SAR-to-Optical domain transformation for easier image recognition

HUGO MIGUEL CARVALHO CURINHA
(Licenciado em Engenharia Informática, Redes e Telecomunicações)

Dissertação para obtenção do grau de mestre em Engenharia Informática e de Computadores

Orientador: Doutor Paulo Alexandre Carapinha Marques

Júri:
Presidente: Doutor Nuno Miguel Soares Datia
Vogais:
Doutor Andrea Radius
Doutor Paulo Alexandre Carapinha Marques

Outubro, 2024

SAR-to-Optical domain transformation for easier image recognition

HUGO MIGUEL CARVALHO CURINHA
(Licenciado em Engenharia Informática, Redes e Telecomunicações)

Dissertação para obtenção do grau de mestre em Engenharia Informática e de Computadores

Orientador: Doutor Paulo Alexandre Carapinha Marques, ISEL/IPL

Júri:
Presidente: Doutor Nuno Miguel Soares Datia, ISEL/IPL
Vogais:
Doutor Andrea Radius, ICEYE
Doutor Paulo Alexandre Carapinha Marques, ISEL/IPL

Outubro, 2024

Agradecimentos

Inicialmente, quero expressar a minha sincera gratidão ao Instituto Superior de Engenharia de Lisboa (ISEL), que me proporcionou um ambiente enriquecedor, onde me foi dada a chance de crescer academicamente, onde me foi permitido explorar diversas áreas e criar um kit de ferramentas para enfrentar o mundo. Queria também agradecer a todos os professores com a qual tive a oportunidade de ter aulas e que me enriqueceram. Gostaria também de agradecer pessoalmente ao Presidente do ISEL José Nascimento, à Vice-presidente Cristina Borges e à Presidente do pedagógico Sandra Aleixo e ao coordenador de mestrado Nuno Datia por todo o apoio dado e por me ter sido oferecido esta oportunidade de participar na jornada deste projeto.

Um agradecimento especial ao meu orientador de tese Doutor Paulo Marques. O Doutor Paulo Marques foi uma peça fundamental para o desenvolvimento desde projeto e para o meu desenvolvimento tanto académico como pessoal, e que graças a todos os esforços de mentoria, me ajudou a tornar um Engenheiro mais completo e capaz. Felizmente tive a oportunidade de conhecer o Doutor Paulo Marques enquanto professor, que desde cedo mostrou um grande cuidado e respeito pelo próximo, e sempre conseguiu cativar os alunos a explorar e a desafiarem-se, apoiando em todos os passo. Rapidamente conseguiu captar-me para um mundo desconhecido de radares pela qual acabei por criar um grande interesse, e que me ofereceu perspetivas sobre um mundo que de outro modo não teria tido a oportunidade de descobrir e de ganhar tanto interesse. Agradeço imenso todo o apoiou que sempre me foi dado, estando presente para me aconselhar e guiar tanto académicamente como pessoalmente, incentivando-me sempre a tornar-me um melhor engenheiro e uma melhor pessoa, e agradeço imenso toda a orientação e paciência que o Doutor Paulo Marques teve no desenvolvimento desta tese.

Gostaria também de agradecer imenso aos meus irmãos de faculdade Inácio, Shiko, Soares, Mafalda, Ana, Ficalho, Guilherme, André, Álvaro e aos Tomás, que sem estes o percurso teria sido muito mais tortuoso e que sempre foram um pilar para superar quaisquer desafios. Foi um percurso árduo e com vários desafios, porém foram estes que me permitiram crescer e que me deram as ferramentas para enfrentar os diversos desafios que me esperam. As longas noites de trabalho árduo irão ficar para sempre comigo. Percorremos imenso juntos e agradeço por termos partilhado este percurso.

Durante a vida conheci pessoas, que por diversos motivo rapidamente se tornaram pilares importantes nas minhas diversas etapas da vida.

