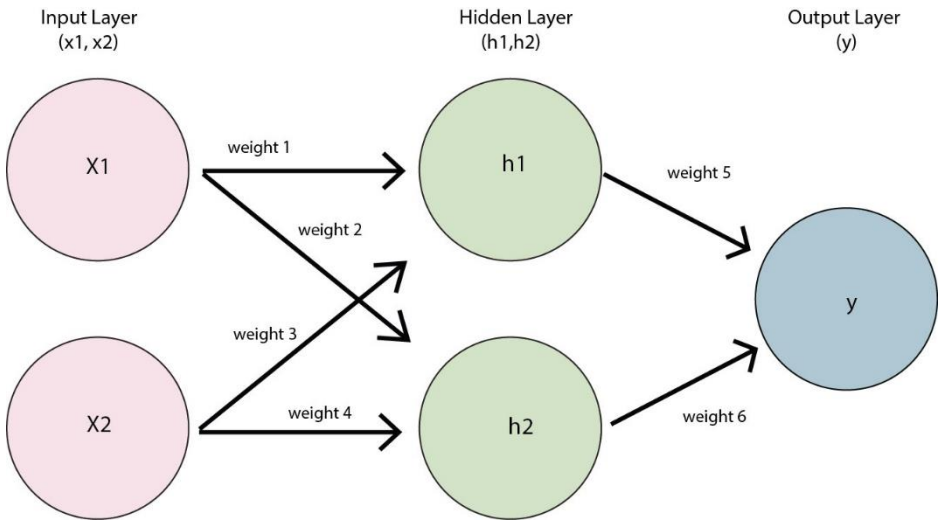


Figure 18 - Architecture of a simple neural network.

In this diagram:

- The input layer consists of two nodes, represented by 'x1' and 'x2'. These nodes receive the input data.
- The hidden layer consists of two nodes, represented by 'h1' and 'h2'. These nodes perform computations based on the input data and apply an activation function.

- The output layer consists of one node, represented by 'y'. This node produces the final output of the network.

Each connection between the nodes is associated with a weight (e.g., w1, w2, w3, w4, w5, w6). These weights determine the strength of the connections between the nodes and are adjusted during the training process.

To compute the output of the hidden layer nodes, we perform the following steps:

1. Multiply the input values (x1, x2) by their corresponding weights (w1, w2, w3, w4).

2. Pass the weighted sum through an activation function (e.g., sigmoid, ReLU) to introduce non-linearities and produce the output values (h1, h2).

Similarly, to compute the final output (y), we perform the following steps:

1. Multiply the output values of the hidden layer nodes (h1, h2) by their corresponding weights (w5, w6).

2. Pass the weighted sum through an activation function to produce the final output value (y).

This is a basic illustration of a neural network architecture [Figure 19]. In practice, neural networks can have multiple hidden layers with varying numbers of nodes, different activation functions, and more complex connections. However, the underlying principles remain similar, involving weighted sums, activation functions, and the flow of information through the network.

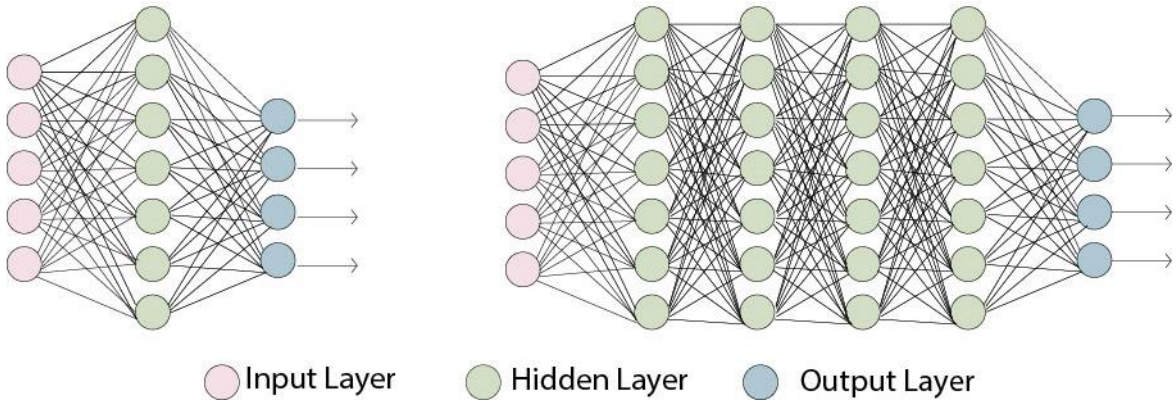


Figure 19 - A deep neural network is simply a neural network with many layers.

Deep Learning (DL) is machine learning (ML) applied to large data sets [Figure 19].

2.3.1. Deep learning application to unstructured data

Machine learning and deep learning also have many subfields, branches, and special techniques. A notable example of this diversity is the separation of Supervised Learning and Unsupervised Learning.

In supervised learning you know what you want to teach the computer, while unsupervised learning is about letting the computer figure out what can be learned. Supervised learning is the most common type of machine learning and will be used on this thesis.

Deep learning (also known as deep structured learning) can solve almost any problem of machine perception, including classifying data, clustering it, or making predictions about it. Deep learning is best applied to unstructured data like images, video, sound, or text.

2.3.2. Deep Learning framework

There are several popular frameworks used for deep learning. Here are some of the most widely used ones:

- TensorFlow: Developed by Google Brain, TensorFlow is one of the most popular deep learning frameworks. It provides a comprehensive ecosystem of tools, libraries, and resources for building and deploying machine learning models. TensorFlow supports both high-level APIs (such as Keras) and low-level APIs for more flexibility.
- PyTorch: Developed by Facebook's AI Research lab, PyTorch is another widely used deep learning framework. It offers a dynamic computational graph, making it easier to define and modify models on the fly. PyTorch has gained popularity for its intuitive and pythonic interface and is favoured by researchers and academics.
- Keras: Originally developed as a user-friendly API for building deep learning models on top of other frameworks, Keras has now become an integral part of TensorFlow. It provides a simple and intuitive interface for constructing neural networks and supports both convolutional and recurrent networks.

On this thesis we are using Python programming language. Python is a high-level programming language that is widely used in various fields, including data science and

machine learning. It has a clean and readable syntax, extensive libraries, and a large community, making it a popular choice for developing deep learning models.

The frameworks used are TensorFlow and Keras. Keras framework is widely used and provide high-level APIs for building and training neural networks. Here's an overview of its framework:

- Keras is a user-friendly, high-level neural networks API written in Python. It is known for its simplicity and ease of use, making it a popular choice for beginners and rapid prototyping.
- It offers a consistent and intuitive API for designing and training deep learning models. It provides a wide range of pre-defined layers, loss functions, optimizers, and evaluation metrics.
- It supports multiple backends, including TensorFlow, Theano, and Microsoft Cognitive Toolkit (CNTK), allowing you to choose the backend that suits your needs.
- With Keras, you can quickly build various types of neural networks, including feedforward networks, convolutional neural networks (CNNs), recurrent neural networks (RNNs), and more.

2.3.3. Convolutional Neural Network, CNN

CNN (Convolutional Neural Network): CNN is a type of neural network commonly used for image recognition and computer vision tasks. Like other Neural Networks, Convolutional Neural Networks has an input, an output, and hidden layers. The difference is that the hidden layers are essential 2D Convolutional layers (Conv2D), 2D tensors (Lecun, 1998).

It is designed to automatically learn and extract meaningful features from input data through a series of convolutional and pooling layers. CNNs are particularly effective in analysing spatial relationships and patterns in images, making them well-suited for tasks like object detection, image classification, and image segmentation [Figure 20], [Figure 21].

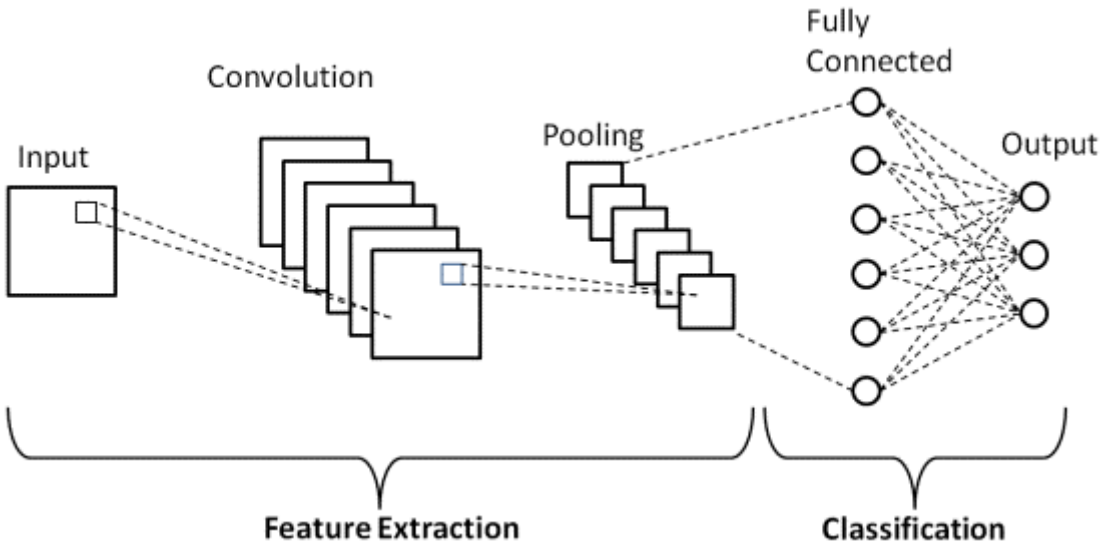


Figure 20 - CNN for gray scale Image

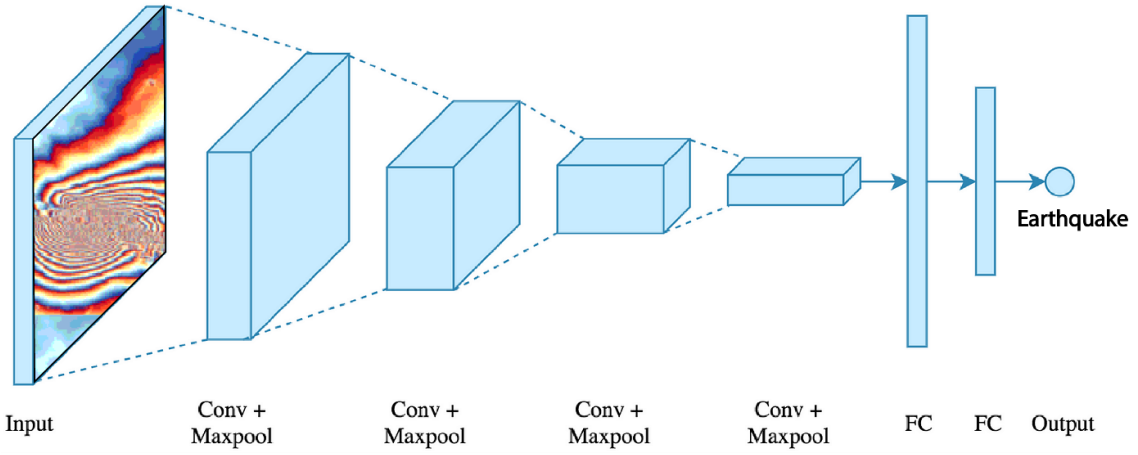


Figure 21 - CNN for RGB Image

Within Deep Learning, a Convolutional Neural Network or CNN is a type of artificial neural network, which is widely used for image/object recognition and classification. Deep Learning thus recognizes objects in an image by using a CNN. Deep learning models are built using neural networks.

Convolutional neural network is composed of multiple building blocks, such as convolution layers, pooling layers, and fully connected layers, and is designed to learn spatial hierarchies of features automatically and adaptively through a backpropagation algorithm.

Training a CNN involves feeding it large amounts of labelled data and adjusting the weights of the network to minimize the error between its predictions and the true labels.

This process is typically done using stochastic gradient descent, a technique that adjusts the weights based on the gradient of the error function.

There are many factors that can affect the performance of a CNN, including the architecture of the network, the size of the training dataset, and the quality of the labels. Researchers are constantly developing new techniques to improve the accuracy and efficiency of CNNs.

Here are the main parameters commonly found in a CNN:

Input size: The dimensions of the input image or data. It is usually represented as height, width, and the number of channels (e.g., RGB images have three channels).

Convolutional layer parameters:

- Number of filters: The number of learnable filters in a convolutional layer. Each filter extracts different features from the input.
- Kernel size: The spatial dimensions of the filters. Common kernel sizes are 3x3, 5x5, or 7x7.
- Stride: The step size at which the filter moves across the input.
- Padding: The number of zero-padding pixels added to the input image to preserve spatial dimensions.
  Pooling layer parameters:
- Pooling type: The type of pooling operation, such as max pooling or average pooling.
- Pooling size: The spatial dimensions of the pooling window.
- Stride: The step size of the pooling operation.

Activation functions: The non-linear functions applied elementwise to the output of convolutional or fully connected layers, such as ReLU (Rectified Linear Unit), sigmoid, or tanh.

Fully connected (dense) layer parameters:

- Number of neurons: The number of neurons in the fully connected layer
- Regularization techniques:
- Dropout: The fraction of neurons to randomly disable during training to prevent overfitting.
- L1/L2 regularization: Regularization terms added to the loss function to reduce overfitting (Ng & A. Y., 2004) .

Optimization parameters:

- Learning rate: The step size for updating the network's weights during training. Adam (Kingma, 2015) is an adaptive learning rate method that computes individual learning rates for different parameters.
- Batch size: The number of samples processed in one forward/backward pass.
- Number of epochs: The number of times the entire dataset has passed through the network during training.

These parameters can be adjusted and optimized to improve the performance and efficiency of a CNN model for specific tasks. Different CNN architectures may have additional parameters specific to their design, but the ones mentioned above are fundamental to understanding the workings of a CNN.

VGG19 and Resents are popular convolutional neural network (CNN) architectures that have been widely used for various computer vision tasks, including image classification.

2.3.3.1 VGG19 (Visual Geometry Group 19)

VGG19 is a variant of the VGG network developed by the Visual Geometry Group at the University of Oxford. It is known for its simplicity and uniform architecture. The "19" in VGG19 represents the total number of weight layers in the network.

The VGG19 architecture consists of several convolutional layers with small 3x3 filters, followed by max-pooling layers 2x2 to reduce the spatial dimensions. The convolutional layers are followed by fully connected layers for classification [Figure 22]. Here's a summary of the VGG19 architecture:

In the VGG19 model summary, the question mark (`?`) is used as a placeholder for the batch size dimension in the input shape. In the case of VGG19, the input shape is specified as `(?, 224, 224, 3)`.

The question mark indicates that the batch size can vary and is not fixed. The other dimensions represent the height, width, and number of channels of the input image, respectively.

During training or inference, you can provide a specific batch size when feeding data to the model. The model can handle different batch sizes, allowing you to process multiple images simultaneously. The question mark in the summary is a placeholder to represent this flexibility.

VGG19 is characterized by its deep architecture and homogeneous structure, which allows it to capture complex image features but also makes it computationally expensive and memory intensive.

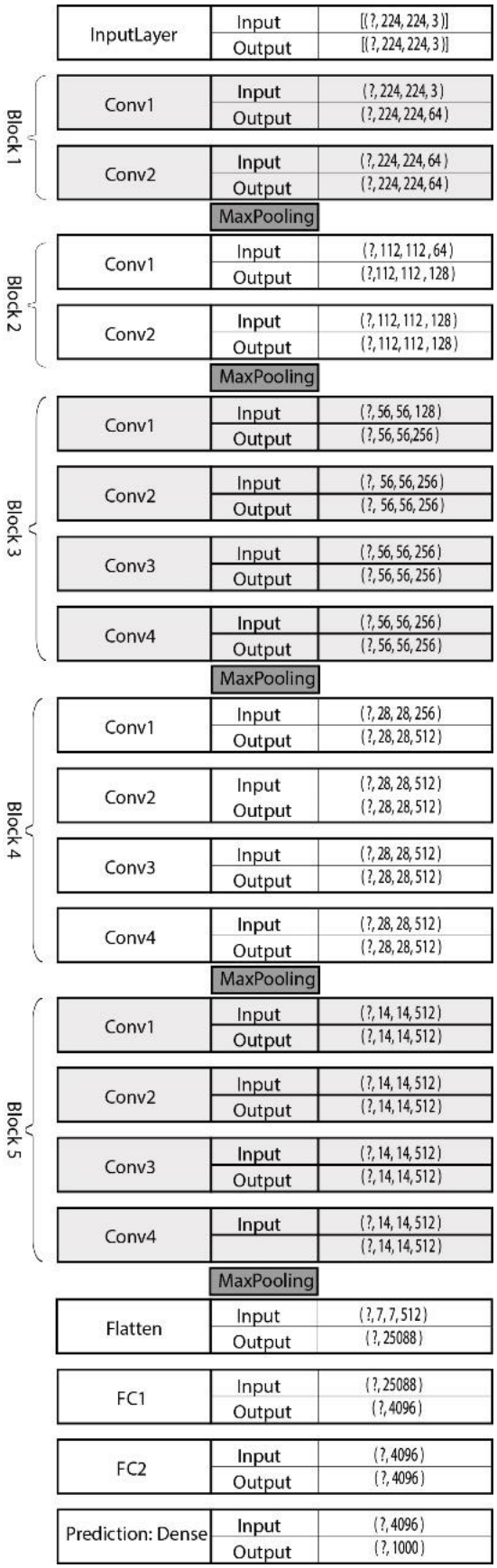


Figure 22 - VGG-19

2.3.3.2. ResNets (Residual Networks)

ResNet is a ground-breaking CNN architecture developed by Microsoft Research. It introduced the concept of residual blocks that significantly improved the training of very deep networks. ResNet was the winner of the 2015 ImageNet competition.

The key idea behind ResNet is the use of residual connections, also known as skip connections, which enable the network to bypass certain layers and learn residual mappings. These connections help alleviate the degradation problem encountered when training extremely deep networks.

Here's a simplified representation of the ResNet-50 architecture, which has 50 weight layers [Figure 23].

Layer	Output Size
Input Image	224 x 224 x 3
Conv7x7-64	112 x 112 x 64
MaxPool3x3	56 x 56 x 64
ResBlock1	56 x 56 x 256
ResBlock2	28 x 28 x 512
ResBlock3	14 x 14 x 1024
ResBlock4	7 x 7 x 2048
AvgPool7x7	1 x 1 x 2048
FC-1000	1000
SoftMax	1000

Figure 23 - Simplified ResNet-50

The ResNet architecture consists of several layers of residual blocks, which contain convolutional layers, batch normalization, and activation functions (typically ReLU). The number of residual blocks varies depending on the specific ResNet variant (e.g., ResNet-50, ResNet-101, ResNet-152).

ResNet models are known for their remarkable depth and performance, enabling them to achieve state-of-the-art results on various computer vision tasks. The skip connections in ResNet allow for efficient gradient propagation and ease the training of deeper networks.

2.3.3.3. InceptionV3, DenseNet201, and Xception

In this study we used other three popular convolutional neural network (CNN) architectures: InceptionV3, DenseNet201, and Xception. Here is a brief review of them.

InceptionV3 is a CNN architecture developed by Google. It is an extension of the original Inception network and is known for its deep and complex structure. InceptionV3 incorporates a module called the Inception module, which uses multiple filter sizes (1x1, 3x3, 5x5) in parallel to capture both local and global features efficiently. This architecture helps to reduce the number of parameters while maintaining good performance. InceptionV3 has achieved excellent results on various image classification tasks and has been widely adopted in computer vision applications.

DenseNet201 is a CNN architecture that introduces the concept of dense connections. It aims to address the vanishing gradient problem and promote feature reuse across layers. In DenseNet, each layer is connected to every other layer in a dense manner. This connectivity pattern enables direct information flow and encourages the network to learn more robust features. DenseNet201 has a high parameter efficiency and has shown remarkable performance in image classification, object detection, and segmentation tasks. It has gained popularity for its ability to learn highly discriminative features from limited training data.

Xception, short for "Extreme Inception," is a CNN architecture proposed by Google. It is inspired by the Inception modules but takes the idea further by replacing the traditional convolutional layers with depthwise separable convolutions. Depthwise separable convolutions split the standard convolution into separate depthwise and pointwise convolutions, reducing the computational complexity. This modification allows Xception to learn more efficient and diverse feature representations. Xception has demonstrated strong performance on image classification tasks and has been utilized in various domains, including fine-grained recognition, object detection, and image segmentation.

These architectures have each made significant contributions to the field of computer vision and have achieved state-of-the-art results in various tasks. Here, we experiment with different architectures and select the one that best suits our needs.

2.3.4. Applications of CNNs

CNNs are being used in an increasing number of applications, from facial recognition software to medical imaging. One notable example is the development of self-driving cars, which rely on CNNs to identify and respond to objects in their environment.

CNNs are also being used in the field of art, where they can be used to generate images or modify existing ones. This has led to the development of new forms of digital art and creative expression.

Convolutional Neural Networks (CNNs) have been successfully applied in the fields of InSAR and earthquake detection. Here are some applications of CNNs in these domains:

1. InSAR Image Processing: CNNs have been used for various tasks in InSAR image processing, including:

- Interferogram Filtering: CNNs can be trained to perform filtering operations on interferograms, reducing noise and artifacts and enhancing the quality of the phase information.
- Phase Unwrapping: Phase unwrapping is a crucial step in InSAR data processing to obtain continuous phase measurements. CNNs have been used to predict phase unwrapping solutions, improving the accuracy and efficiency of the unwrapping process.
- Deformation Monitoring: CNNs can be employed to analyze sequences of interferograms and extract deformation patterns over time. This enables the detection and monitoring of ground movements associated with earthquakes, volcanic activity, subsidence, or landslides.

2. Earthquake Detection and Seismic Signal Analysis: CNNs have shown promise in earthquake detection and seismic signal analysis tasks, including:

- Event Detection: CNNs can be trained to classify seismic signals and identify earthquake events from continuous seismic data. The networks learn to recognize patterns and discriminate earthquake signals from noise and other non-seismic signals.
- Magnitude Estimation: CNNs can be used to estimate the magnitude of earthquakes based on the seismic waveforms. By learning the relationship between input waveforms and corresponding magnitudes, CNNs can provide real-time estimates of earthquake magnitudes.

- Seismic Image Classification: CNNs can process seismic images derived from various seismic data representations, such as spectrograms or time-frequency transforms. They can be trained to classify seismic images into different categories, such as earthquake types or specific geological features.

It's worth noting that the application of CNNs in InSAR and earthquake detection is an active research area, and new advancements and techniques are constantly being explored. These applications demonstrate the potential of CNNs to improve the analysis and interpretation of InSAR data and seismic signals, aiding in better understanding and monitoring of the Earth's surface and seismic activity.