Aos meus irmãos de secundário Pedro, Barata, Jorje, Viegas, Daniela, Marta, Filipa, Mariana, Rita, que são irmãos que me acompanham desde cedo, com a qual pode experenciar diversas etápas da minha vida e que se tornaram como família, aos meus irmãos que conheci online Maegl, Nez, Ben e Gabriel, que começou por um interesse niche em comum e rapidamente se tornaram irmãos para levar para a vida, aos meus irmãos de estágio Costa, Mafalda, Alice e Silke que a vida nos trouxe ao acaso mas nos tornamos inseparaveis e que apesar de há distancia posso chamar de irmãos, e aos meus irmãos de trabalho Diego e Clayton que sempre me apoiaram tanto profissionalmente como pessoalmente, ajudando-me a superar diversos desafios, a todos os maiores agradecimentos por todo o apoio que me deram.

Aos meus avós Nidle e Luis o agradecimento é imenso que sempre fizeram de tudo para os próximos estarem bem e felizes, que sofrem com os problemas dos outros, não descansado até terem garantias. Não há palavras que descrevam a alegria e a gentileza e por todos os impossíveis que fizeram para me verem feliz e o quão estão sempre prontos a enfrentar qualquer Goliath que a vida lhes atirar. O meu amor por vocês vai para além do infinito.

Aos meus avós Domingo e Ana, que não estão mais conosco, que foram pessoas alegres, bondosas e sábias, oferecendo uma perspetiva calma e calculada sobre qualquer situação. Pelo que lutaram e alcançaram sempre foram fonte de inspiração e sempre conseguiram transmitir segurança e serinidade aos que vos rodeiam. Agradeço todas as memórias que criei e todo o apoio e ensinamento passado, o meu amor por voês é eterno e levarei as vossas memórias para a vida.

Porém o maior e dos mais profundos dos agradecimentos vai para os meus pais Filipe e Carla Curinha, não há palavras que descrevam o quão vos amo, o quão importantes sempre foram e sempre serão na minha vida, o quão vos admiro e idolizo, e o quão vos agradeço por tudo o que fizeram e me ensinaram. Existem valores instriscos que devem ser protegidos e quando observados é impossivel não se tornarem um objetivo. A bondade, a força de vontade, a perseverancia, a alegria e a preocupação nascem consoco, porém é ao longo da vida que são nutridos e amplificados pelos ambientes que nos rodeiam. Por todo o esforço que fizeram, por todas a batalhas que enfrentaram, por todos os bons e maus momentos, por todas as piadas e os momentos de alegria, por todos os abraços apertados, mas especialmente por todo o carinho, agradeço-vos por tudo e amo-vos do fundo do meu coração. Um especial agradecimento à minha mãe, não é possivel expressar o quão te amo, o quão te admiro pela postura serena e sábia face a qualquer situação, o quão resiliente uma pessoa pode ser, capaz de fazer face a tudo e a todos, expressando sempre o mais profundo e sincero carinho que só pode ser protegido. Obrigado do fundo do coração por tudo o que fizeram por mim.

Statement of integrity

I declare that this dissertation / project work / internship report is the result of my personal and independent research. Its content is original, and all sources listed in the bibliographic references were consulted and are duly mentioned in the text. I further declare that all scientific and technical references relevant to the development of the work are duly cited and included in the bibliographic references.

The author

Assinado por: **Hugo Miguel Carvalho Curinha**
Num. de Identificação: 15437353
Data: 2024.12.06 17:53:49+00'00'

Resumo

Num mundo constantemente a evoluir, com desafios como alterações climáticas, levou ao desenvolvimento de sistemas espaciais capazes de captar imagens aérias de zonas de interesse.

Sistemas óticos são sistemas passivos que utilizão a radiação emitida pelo sol para captar imagens. Estes sistemas operam na banda do visível, que para um ser humano oferece imagens que refletem a nossa visão, o que facilita a sua interpretação. No entanto, a sua eficácia depende de condições luminosas e meteorológicas. Para superar estas limitações, os radares de abertura sintética (SAR) foram desenvolvidos, uma vez que, devido ao facto de serem ativos e à frequencia de operação, torna-os independentes das condições climatéricas. No entanto, os sistemas SAR captam o retroespalhamento do sinal, observável numa imagem grayscale, o que dificulta a interpretação.