2.4. Deep learning for Earthquake detection

The application of deep learning in InSAR is recent, having found only a few studies where machine learning was used to detect earthquake fringes. However, there are few studies applied to detect deformation from other sources. Most of them are about these applications: Landslides deformation (Liu et al., 2022), deformation prediction from Time-Series(Han et al., 2022), (Ma et al., 2020); Sustained Deformation in InSAR Time Series (Anantrasirichai et al., 2019b); Subsidence deformation (Radman et al., 2021), Volcanic deformation (Anantrasirichai et al., 2019a), (Anantrasirichai et al., 2018), (Sun et al., 2020), (Valade et al., 2019), Glacier deformation (Yang et al., 2022).

2.4.1. Earthquake detection using Deep learning with satellite images.

There is a study about Earthquake and Deep learning with satellite images in May 2020, but it did not use the InSAR Interferograms. The data source is Google Earth satellite images. This Study is Automatic detection of earthquake-induced ground failure effects through Faster R-CNN deep learning-based object detection using satellite images(Hacıefendioğlu et al., 2021). The seismically induced ground failure is defined as any earthquake-generated process that leads to deformations within a soil medium, which in turn results in permanent horizontal or vertical displacements of the ground surface. As a result, relative movements occur on the ground and structures affected by these movements and thus they may be damaged. Determining earthquake-induced ground failure areas is important to carry out damage assessment studies more quickly and reliably and to prevent more destructive damages. Large earthquake-induced ground failure areas or limited access to the areas due to earthquake causes costly and unsafe fieldwork. Using satellite photographs, earthquake-induced ground failure areas can be easily and reliably detected, and the fieldwork can be planned quickly. This study aimed at determining the post-earthquake-induced ground failure areas and buildings or

structures partially ruined (damaged) by using a deep learning-based object detection method, using Google Earth satellite images after an earthquake. The data set obtained after the earthquake occurred in the 2018 Palu region of Indonesia was used. This data set is divided into two parts for training and test areas. A descriptive approach is considered for detecting the earthquake induced ground failure areas and damaged structures from collected images from Google Earth software using satellite photographs, using a pretrained Faster R-CNN. To demonstrate the effectiveness of the proposed method, the data set was first created with Google Earth Pro software and it was generated with 392 images for the earthquake-induced ground failure area and 223 images for the damaged area with a resolution of 1024× 600 pixels. The analyses were carried out by taking into account different image scales. As a result of the analyses, it was concluded that the earthquake-induced ground failure effects (liquefied soil) and damaged structures can be detected to a large extent by using object detection-based deep learning methods(Hacıefendioğlu et al., 2021).

There is only a few papers about Applications of deep learning in detecting Earthquake deformation from InSAR interferograms. First one is Identification of surface deformation in InSAR using machine learning (Brengman & Barnhart, 2021), and the second one is Deformation Fringes Detection in SAR interferograms Using Deep Learning (Silva et al., 2021, 2022).

2.4.2. Identification of surface deformation in InSAR using machine learning

This study, Identification of surface deformation in InSAR using machine learning (Brengman & Barnhart, 2021), presents a CNN, developed to detect, locate, and classify the presence of seismic-like surface deformation within an interferogram. They called this model SarNet. SarNet is trained with using 4×10^6 synthetic interferograms, including both wrapped and unwrapped forward modelled co-seismic-like surface deformation with synthetic noise representative of the atmospheric and topographic noise found in interferograms. The results show that SarNet obtains an overall accuracy of 99.74% on a validation data set. It uses class activation maps (CAMs) to show that SarNet returns the location of surface deformation within the interferogram. They employ a transfer learning method to translate the accuracy of SarNet trained on synthetic data to real interferograms with manually classified co-seismic surface displacement. They train SarNet on 32 interferograms containing labelled co-seismic surface deformation as well as noise. The results show that, through transfer learning, SarNet obtains an overall accuracy of 85.22% on a real InSAR interferograms, and that SarNet returns the location of the surface deformation within the interferogram. SarNet is a first step in automated

detection, location, and classification of co-seismic surface deformation in InSAR and is a starting point for future research using more varied classification schemes, characterizing, and reducing noise in interferograms, and automated real-time detection of surface deformation in interferograms. The sparsity of data, resultant from limited earthquake frequency, as well as variability in surface deformation patterns makes this classification problem difficult. An increase in labelled training data for co-seismic surface deformation will naturally accrue over time, likely increasing the accuracy of this and future networks. Their results suggest that a more diverse classification scheme, separating surface deformation signals by deformation style (e.g. small/broad fringes vs. large/tight fringes) could improve the overall classification ability of this and future networks. They demonstrate the applicability of machine learning techniques in InSAR analysis of co-seismic or co-seismic-like surface deformation and provide an architecture from which to build future machine learning algorithms.

2.4.3. Deformation Fringes Detection in SAR interferograms Using Deep Learning

The other study is Deformation Fringes Detection in SAR interferograms Using Deep Learning (Silva et al., 2021). Main The main objective aims to explore AI techniques, such as Machine Learning and Deep Learning techniques, for the automation of seismic deformations detection in SAR interferograms and to estimate the earthquake's impact area. A dataset with deformation annotations was created from COMET public data to train and evaluate Deep Learning models. The results achieved by the detection methods were satisfactory and supported the idea that they can be used to automatically detect seismic deformations successfully - the best classification model achieved an AUC of 0.86. The segmentation models were more limited, with the best model reaching a Dice coefficient of 0.64 – this result is probably due to the small size of the created dataset. More tests need to be carried out in the near future to test this hypothesis.

2.4.4. Detecting Earthquakes in SAR Interferograms with vision transformer

The most recent study is Detecting Earthquakes in SAR Interferograms with vision transformer. It explores the contribution of deep learning vision transformer models to automatically detect seismic deformation in SAR interferograms. A VGG19 model is trained as baseline and ViT model uses 256×256 pixels patches and the full interferogram. The ViT model outperforms the state-of-the-art both for patch and full interferogram approaches, achieving 0.88 and 0.92 F1- score, respectively. The results achieved show that transformers are a good option, providing even better results than

Convolutional Neural Networks in this test. This way, we are even closer of obtaining results that can create practical products that may help providing a faster response in earthquakes scenarios.

3. Methods

In this paper, we evaluate state-of-the art CNN architectures to detect earthquake fringes. One is self-defined CNN, and the others are some popular pre-trained CNN architectures such as VGG-19, ResNet50, ResNet101, ResNet152, InceptionV3, DeneNet201 and Xception. We check how the results and accuracy of them change. As we mentioned before we created a dataset of interferograms. The interferograms are created by In-Sar techniques using Sentinel-1 images (SAR)(*COMET*, n.d.).

3.1. Data

Data preparation is the process to obtain a dataset of InSAR images that contain earthquake fringes. This dataset should include pairs of interferograms include earthquake fringes and non-earthquake surface displacements.

Since such a collection is not publicly available at the moment, we started producing a dataset from interferograms with and without earthquake fringes. Our team had prepared a dataset from 2019 to 2021, and we expanded this dataset with more interferograms. Now we have a Dataset which contains the interferograms from 2019 to 2023. Therefore, we experiment and record our observations based on this dataset.

We used the Interferograms available at public LiCS database from COMET Centre[1].

First, a python program was created that automatically downloads all the available interferograms for a chosen area. This step was necessary because the website did not have a platform to download multiple data simultaneously and download the interferograms individually was not viable.

In the table below [Table 1], brief information about the selected zones is presented.

Total selected zones are 59 and the selected earthquakes are in the period of 2019 - 2023 [Table 1]. The rest of table 1 is continued in [attachments 2].

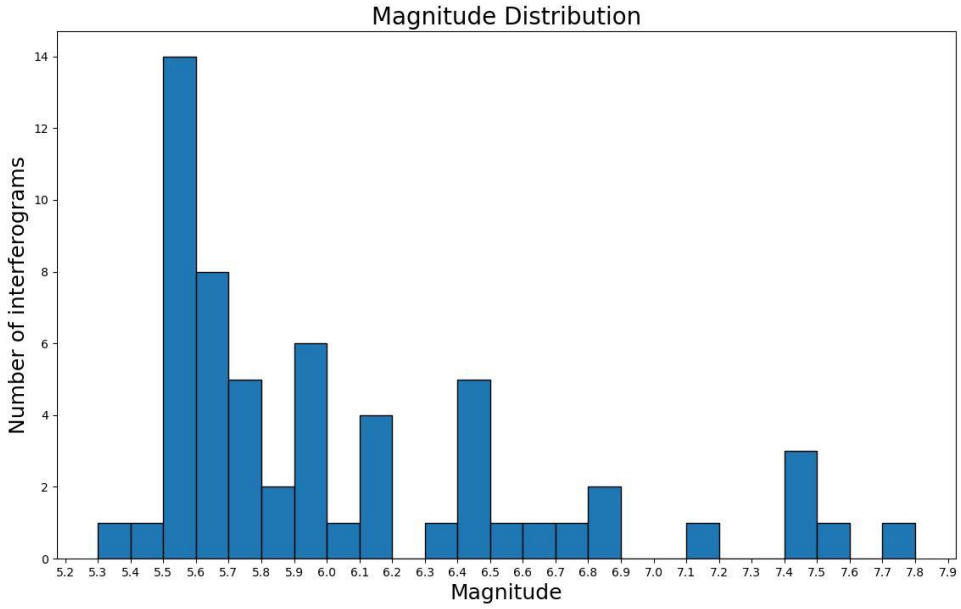
[1] https://comet.nerc.ac.uk/comet-lics-portal/

Table 1 - Selected zones

zone	date	Magnitude	Depth (Km)	Interferograms include earthquake
Álftanes, Iceland	20/10/2020	5.5	10	22
Arzak, China	19/01/2020	6	6	5
Challis, Idaho	31/03/2020	5.6	15	5
Chukotskiy Avtonomnyy, Russia	9/1/2020	6.4	10	10
Dali, China	21/05/2021	6.1	10	4
Elazig, Turkey	24/01/2020	6.7	10	39
…	….	…	…	…

By studying the [Table 1] we can see that the selected earthquakes magnitude varies from 5.3 to 7.8.

Here is a Histogram of magnitude distribution for selected zones.



Graph 1 - Histogram of Earthquake's Magnitude distribution

As we can see from the histogram, most of the earthquake in this period have magnitude between 5.5 to 5.7, which they consider as small magnitude. The fringes are not very easy to detect. For making our train, validation and test set, we should fairly distribute the fringes in each set. Therefor we divide the magnitudes in three categories. Small for 5 to 6, medium for 6 to 7 and large for 7 and above [Figure 24].

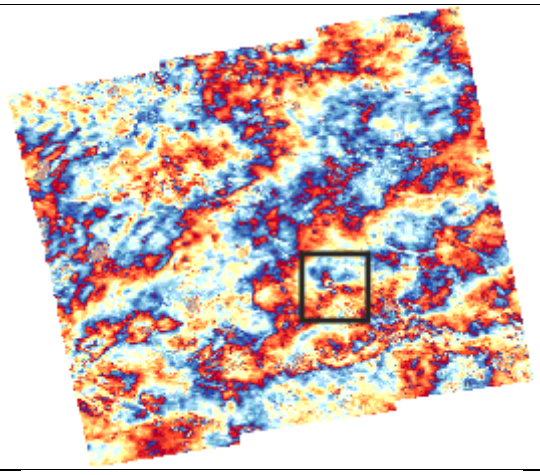
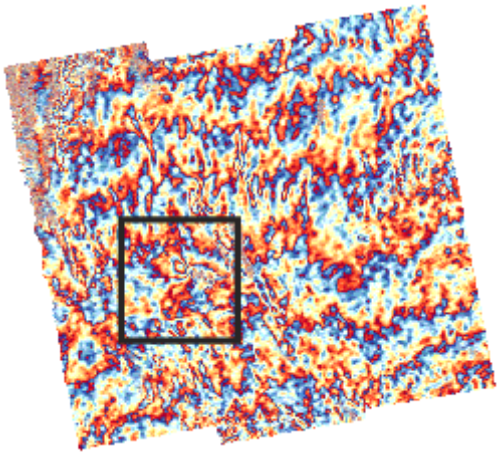
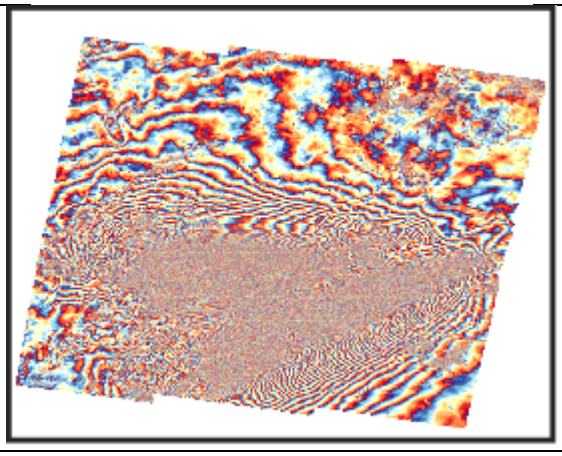
Qinghai, China Magnitude: 5.5	
Dali, China Magnitude: 6.1	
Nurda, Turkey Magnitude: 7.8	

Figure 24 - Fringes of different earthquake's magnitudes

It includes interferograms with earthquake deformation and without earthquake deformation. They are in different dimensions [Figure 24].

3.2. Pre-processing Data and Data Characterization

First of all, we Assigned labels to the InSAR images to indicate whether they contain earthquake fringes or not. The labelled dataset will be the input for our classification CNN models.

It is common to preprocess and normalize the dataset before feeding it into a CNN [Figure 25]. This preprocessing step helps ensure that the input data is consistent in terms of colour channels, intensity range, or any other relevant characteristics. We employed techniques such as resizing and cropping. Resizing involves uniformly scaling the input images to a specific size [Figure 26], while cropping involves extracting a smaller region from larger images [Figure 27].

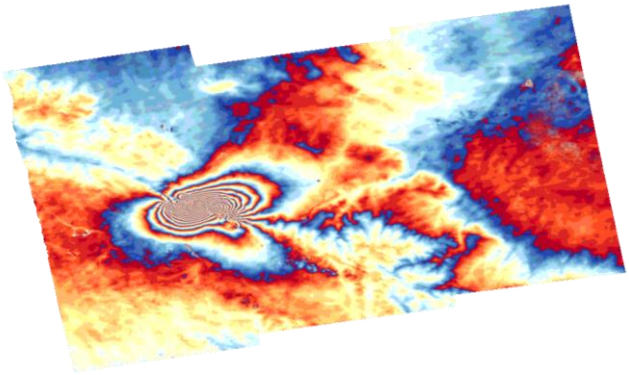


Figure 25 - Dimensions of Primary Interferogram (1019 * 806)

First, we resized the images to 1024*1024 pixels [Figure 26], Then they are cropped to 256*256 window's size [Figure 27].

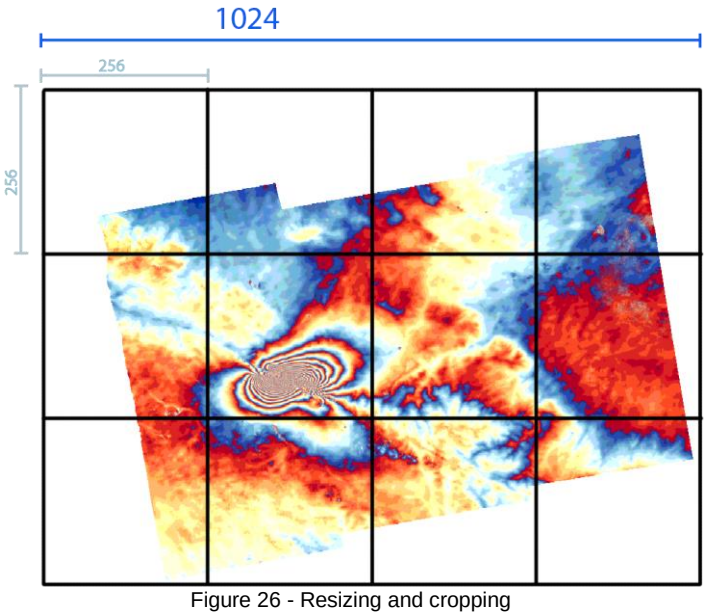


Figure 26 - Resizing and cropping

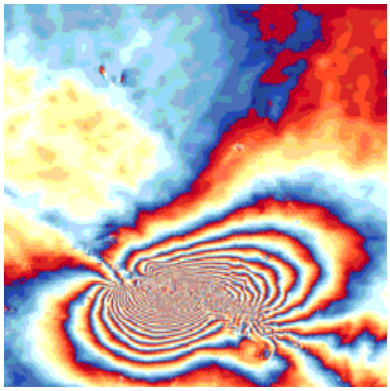


Figure 27 - Labelled as Earthquake

Finally, only the windows that contain more than 90% of earthquake fringe, are saved as positive data and labelled as earthquake [Figure 27].

To assure that models are not trained and evaluated with the same earthquakes interferograms, data was distributed by earthquakes into three sets.

Final Dataset made of train set (~60% of the data), validation set (~25% of the data), and test set (~15% of the data). All sets guaranteed different sizes of earthquakes since earthquakes were individually assigned to their set (Table 2).

Table 2 - Total Dataset (2019- 2023)

Set	Earthquake	Non-Earthquake
Train	653	23057
Validation	423	6126
Test	275	4888

3.3. Training models

We designed a CNN architecture suitable for image classification. Typically, this involves stacking convolutional layers, pooling layers, and fully connected layers.

For activation function we used ReLU for the hidden layers and sigmoid for output layer, because we have binary classification.

Finally, we experimented different network architectures, layer configurations, and hyperparameters to find the best model performance.

We had challenge with imbalanced data. An imbalanced dataset is a dataset where there's a substantial mismatch between the number of records belonging to each category. Real-world datasets can be highly imbalanced, which may affect performance of statistical algorithms or machine learning models. To deal with this challenge we used several techniques such as Data Augmentation and Transfer learning. We will go through each of them in the following.

3.3.1. Data Augmentation

Data augmentation is a technique used to artificially increase the size of a dataset by applying various transformations or modifications to the existing data. It is commonly used in deep learning tasks, such as image classification, to improve model performance and reduce overfitting.

By applying data augmentation, we can create additional variations of our earthquake and non-earthquake images, which helps the model generalize better and improve its ability to classify unseen examples. The specific augmentation techniques and parameters used may vary depending on dataset and the desired transformations. We

experimented with different augmentation techniques such as rotation, shifting, zooming, flipping, and more to find the ones that work best for our problem.

3.3.2. Focal loss

Focal Loss is a loss function specifically designed to address class imbalance in object detection tasks. However, it can also be applied to other classification problems with imbalanced datasets. Focal Loss gives more emphasis to hard, misclassified examples while downplaying the well-classified examples. This helps the model focus on learning from the challenging examples and can lead to improved performance, especially in the presence of imbalanced classes.

The Focal Loss introduces a modulating factor called the "focusing parameter" that downweighs the loss contribution from well-classified examples. The loss function is defined as follows:

- $FL(pt) = -\alpha t(1 - pt)\gamma * log(pt)$

Where:

- `$pt$`is the predicted probability of the true class.
- `$\alpha t$`is the class balancing factor that adjusts the weight given to each class. It can be set to handle class imbalance by assigning a higher weight to the minority class and a lower weight to the majority class.
- `γ` is the focusing parameter that controls the degree of emphasis placed on hard, misclassified examples. A larger value of γ increases the focus on difficult examples.

3.3.3. Transfer Learning

Transfer learning is a popular technique in deep learning, particularly for Convolutional Neural Networks (CNNs), where a pre-trained model trained on a large-scale dataset is utilized as a starting point for a new task. By leveraging the knowledge learned from the source task, transfer learning allows the model to benefit from the features and representations extracted by the pre-trained model. Here's a general approach to implementing transfer learning for our CNN:

We Selected some pre-trained CNN models that has been trained on a large and diverse dataset (ImageNet). The choices are VGG19, ResNet50, ResNet101, ResNet152, Inception, Xception and Densnet201. These models have already learned to extract meaningful features from images and can be a valuable starting point. we imported our pre-trained models, from Keras library.

The fully connected layers at the end of the pre-trained model are specific to the original task on which the model was trained (e.g., ImageNet's 1000-class classification). We Remove these layers, as they are task-specific and not suitable for our specific classification task.

Then we Added new layers to the pre-trained model to adapt it to our specific task Typically, this involves adding a few new layers, including one or more fully connected layers, followed by an output layer. The number of neurons in the final output layer should correspond to the number of classes in our classification problem.

Next, we froze the weights of the pre-trained layers to prevent them from being updated during the initial training phase. This step ensures that the pre-trained weights are preserved, and only the weights of the new layers are updated during training. This is especially important when we have a small dataset, as freezing the pre-trained layers helps prevent overfitting.

Now it is time to train the modified CNN using our own dataset. Since the pre-trained layers are frozen, the initial training phase primarily focuses on updating the weights of the new layers. Gradually, as the new layers learn, we can unfreeze some of the pre-trained layers to fine-tune them along with the new layers. This step can be optional and depends on the size and similarity of your dataset to the original pre-training dataset.

If our dataset is large and similar to the original pre-training dataset, we can consider fine-tuning the pre-trained layers. Unfreeze some of the pre-trained layers and train them along with the new layers. This step allows the model to learn more task-specific features from our dataset.

Next, we evaluated the performance of our trained model on a separate validation set. Fine-tuned the model and hyperparameters. Finally, we tested the model on unseen data to assess its generalization ability.

Transfer learning can significantly benefit our CNN by leveraging the representations learned from large-scale datasets. It helps in cases where we have limited data or computational resources. By starting from a pre-trained model, we can build models that achieve good performance with fewer training samples and training time.

3.4. Evaluation

Evaluation methods help in comparing and selecting the best model among different architectures, hyperparameter settings, or training strategies. By quantitatively evaluating the models, we can determine which one performs better on our classification task. Evaluation methods provide a quantitative measure of how well the CNN model is performing on the classification task. They give us insights into the model's accuracy, precision, recall, F1 score, or other relevant metrics, which can help us understand its strengths and weaknesses. They also allow us to compare our CNN model's performance against established benchmarks or previous state-of-the-art results. This helps to determine if our model is achieving competitive performance or if there is room for improvement.

By evaluating the model's performance under different hyperparameter configurations, you can identify the optimal set of hyperparameters that maximize performance.

Overall, evaluation methods provide quantitative metrics and insights that guide model development, optimization, and decision-making processes. They help to assess the CNN model's performance, understand its limitations, and make improvements to achieve better classification results.

considering the specific requirements of our task, we chose these evaluation metrics.