Para combinar as vantagens de ambos, foi criado um modelo de convoluções neuronais, baseado em autoencoders, que converte imagens SAR em imagens óticas. O modelo, composto por um codificador e um descodificador, comprime a informação das imagens SAR numa representação latente e tenta reconstruir a versão ótica correspondente. Um problema destes modelos é a perda de contexto à medida que a imagem passa pelos layers da rede. Para supererar este desafio "skip connections" foram implementados. Estes transportam informação de camadas do codificador para o descodificador, melhorando a reconstrução da imagem.

Para testar o modelo, utilizou-se métricas, como o erro quadrático médio, que compara pixel a pixel a diferença entre o valor real e o produzido para cada banda de cor. Outra função avaliada foi o índice de similaridade estrutural, que para além de captar a informação de cor, capta também a informação estrutural da imagem como o contraste.

Foram também obtidas imagens manuais de Sentinel-1 e Sentinel-2, devido à escassez de datasets. Apesar das limitações computacionais que impediram o uso de redes mais complexas, a abordagem demonstrou ser capaz de converter imagens SAR em imagens óticas.

Palavras-chave

Radares de abertura sintética; Redes neuronais; Redes neuronais convolucionais; Sistemas de deteção remota ótica; Transformações de domínio.

Abstract

Optical imaging systems work by capturing the light's photons, that upon interaction with objects, scatter randomly. By sensing and processing the received particles it's possible to create meaningful representations in the form of images. These systems offer rich spectral information, as they capture the frequencies in the visible band, which makes it desirable when being visually analyzed by humans as these reflect how humans see. Yet one of the biggest limitations is that these are heavily affected by weather and lighting conditions. Some cloud coverage or the lack of light can be enough to block the image acquisition, making these systems unusable unless coupled with other types of sensors such as Infrared. Sythenthic Aperture Radar(SAR) are remote sensing systems, that offer significant advantages in land monitoring, scientific research, security applications, and others. It is an active remote sensing technique that uses microwave signals to obtain high-resolution images of the Earth's surface. Because of the frequencies that are usually used these are less susceptible to weather conditions, meaning it's possible to obtain images whether day or night and no matter the cloud coverage. However, inherent limitations, such as speckle noise or geometric distortions, and the fact that it loses some crucial information like color information, make human interpretation of these a complex task.

This thesis proposes a Convolutional Neural Network (CNN)-based autoencoder that extracts the relevant information from the SAR image and reconstructs the optical counterpart, therefore enabling image interpretation by non-specialized personnel.

Keywords

Convolutional neural network; Domain transformation; Neural networks; Optical remote sensing systems; Synthetic aperture radar.