- Accuracy: Accuracy is the most straightforward evaluation metric and measures the percentage of correctly classified samples out of the total samples in the dataset. It is calculated as the ratio of the number of correctly predicted samples to the total number of samples.
- Precision, Recall, and F1 Score: Precision measures the proportion of correctly predicted positive instances out of the total instances predicted as positive. Recall, also known as sensitivity or true positive rate, measures the proportion of correctly predicted positive instances out of the actual positive instances. F1 score is the harmonic mean of precision and recall provides a single metric to evaluate the model's performance, balancing both precision and recall.
- ROC: Receiver Operating Characteristic (ROC) curve is a graphical representation of the trade-off between the true positive rate (TPR) and the false positive rate (FPR) for different classification thresholds. Area Under the Curve (AUC) is calculated from the ROC curve and provides a single value to measure the model's performance. A higher AUC indicates better performance.

4. Results

In this chapter, the results of the different earthquake fringes detection and classification methods are presented, discussed, and compared to the published related works. First, in section 5.1, we present the results of the classification with a simple CNN which is trained on imbalanced dataset; followed, in section 5.2, by the results of the classification with Transfer learning method.

4.1. Tools and libraries

To use TensorFlow and Keras to build a CNN in Python, we would typically start by installing the necessary libraries and importing them into our Python environment [Table 3]. Then, we can define our CNN architecture using the layers provided by Keras, configure the model with appropriate loss and optimization functions, and train the model using labelled data. Finally, we can evaluate the trained model and use it to make predictions on new, unseen data.

Table 3 - Libraries Versions

Libraries	Version
Python version	3.10.12
Keras version:	2.12.0
TensorFlow version:	2.12.0
NumPy version:	1.22.4
Scikit-learn version:	1.2.2

4.2. Classification with Simple Convolutional Neural Network

We started our image classification with a simple CNN. Here, we go through some settings and details. Finally, we check the results and make a conclusion.

4.2.1. Settings

Here's a brief summary of the model settings:

1. Import the necessary libraries: The code starts by importing the required libraries, including Keras, matplotlib, and scikit-learn.

2. Compile the model: The model is compiled using binary cross-entropy as the loss function, Adam optimizer with a learning rate of 0.0001, and accuracy as the evaluation metric.

Data preprocessing: An instance of the ImageDataGenerator is created to rescale the pixel values of the images. The data generator is then used to load and preprocess the training and testing data from specified directories.

Evaluation: The model is evaluated on the testing data using the evaluate_generator method, which returns the loss and accuracy metrics.

Obtain predictions and true labels: The model is used to predict the labels for the testing data. The true labels are also obtained from the test generator.

Calculate evaluation metrics: Various evaluation metrics, including ROC AUC score, accuracy, F1-score, precision, and recall, are calculated based on the predicted labels and true labels.

Plot ROC curve: The ROC curve is plotted using the false positive rate (fpr) and true positive rate (tpr) obtained from the roc_curve function. The plot shows the trade-off between the true positive rate and false positive rate.

4.2.2. Results

Here's a table summarizing the evaluation metrics obtained from the model:

Table 4 - Simple CNN metrics

Model	Accuracy	Precision	Recall	F1-Score	AUC
Simple CNN	0.94	0.95	0.99	0.97	0.51

Based on the results of evaluation metrics [Table 4], we can make the following conclusions:

- Test Loss: The test loss value is 1.2877. This indicates the average loss of the model's predictions on the test data.
- Test Accuracy: The test accuracy is 0.9403, indicating that the model correctly predicted the class label for approximately 94.03% of the test samples.
- AUC Score: The AUC score is 0.51. The Area Under the Curve (AUC) [Figure 29],is commonly used to evaluate the performance of a binary classifier. A score of 0.51 suggests that the model's predictions are random and have no discriminatory power.
- Accuracy: The overall accuracy is 0.9353, which is the same as the test accuracy mentioned earlier. It represents the proportion of correctly classified samples.
- F1-Score: The F1-score is 0.9666, which is a measure of the model's balance between precision and recall. It combines both metrics, and a value close to 1 indicates good performance.

- Precision: The precision score is 0.9465, representing the proportion of correctly predicted positive instances out of all predicted positive instances. It indicates the model's ability to avoid false positive predictions.
- Recall: The recall score is 0.9875, indicating the proportion of true positive instances that were correctly identified by the model. It represents the model's ability to avoid false negative predictions.

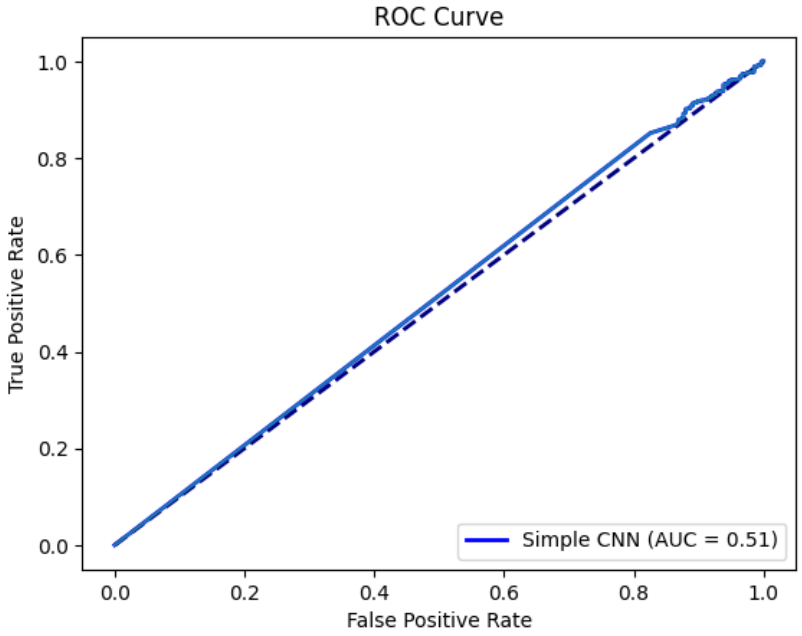


Figure 28 - ROC Curve for Simple CNN

4.2.2. Conclusion

In conclusion, the model achieved high accuracy and F1-score on the test data, indicating good performance in terms of overall classification accuracy and the balance between precision and recall. However, the AUC score suggests that the model's predictions do not have a significant discriminatory power, as it is close to random guessing. It may be necessary to further investigate and improve the model's performance to enhance its discriminatory ability.

4.3. Classification with Transfer Learning

In this part, we try transfer learning techniques. We selected some most popular pretrained architectures and experiment different approaches from each one.

4.3.1. Classification with VGG-19

VGG19 is a deep convolutional neural network architecture. The first model in this part is trained with VGG_19 architecture, and data Augmentation technique.

4.3.1.1. Settings

Here is a summary of VGG_19 with Augmentation settings:

Model architecture:

- The base model is VGG19, pre-trained on the ImageNet dataset, without the top (fully connected) layers, to exclude the fully connected layers from the model. We kept last block of VGG19 model to be trainable. This selective trainable setting is to fine-tune only a portion of the pre-trained model while keeping the earlier layers fixed. By freezing the initial layers, we retain the learned representations and adapt only the later layers to the specific task and dataset we're working on. This approach is often employed when the new dataset is small and similar to the original dataset used to train the pre-trained model.
- Additional layers are added on top of the base model, including convolutional layers, max pooling layers, and fully connected layers.
- The final output layer has 1 neuron with sigmoid activation, indicating binary classification.

2. Training configuration:

- The model is compiled with the Adam optimizer and a learning rate of 0.0001.
- The loss function used is binary cross-entropy.
- The model's performance is measured using accuracy as the metric.

3. Data augmentation:

- The training set is augmented using the ImageDataGenerator class from Keras, with rescaling, horizontal flipping, and shearing.
- The validation and test sets are only rescaled.

4. Training process:

- The model is trained for 20 epochs using the fit() method.
- Model checkpoints are saved using the ModelCheckpoint callback to monitor the validation loss and save the best model.

5. Evaluation and analysis:

- The model's performance is evaluated on the test set using the evaluate() method, calculating the test loss and accuracy.
- Predictions are generated on the test set using the predict() method.
- The ROC curve, AUC score, precision-recall curve and classification report are plotted and printed.

VGG19 with focal loss

The second model is VGG19 which is trained with focal loss. Here is a summary of its settings:

Model architecture:

- Similar to the first model, the base model is VGG19 without the top layers.
- Additional layers are added, including convolutional layers, max pooling layers, and fully connected layers.
- The final output layer has 1 neuron with sigmoid activation, indicating binary classification.

2. Training configuration:

- The model is compiled with the Adam optimizer and a learning rate of 0.0001.
- The loss function used is a custom focal loss function.
- The model's performance is measured using accuracy as the metric.

3. Data preparation:

- The data preparation steps for the training, validation, and test sets are the same as in the first model.

4. Training process:

- The model is trained for 20 epochs using the fit() method.
- Model checkpoints are saved using the Model Checkpoint callback to monitor the validation loss and save the best model.

5. Evaluation and analysis:

- The model's performance is evaluated on the test set using the evaluate() method, calculating the test loss and accuracy.
- Predictions are generated on the test set using the predict() method.
- The ROC curve [Figure 29], AUC score, and classification report, are plotted and printed.

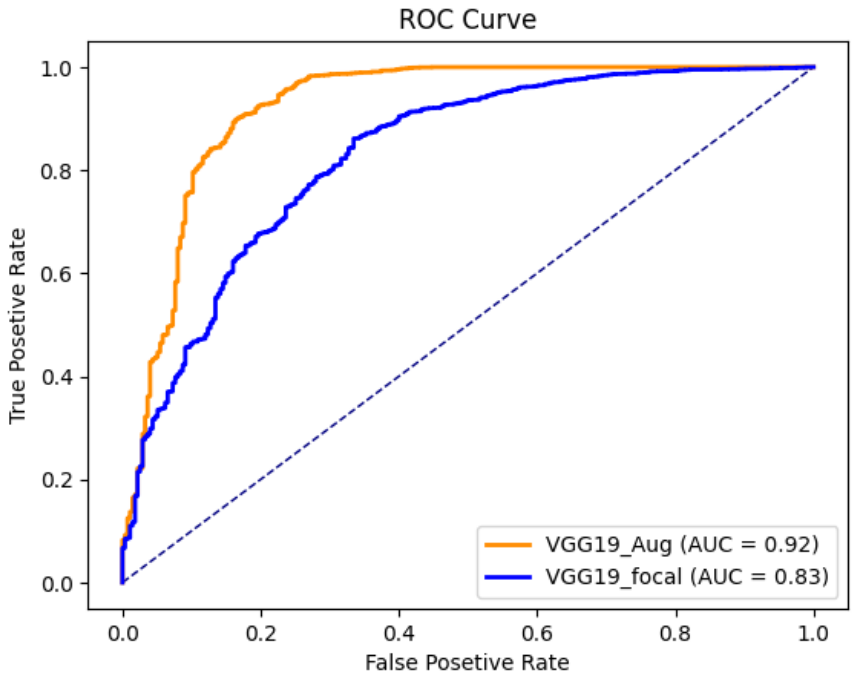


Figure 29 - ROC Curve for VGG19

VGG19 with Focal Loss:

- The model architecture and settings are similar to VGG19 with Augmentation, except for the loss function used.
- The focal loss function is defined separately, which takes two parameters: gamma and alpha. Gamma controls the focusing strength, and alpha balances the contribution of positive and negative samples.
- The model is compiled with the focal loss function instead of binary cross-entropy.
- The model is trained, evaluated, and the evaluation metrics are calculated in a similar way as VGG19 with Augmentation.

In both cases, the models are trained and evaluated using the specified directories for the train, validation, and test sets, with a defined image size, batch size, and number of epochs. The models are saved using model checkpoints, and the evaluation metrics are printed and plotted for comparison between the two models.

5.3.1.2. Results

Based on the obtained training results, we have trained two models using the VGG-19 architecture on the same dataset. One model was trained with data augmentation, while the other model was trained with focal loss. Let's compare the results and draw a conclusion:

Table 5 - VGG19 metrics

Model	Accuracy	Precision	Recall	F1-Score	AUC
VGG-19 with Data Augmentation	0.97	0.97	0.47	0.64	0.92
VGG-19 with Focal Loss	0.95	0.94	0.2	0.3	0.83

From the table [Table 5], we can observe the following:

Based on the provided results, we can draw the following conclusions:

The model VGG-19 with Data Augmentation achieved high accuracy and precision, indicating that it accurately predicted the positive cases. However, the recall and F1-score are relatively low, suggesting that it struggled to correctly identify all the positive cases. The AUC score of 0.92 indicates good overall performance, but there is room for improvement in terms of recall and F1-score.

The VGG-19 with Focal Loss model also achieved a high accuracy and precision, indicating accurate predictions of positive cases. However, the recall and F1-score are significantly lower than the Data Augmentation model, indicating that it struggled to identify a large portion of the positive cases. The AUC score of 0.83 suggests lower overall performance compared to the Data Augmentation model.

5.3.1.3. Conclusion

In conclusion, the VGG-19 model with Data Augmentation showed better overall performance than the VGG-19 model with Focal Loss. While the Data Augmentation model achieved higher accuracy, precision, recall, F1-score, and AUC, the Focal Loss model had lower recall and F1-score, indicating difficulties in correctly identifying positive cases. However, further analysis and experimentation may be needed to determine the best approach and optimize the model's performance for our specific task and dataset.

4.3.2. Classification with ResNet50, ResNet101, ResNet152

At this part, we defined and trained three different models using the ResNet architecture: ResNet50, ResNet101, and ResNet152. Here is a summary of the models settings:

4.3.2.1. Settings

1. ResNet50:

- Load the ResNet50 model without the top (fully connected) layers from the Keras applications.
- Freeze all layers by setting their `trainable` attribute to False, except for the layers in the last block after a specific layer named ‘conv5_block1_1_conv’.
- Create a Sequential model.
- Add the base ResNet50 model to the Sequential model.
- Add additional layers to the Sequential model: two convolutional layers, max pooling layer, flatten layer, two fully connected layers with dropout, and batch normalization.
- Add the final output layer with a sigmoid activation function.
- Compile the model using the Adam optimizer, binary cross-entropy loss function, and accuracy metric.
- Fit the model on the training and validation data using the defined data generators.
- Evaluate the model on the test set using the test data generator.
- Calculate and plot the ROC curve and precision-recall curve for the model.

2. ResNet101:

- Load the ResNet101 model without the top layers.
- Freeze layers, similar to ResNet50.
- Create a Sequential model.
- Add layers to the Sequential model, similar to ResNet50.
- Compile the model, similar to ResNet50.
- Fit the model on the training and validation data, similar to ResNet50.
- Evaluate the model on the test set, similar to ResNet50.
- Calculate and plot the ROC curve and precision-recall curve for the model.

3. ResNet152:

- Load the ResNet152 model without the top layers.
- Freeze layers, similar to ResNet50.
- Create a Sequential model.
- Add layers to the Sequential model, similar to ResNet50.

- Compile the model, similar to ResNet50.
- Fit the model on the training and validation data, similar to ResNet50.
- Evaluate the model on the test set, similar to ResNet50.
- Calculate and plot the ROC curve and precision-recall curve for the model.

Each model follows a similar structure, with the main difference being the base ResNet architecture used (ResNet50, ResNet101, or ResNet152). The models are trained using the same data generators, and their performance is evaluated using common evaluation metrics such as accuracy, loss, ROC curve, and precision-recall curve.

5.3.2.2. Results

let's compare the performance of the three models (ResNet-50, ResNet-101, and ResNet-152) using the metrics achieved. Here's a table summarizing the results:

Table 6 - ResNet metrics

Model	Accuracy	Precision	Recall	F1-Score	AUC
ResNet-50	0.9257	0.9456	0.9779	0.9615	0.6096
ResNet-101	0.9466	0.9467	1	0.9726	0.9619
ResNet-152	0.9358	0.9542	0.9793	0.9666	0.9653

From these results, we can observe the following conclusions:

- AUC Score: ResNet-152 achieved the highest AUC score (0.9653), followed by ResNet-101 (0.9619), and ResNet-50 (0.6096). A higher AUC score indicates better discrimination between the classes.
- Precision: ResNet-152 exhibited the highest precision (0.9542), followed closely by ResNet-101 (0.9467), and ResNet-50 (0.9456). Precision measures the model's ability to correctly classify positive instances.
- Test Loss and Test Accuracy: ResNet-101 achieved the lowest test loss (0.6482) and the highest test accuracy (0.9466). ResNet-152 had the second-lowest test loss (0.2245) and the second-highest test accuracy (0.9358). ResNet-50 had the highest test loss (0.2935) and the lowest test accuracy (0.9257).

- F1-Score: ResNet-101 obtained the highest F1-score (0.9726), followed by ResNet-152 (0.9666), and ResNet-50 (0.9615). The F1-score considers both precision and recall, providing a balanced measure of the model's performance.
- Recall: ResNet-101 achieved perfect recall (1.0000), indicating that it correctly classified all positive instances. ResNet-152 had the second-highest recall (0.9793), and ResNet-50 had a slightly lower recall (0.9779).

5.3.2.3. Conclusion

Overall, ResNet-101 and ResNet-152 consistently performed better than ResNet-50 across most metrics. ResNet-152 showed the highest AUC score and precision, while ResNet-101 exhibited the lowest test loss, highest test accuracy, and highest F1-score. The choice of the best model depends on the specific requirements and priorities of your application.

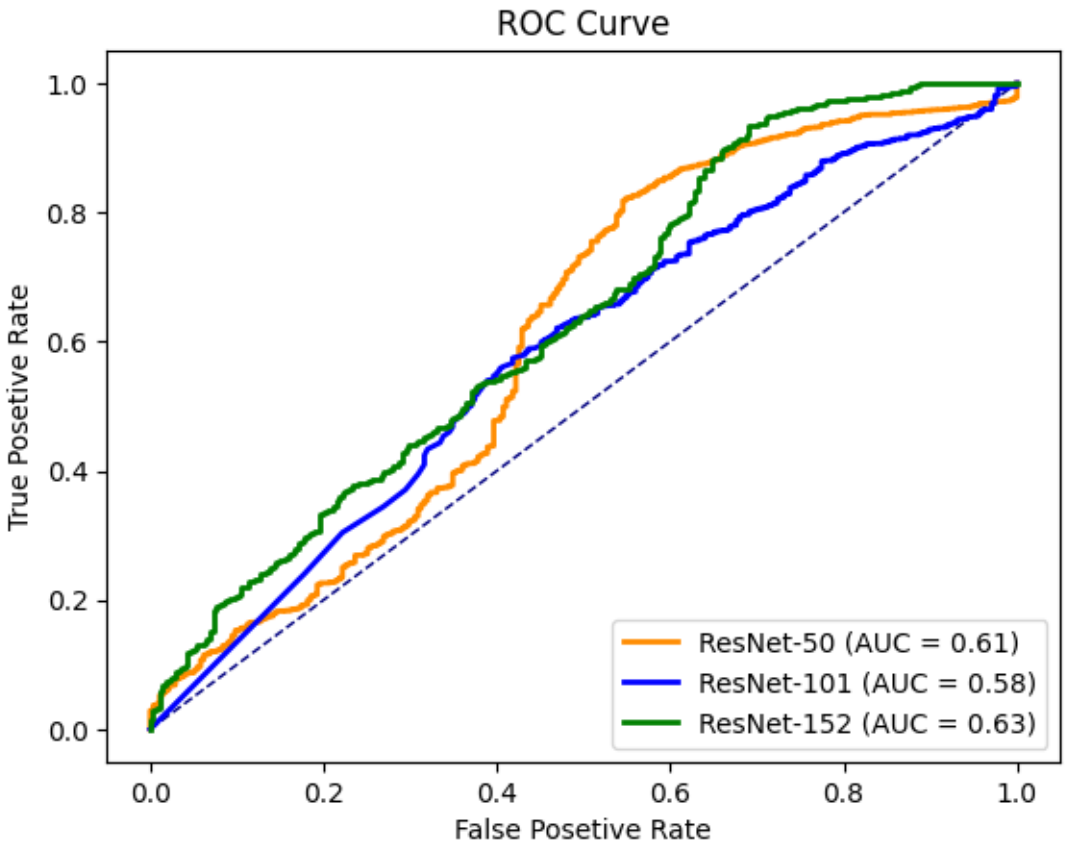


Figure 30 - ROC Curve for ResNet

4.3.3. Classification with InceptionV3, DeneNet201 and Xception

At this part we trained and evaluated three different models for image classification: InceptionV3, DenseNet201, and Xception. Here is a brief summary of the settings and conclusion for each model.

4.3.3.1. Settings

InceptionV3

- Base model: InceptionV3 (pre-trained on ImageNet without the top layers).
- Trainable layers: Only the last block (mixed10) is set as trainable, while the rest are frozen.
- Additional layers: Two convolutional layers, max pooling, flattening, two fully connected layers, dropout, batch normalization, and a final output layer.
- Compilation: Adam optimizer with a learning rate of 0.0001, binary cross-entropy loss, and accuracy metric.
- Training: 20 epochs, using the ImageDataGenerator for data augmentation on the training set.
- Evaluation: Evaluation on the test set using test_loss_InceptionV3 and test_acc_InceptionV3.
- Performance metrics: ROC curve, AUC score, precision-recall curve.

DenseNet201

- Base model: DenseNet201 (pre-trained on ImageNet without the top layers)
- Trainable layers: Only the last block (conv5_block32_concat) is set as trainable, while the rest are frozen
- Additional layers: Two convolutional layers, max pooling, flattening, two fully connected layers, dropout, batch normalization, and a final output layer
- Compilation: Adam optimizer with a learning rate of 0.0001, binary cross-entropy loss, and accuracy metric
- Training: 20 epochs, using the ImageDataGenerator for data augmentation on the training set
- Evaluation: Evaluation on the test set using test_loss_DenseNet201 and test_acc_DenseNet201
- Performance metrics: ROC curve, AUC score, precision-recall curve

Xception

- Base model: Xception (pre-trained on ImageNet without the top layers)
- Trainable layers: Only the last block (block14_sepconv1) is set as trainable, while the rest are frozen
- Additional layers: Two convolutional layers, max pooling, flattening, two fully connected layers, dropout, batch normalization, and a final output layer
- Compilation: Adam optimizer with a learning rate of 0.0001, binary cross-entropy loss, and accuracy metric
- Training: 20 epochs, using the ImageDataGenerator for data augmentation on the training set
- Evaluation: Evaluation on the test set using test_loss_Xception and test_acc_Xception
- Performance metrics: ROC curve, AUC score, precision-recall curve

Each model is trained using the same training, validation, and test directories, as well as the same image dimensions and batch size. The models are saved using checkpoints and the performance metrics are calculated and plotted using matplotlib.