Index

Figure index

Table index

Abbreviations

AI Artificial Intelligence. 22, 33, 46, 55, 75, 76

AIS Automatic Identification System. 2, 3

API Application Programming Interface. 67, 91

CFAR Constant False Alarm Rate. xiii, 92, 93, 98

CIE International Commission on Illumination. 58

CNN Convolutional Neural Network. 3, 23, 24, 31, 33, 34, 40, 48, 50–52, 54, 95

CPU Central Processing Unit. 76–78

DEM Digital Elevation Model. 62, 74

EC European Commission. 59

ELU Exponential Linear Unit. xii, 40, 41

EMR Electromagnetic Radiation. 5, 6

ESA European Space Agency. 59, 67

EW Extra Wide Swath Mode (SAR). 59

FPN Feature Pyramid Networks. 95, 96

GAN Generative Adversarial Networks. 55, 91, 95, 97

GPU Graphics Processing Unit. 76–78

GRD Ground Range Detected (SAR data). 60, 61, 67

GUI Graphical User Interface. 67

IR Infrared. 2, 10, 21

IW Interferometric Wide Swath Mode (SAR). 59, 63

KAN Kolmogorov-Arnold Networks. 103, 104

LSTM Long short-term memory. 54

ML Machine Learning. 8, 21, 22, 27, 45, 46

MSE Mean Squared Error. xii, xiii, 22, 41–45, 55, 80–84, 88, 95

NLP Natural Language Processing. 76

NN Neural Network. 8, 21–24, 27, 31, 33, 35, 36, 41, 42, 44, 45, 48, 103, 104

OCN Ocean Component Network. 60, 61

OSW Ocean Swell Spectra. 61

OWI Ocean Wind Fields. 61

PCA Principal Component Analysis. xii, 29, 30

PRF Pulse Repetition Frequency. 14

PRI Pulse Repetition Interval. 14

PSNR Peak Signal-to-Noise Ratio. 45

RAM Random Access Memory. 76, 78

ReLU Rectified Linear Unit. xii, 38–40, 55, 79

RGB Red Green Blue (color space). xi, xii, 28–30, 56–58, 79, 85, 88

RNN Recurrent Neural Network. 38, 54

RVL Surface Radial Velocities. 61

SAR Synthetic Aperture Radar. xi–xiii, xv, 1–4, 10–13, 15–19, 21, 24, 27–31, 50, 51, 53, 54, 56, 59, 60, 63, 67, 68, 72–74, 79–84, 91–93, 95, 97, 98, 101

SGD Stochastic Gradient Descent. 46

SLAR Side Looking Airborne Radar. 11, 12

SLC Single Look Complex (SAR data). 60, 67, 74

SM StripMap Mode (SAR). 59

SSIM Structural Similarity Index. xii, xiii, 43, 44, 55, 80–84, 95

tanh Hyperbolic tangent. 36–38, 65

VRAM Video Random Access Memory. 51, 55, 75–78, 91

WV Wave Mode (SAR). 59

WWW World Wide Web. 1

Chapter 1

Introduction

Since the dawn of time, information has been a cornerstone of human civilization as it is one of the most valuable possessions one can have. It allows us to construct knowledge bases, from which we can take critical decisions with the assurance of having all the data needed for a more complete and accurate picture of the problem. This can be applied to all fields, from medical tests and scans to classify diseases, to city planning and construction. As humanity progressed, the role of information expanded dramatically, especially with the introduction of the World Wide Web (WWW) in the late 20th century. This triggered an exponential growth in data generation and accessibility, transforming how we gather, analyze, and disseminate information.

Through the internet, societies were able to create an interconnected world where data can be shared instantly and through multiple means. From the multiple sensors around the world collecting information about the space around us, to the multitude of social media platforms that allowed anyone connected to the internet to voice their opinions or share photos with the click of a button. The huge amount of information being shared led to the development of tools capable of processing and producing knowledge from this data. Depending on the problem, there are multiple approaches, ranging from Business Intelligence tools such as PowerBI, which allows users to visualize and analyze data themselves in order to make decisions, to machine learning/neural network algorithms, where the job of analyzing and classifying the data is left to the algorithm or model.

In the realm of Earth observation and monitoring, the importance of reliable and precise data has never been more critical. With global challenges such as the huge increase in vessels cruising the waters, some of which may engage in illegal activities, coupled with a high number of geopolitical problems that require active monitoring, the importance of these systems becomes apparent. Climate change introduces additional challenges, such as rising sea levels placing some urban areas in danger. All these challenges created the need to quickly and accurately assess situations at any given time of day and in any location on Earth. This has led to the development of spaceborne imaging systems able to create detailed images of areas of interest, including optical imaging and Synthetic Aperture Radar (SAR), among others, each with unique capabilities and limitations. SAR are active systems that produce grayscale

images, where each pixel value represents the backscatter amplitude of the emitted signal. These can be use for monitoring sea level rise by performing change detection, control forest fire progression or oil spills at sea by searching for pixels that contrast with its surroundings. In times of conflict it can also be used for monitoring incoming and outgoing crafts, from planes as seen in Figure 1.2, to ships as seen in Figure 1.1.