5.3.3.2. Results

Here's a table comparing the results:

Table 7 - InceptionV3, DensNet201 and Xception metrics

Model	Accuracy	Precision	Recall	F1-score	AUC
InceptionV3	0.95	0.961	0.988	0.974	0.73
DenseNet201	0.971	0.977	0.993	0.985	0.94
Xception	0.963	0.971	0.991	0.981	0.83

Based on the achieved metrics [Table 7], we can draw the following conclusions about the models:

Accuracy: DenseNet201 has the highest accuracy (97.1%), followed closely by Xception (96.3%) and InceptionV3 (95.0%).

- Score: DenseNet201 achieves the highest F1 score (0.985), indicating a good balance between precision and recall. Xception follows closely with an F1 score of 0.981, while InceptionV3 has a slightly lower F1 score of 0.974.

- AUC: DenseNet201 has the highest AUC score (0.940), indicating a strong performance in terms of ranking the predicted probabilities. Xception has an AUC of 0.830, and InceptionV3 has the lowest AUC score of 0.730.
- Precision: DenseNet201 has the highest precision (0.977), followed by Xception (0.971) and InceptionV3 (0.961). Precision measures the proportion of true positive predictions among the total predicted positives.
- Recall: DenseNet201 achieves the highest recall (0.993), indicating its ability to correctly identify the majority of positive samples. Xception has a slightly lower recall of 0.991, while InceptionV3 has the lowest recall of 0.988.

5.3.3.3. Conclusion

Overall, DenseNet201 appears to be the most robust model based on its high accuracy, F1 score, AUC [Figure 1], precision, and recall. However, both Xception and InceptionV3 also demonstrate strong performance across multiple metrics, making them viable options depending on specific requirements.

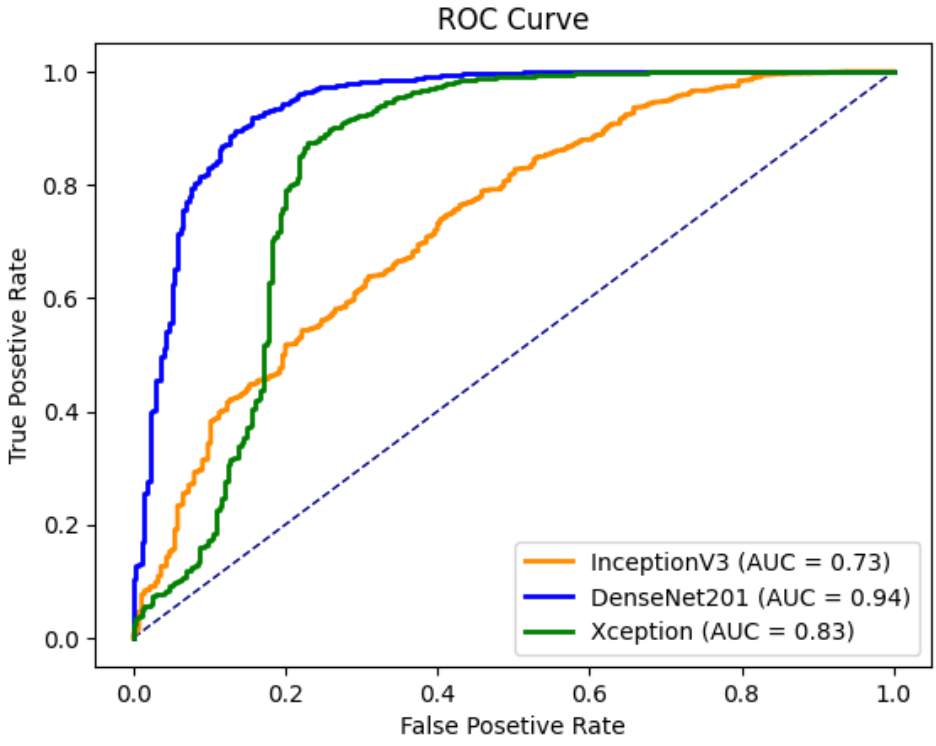


Figure 31 - ROC Curve for InceptionV3, DensNet201, Xception

4.4. Overall comparison and discussion

Based on the provided results, we can analyze the performance of each model and determine the best one:

- Simple CNN: The Simple CNN model achieves a moderate accuracy of 94%, with a relatively high precision of 0.95 and F1-score of 0.97. However, the low recall (0.99) suggests a high number of false negatives, resulting in an imbalanced performance. The AUC score of 0.50 indicates poor discriminatory power.
- VGG-19 with Data Augmentation: This model improves upon the Simple CNN with a higher accuracy of 97%. It demonstrates balanced precision (0.97) and F1-score (0.64), but the low recall (0.47) suggests it may struggle to correctly identify positive samples. The AUC score of 0.92 indicates good discriminatory power.
- VGG-19 with Focal Loss: While achieving a similar accuracy of 95% to the Simple CNN, this model shows lower precision (0.94), recall (0.20), and F1-score (0.30), indicating suboptimal performance. The AUC score of 0.83 is relatively higher, but it still falls behind other models.
- ResNet-50: This model achieves an accuracy of 92.57%, with a balanced precision (0.9456) and F1-score (0.9615). However, the low recall (0.9779) suggests a relatively high number of false negatives. The AUC score of 0.6096 indicates limited discriminatory power.
- ResNet-101: With an improved accuracy of 94.66%, this model demonstrates balanced precision (0.9467), recall (1.0000), and F1-score (0.9726). The AUC score of 0.9619 indicates good discriminatory power.
- ResNet-152: Similar to ResNet-101, this model achieves a high accuracy of 93.58% and demonstrates balanced precision (0.9542), recall (0.9793), and F1-score (0.9666). The AUC score of 0.9653 indicates strong discriminatory power.
- InceptionV3: This model achieves a slightly higher accuracy of 95%, with balanced precision (0.961) and F1-score (0.974). The recall (0.988) indicates a good ability to identify positive samples. However, the AUC score of 0.730 is comparatively lower.
- DenseNet201: With the highest accuracy of 97.1%, this model demonstrates balanced precision (0.977), recall (0.993), and F1-score (0.985). The AUC score of 0.940 indicates good discriminatory power.

- Xception: This model achieves an accuracy of 96.3% with balanced precision (0.971), recall (0.991), and F1-score (0.981). The AUC score of 0.830 indicates relatively good discriminatory power.

Based on these results, DenseNet201 emerges as the best-performing model. It exhibits the highest accuracy, balanced precision, recall, and F1-score. Additionally, it achieves a good AUC score, indicating its strong discriminatory power. However, it's worth considering the specific requirements and trade-offs associated with each model before making a final decision.

5. Conclusion

This dissertation focuses on exploring Deep Learning techniques for the detection of earthquake fringes in SAR interferograms and the classification of seismic deformations. The research begins with an extensive literature review, which covers the fundamental concepts of InSAR, Artificial Intelligence, and Deep Learning. State-of-the-art Deep Learning methods applied to InSAR data are identified through this review.

Based on the insights gained from the literature review, a methodology is developed to address the classification of earthquake fringes in InSAR tasks. It is determined that Sentinel-1 interferograms provide sufficient information for studying seismic deformations, and Deep Learning methodologies are well-suited for detecting deformation fringes. While the challenge of limited annotated datasets is acknowledged, the research demonstrates that successful training of Deep Learning models is feasible.

To train classification models, an InSAR dataset is created and preprocessed through normalization and manipulation techniques suitable for model input. However, the dataset proves to be imbalanced due to the small size of deformation areas compared to areas without deformations. In most interferograms, no deformations are present at all. To address this issue, the research explores two approaches: "data augmentation" and a loss function called "focal loss".

In the preliminary classification experiments, a simple convolutional neural network is created and trained with the imbalanced dataset. The achieved results for F1-Score and AUC are 0.97 and 0.51 respectively, indicating room for improvement. To facilitate comparison, transfer learning is employed by training and evaluating several popular pretrained models, namely VGG19, ResNet50, ResNet101, ResNet152, InceptionV3, DenseNet, and Xception.

To mitigate the impact of imbalanced data, "Data Augmentation" is applied to all selected pretrained architectures. Additionally, a special focus is given to the VGG19 model by incorporating the "Focal loss" as its loss function. As a result, two VGG19 models are trained: one with data augmentation on its train set, and the other with the focal loss. All other pretrained models are evaluated with the use of data augmentation.

The choice of these techniques and models is driven by time constraints and the valuable insights gained from this experiment. It is acknowledged that further experimentation is warranted, and future goals include exploring additional techniques and models.

Based on the obtained results, DenseNet201 demonstrates superior performance compared to other models. It achieves remarkable accuracy (0.971) along with well-balanced precision (0.977), recall (0.993), and F1-score (0.985). Furthermore, DenseNet201 exhibits a strong discriminatory power, as indicated by a high AUC score of 0.94.

These findings suggest that the implementation of deep learning methods has yielded promising outcomes. However, we acknowledge that further advancements can be made in the future by delving deeper into the application of these techniques. Continued exploration and refinement of deep learning methodologies hold the potential for even more significant improvements in the results.

The work carried out in the scope of this dissertation, allowed to answer to the 3 research questions:

Question 1: Although there are few published papers about detecting earthquake' fringes by Deep Learning, how efficiently is it possible to get better results with the increasing dataset?

Answer 1: Comparing with the results achieved by similar methods in two years ago with smaller dataset, we can see that we had a big improvement on final results. Their best classification model which was VGG19, achieved an AUC of 0.86, But in our studty the best model, DensNet201, achieved an AUC of 0.94 and with VGG-19 with Data Augmentation, achieved an AUC of 0.92.

Question 2: How does the depth or the number of layers in variants of a specific CNN architecture (e.g. ResNet) affect the result of earthquake classification?

Answer 2: According to the results presented in [Table 6], we can see that, the deeper variants (ResNet-101 and ResNet-152) generally outperform ResNet-50 in earthquake classification based on the provided metrics. ResNet-101 achieves the highest precision, recall, F1-score, and AUC, indicating superior overall performance. However, it's important to note that these observations are specific to the given dataset and task.

5.1. Challenges and Limitations

Despite their impressive capabilities, convolutional neural networks (CNNs) encounter several challenges and limitations. One prominent issue is the necessity for large amounts of labeled data, which can be both time-consuming and costly to obtain. Additionally, there is a concern regarding potential bias in the training data, leading to inaccurate or unfair predictions.

Moreover, CNNs have certain limitations in tasks that demand more intricate reasoning or contextual comprehension. To address these challenges and broaden the capabilities of deep learning, researchers are actively working on developing novel network architectures and techniques.

In conclusion, CNNs provide a powerful tool for enhancing the accuracy and efficiency of InSAR measurements. Despite the existing challenges and limitations, the potential advantages of employing CNNs in InSAR are substantial. Continued advancements in addressing these challenges and exploring new techniques will further amplify the benefits of utilizing CNNs in the field of InSAR.

5.2. Future work

Combining Convolutional Neural Networks (CNNs) with Interferometric Synthetic Aperture Radar (InSAR) data opens up several exciting possibilities for future work. Here are a few potential directions you could explore:

- Change Detection: InSAR provides detailed information about ground surface deformations over time. By employing CNNs, you can develop algorithms to automatically detect and classify changes in the radar images, such as infrastructure damage, land subsidence, or geological events. This can have applications in disaster monitoring, urban planning, and environmental monitoring.
- Feature Extraction: InSAR data contains rich spatial and temporal information. CNNs can be used to extract meaningful features from InSAR images, enabling improved analysis and interpretation. You can explore different CNN architectures and techniques to capture important patterns and structures, facilitating tasks such as land cover classification, landform mapping, or target detection.
- Image Registration: InSAR datasets often require precise image registration due to their multi-temporal nature. CNNs can assist in accurate and automated image registration, reducing manual effort and improving the quality of the interferometric analysis. By training CNNs to align InSAR images, you can enhance the co-registration process, leading to more reliable deformation measurements.
- Denoising and Artifact Removal: InSAR data is susceptible to noise and artifacts caused by various factors such as atmospheric disturbances, system errors, or geometric distortions. CNNs can be employed to denoise InSAR images and

remove unwanted artifacts, enhancing the quality and reliability of the deformation maps and other derived products.

- Fusion with Optical Imagery: CNNs can be utilized to fuse InSAR data with optical imagery, such as aerial or satellite images. By integrating these complementary data sources, you can exploit the strengths of each modality to improve land cover classification, object detection, or change analysis. The fusion of CNN-based analysis with multi-sensor data can provide a more comprehensive understanding of the observed phenomena.
- Deep Learning for InSAR Inversion: InSAR inversion is the process of estimating physical parameters such as surface displacements or deformation models from the interferometric data. Deep learning techniques, including CNNs, can be explored to improve the accuracy and efficiency of the inversion process, enabling more precise estimation of geophysical properties from InSAR observations.

These are just a few potential research directions that combine CNNs and InSAR. Depending on our interests and expertise, we can delve into any of these areas and contribute to advancing the field by developing novel algorithms, exploring new architectures, or addressing specific challenges in InSAR data analysis using deep learning techniques.

References

A. Richard Thompson, J. M. M. and G. W. S. J. (n.d.). *Interferometry and Synthesis in Radio Astronomy*.

Anantrasirichai, N., Biggs, J., Albino, F., & Bull, D. (2019a). A deep learning approach to detecting volcano deformation from satellite imagery using synthetic datasets. *Remote Sensing of Environment*, *230*. https://doi.org/10.1016/j.rse.2019.04.032

Anantrasirichai, N., Biggs, J., Albino, F., & Bull, D. (2019b). The Application of Convolutional Neural Networks to Detect Slow, Sustained Deformation in InSAR Time Series. *Geophysical Research Letters*, *46*(21), 11850–11858. https://doi.org/10.1029/2019GL084993

Anantrasirichai, N., Biggs, J., Albino, F., Hill, P., & Bull, D. (2018). Application of Machine Learning to Classification of Volcanic Deformation in Routinely Generated InSAR Data. *Journal of Geophysical Research: Solid Earth*, *123*(8), 6592–6606. https://doi.org/10.1029/2018JB015911

Brengman, C. M. J., & Barnhart, W. D. (2021). Identification of Surface Deformation in InSAR Using Machine Learning. *Geochemistry, Geophysics, Geosystems*, *22*(3). https://doi.org/10.1029/2020GC009204

COMET. (n.d.). https://comet.nerc.ac.uk/comet-lics-portal/

Copernicus. (n.d.).

Hacıefendioğlu, K., Başağa, H. B., & Demir, G. (2021). Automatic detection of earthquake-induced ground failure effects through Faster R-CNN deep learning-based object detection using satellite images. *Natural Hazards*, *105*(1), 383–403. https://doi.org/10.1007/s11069-020-04315-y

Han, J., Yang, H., Liu, Y., Lu, Z., Zeng, K., & Jiao, R. (2022). A Deep Learning Application for Deformation Prediction from Ground-Based InSAR. *Remote Sensing*, *14*(20). https://doi.org/10.3390/rs14205067

Sousa Joaquim. (2021). *Mestrado em Deteção Remota INTERFEROMETRIA SAR*.

JC Curlander, R. M. (1991). *Synthetic aperture radar systems and signal processing*. New York: John Wiley & Sons.

Jensen, J. R. (2014). *Remote sensing of the environment : an earth resource perspective*. Pearson.

Kingma, D. P. , & B. J. L. (2015). Adam: A method for stochastic optimization. . *A Method for Stochastic Optimization. 3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings, 1–15.*

Lecun, Y. , L. B. Y. B. and P. H. (1998). *Gradient-based learning applied to document recognition*.

Liu, Y., Yao, X., Gu, Z., Zhou, Z., Liu, X., Chen, X., & Wei, S. (2022). Study of the Automatic Recognition of Landslides by Using InSAR Images and the Improved Mask R-CNN Model in the Eastern Tibet Plateau. *Remote Sensing*, *14*(14), 3362. https://doi.org/10.3390/rs14143362

Ma, P., Zhang, F., & Lin, H. (2020). Prediction of InSAR time-series deformation using deep convolutional neural networks. *Remote Sensing Letters*, *11*(2), 137–145. https://doi.org/10.1080/2150704X.2019.1692390

Ng, & A. Y. (2004). L1 and L2 regularisation comparisation. . *Proceedings of the 21 St International Conference on Machine Learning.*

Paul A Rosen. (2009). *Principles and Theory of Radar Interferometry*.

Radman, A., Akhoondzadeh, M., & Hosseiny, B. (2021). Integrating InSAR and deep-learning for modeling and predicting subsidence over the adjacent area of Lake Urmia, Iran. *GIScience and Remote Sensing*, *58*(8), 1413–1433. https://doi.org/10.1080/15481603.2021.1991689

Silva, B., Sousa, J. J., & Cunha, A. (2022). Detecting Earthquakes in SAR Interferogram with Vision Transformer. *International Geoscience and Remote Sensing Symposium (IGARSS)*, *2022-July*, 739–742. https://doi.org/10.1109/IGARSS46834.2022.9883523

Silva, B., Sousa, J. J., Lazecky, M., & Cunha, A. (2021). Deformation Fringes Detection in SAR interferograms Using Deep Learning. *Procedia Computer Science*, *196*, 151–158. https://doi.org/10.1016/j.procs.2021.11.084

Sun, J., Wauthier, C., Stephens, K., Gervais, M., Cervone, G., la Femina, P., & Higgins, M. (2020). Automatic Detection of Volcanic Surface Deformation Using Deep Learning. *Journal of Geophysical Research: Solid Earth*, *125*(9). https://doi.org/10.1029/2020JB019840

Valade, S., Ley, A., Massimetti, F., D'Hondt, O., Laiolo, M., Coppola, D., Loibl, D., Hellwich, O., & Walter, T. R. (2019). Towards global volcano monitoring using

multisensor sentinel missions and artificial intelligence: The MOUNTS monitoring system. *Remote Sensing*, *11*(13). https://doi.org/10.3390/rs11131528

wikipedia. (n.d.).

Yang, Z., Xi, W., Yang, Z., Shi, Z., & Qian, T. (2022). Monitoring and Prediction of Glacier Deformation in the Meili Snow Mountain Based on InSAR Technology and GA-BP Neural Network Algorithm. *Sensors*, *22*(21), 8350. https://doi.org/10.3390/s22218350

Attachments

Attachments 1.

Here's a simple example of how you can build a CNN using Tensorf

low and Keras [Figure 33].

```python
import tensorflow as tf
from tensorflow import keras

# Define the CNN architecture
model = keras.models.Sequential([
    keras.layers.Conv2D(32, kernel_size=(3, 3), activation='relu', input_shape=(64, 64, 3)),
    keras.layers.MaxPooling2D(pool_size=(2, 2)),
    keras.layers.Flatten(),
    keras.layers.Dense(64, activation='relu'),
    keras.layers.Dense(10, activation='softmax')
])

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

# Train the model
model.fit(x_train, y_train, batch_size=32, epochs=10, validation_data=(x_val, y_val))

# Evaluate the model
loss, accuracy = model.evaluate(x_test, y_test)
print('Test loss:', loss)
print('Test accuracy:', accuracy)

# Make predictions
predictions = model.predict(x_test)

```

Figure 32 - Simple CNN model with two convolutional layers.

In the example above, we create a simple CNN model with two convolutional layers, a max pooling layer, a flatten layer, and two dense layers. The model is compiled with an optimizer, loss function, and evaluation metrics. We then train the model using labelled data and evaluate its performance. Finally, we use the trained model to make predictions on test data.

Note that the example assumes you have appropriately formatted input data (`x_train`, `y_train`, etc.) and the required data pre-processing steps have been performed (e.g., normalization, resizing images to a consistent size).

This is just a basic example to give you an idea of how to use Python, TensorFlow, Keras, and CNNs together. Depending on your specific task and dataset, you may need to modify the architecture, hyperparameters, and training process accordingly.

Attachments 2.

zone	date	Magnitude	Depth (Km)	Interferograms include earthquake
Grindavík, Iceland	24/02/2021	5.6	10	37
Hotan, China	25/06/2020	6.4	10	14
Idgah, Pakistan	30/12/2019	5.4	14	7
Kanallakion, Greece	21/03/2020	5.7	10	14
Karakenja, Tajikistan	24/01/2020	5.5	10	2
Kirkagac, Turkey	22/01/2020	5.6	6	20
Magna, Utah	18/03/2020	5.7	12	8
Mamurras, Albania	26/11/2019	6.4	22	24
Mohr, Iran	9/6/2020	5.5	10	19
Nagqu, China	19/03/2021	5.7	10	11
Oaxaca, México	23/06/2020	7.4	26	14
Petrinja, Croatia	29/12/2020	6.4	10	24
Qinghai, China	21/05/2021	7.4	10	29
Ridgecrest, California	6/7/2019	7.1	8	29
Saray, Turkey	23/02/2020	5.8	10	17
Tallaboa, Puerto Rico	7/1/2020	6.4	9	3
Tonopah, Nevada	15/05/2020	6.5	3	28
Turt, Mongolia	11/1/2021	6.8	10	3
Týrnavos, Greece	4/3/2021	5.5	10	26
Xegar, China	20/03/2020	5.7	10	13
Xizang, China	22/07/2020	6.3	10	12
western Xizang, China	29/03/2021	5.6	10	14
Yedisu, Turkey	14/06/2020	5.9	10	17
Bandare-e lenge, Iran	1/7/2022	6.1	10	7
Masjed soleyman, Iran	4/10/2021	5.6	10	2
Nurda, Turkey	6/2/2023	7.8	18	12
Khowy, Iran	18/01/2023	5.8	10	3
Ekinozu, Turkey	6/2/2023	7.5	10	11
Tsurib, Russia	8/12/2022	5.5	10	7
Duzce, Turkey	23/11/2022	6.1	4	14
Qala I Naw, Afghanistan	17/1/2022	5.3	19	13
Afghanistan	21/6/2022	5.9	10	9
Pakistan	6/10/2021	5.9	9	8
Novobod, Tajikistan	22/2/2023	5.9	6	1
Murghob, Tajikistan	23/2/2023	6.8	21	14
Kyrgyzstan	28/12/2022	5.6	10	15
Rikaze, China	30/11/2021	5.5	5	6
Nagqu, China	14/8/2022	5.5	10	9
Northen-Qinghai, China	19/10/2022	5.5	9	0
Qinghai, China	16/6/2021	5.5	10	16
Laojunmiao, China	25/8/2021	5.6	10	0
Laojunmiao, China	23/1/2022	5.6	10	9
Qinghai, China	21/5/2021	7.4	10	12
Dali, China	21/5/5/2021	6.1	10	7

Tianpeng, China	9/6/2022	5.9	14	0
Linqiong, China	1/6/2022	5.9	10	6
Kangding, China	5/9/2022	6.6	10	13
Phôngsali, Laos	24/12/2021	5.7	10	0
Altai, Mongolia	17/1/2022	5.5	10	0
Leleti, Romania	14/2/2023	5.5	10	0
Flórina, Greece	9/1/2022	5.5	10	0
grat, Ethiopia	26/12/2022	5.5	10	10
Lambaréné, Gabon	4/12/2022	5.5	10	0