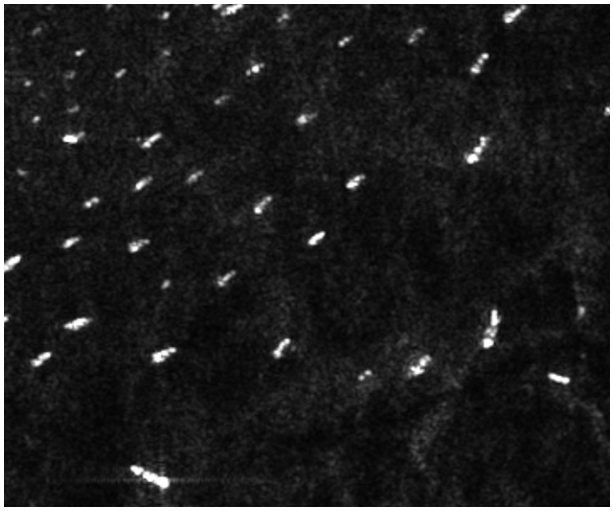

Figure 1.1 SAR image of vessels at sea represented by the white speckles

Figure 1.2 SAR image of a military hanger with the aircrafts and its shadows

One example of this is security control of a given area, like port control, where the operators are observing every vessel's positioning in order to validate their work. However, due to the number of vessels that can be arriving or departing at any given time, it can become a complex task for human operators, even with the aid of Automatic Identification System (AIS) message reports, which are sent by every legal vessel approximately every 3 minutes. A problem arises when the crew is performing illegal activities, such as clearing oil tanks in the ocean, where they will usually turn off their AIS transponder to appear invisible. This creates the need to have an imaging system capable of capturing any desired area at any given time with good resolution.

For this purpose, many types of imaging sensors are sent into space on satellites that rotate over the Earth, capturing images as they travel. Many types of sensing systems, like SAR, optical, thermal, Infrared (IR), and others, have been developed to capture different properties

of the world, each with its own advantages and disadvantages, creating high-quality images that give us a more complete picture of the relevant areas. The Figure 1.3 showcases the situation, where in a compact space the number of vessels at sea is considerable and monitoring all of these vessels is an hard task, specially when bad faith actors turn off their AIS transponder for illegal activities.

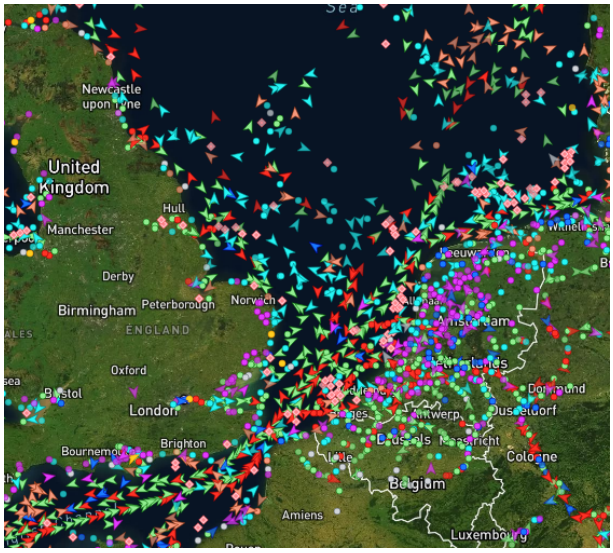

Figure 1.3 Example fetched from MarineTraffic showcasing the number of vessels at sea

To address these challenges, SAR systems have been increasingly used as they are active remote sensing systems that do not rely on ambient light but instead use microwave signals to illuminate their targets. This allows them to penetrate cloud cover and operate effectively both day and night, making them an invaluable tool for continuous monitoring. The images produced by SAR, however, come with their own set of challenges, such as speckle noise and geometric distortions, which can complicate data interpretation. Further, unlike optical systems, SAR images do not represent color information. Instead, they represent the backscattered energy reflected on the surface, which is affected by many variables such as construction materials, surrounding objects, angle of capture, etc. As our eyes rely on color information to better detect and distinguish patterns in the real world, it becomes harder to interpret SAR images. This usually leads to the need for dedicated teams that specialize in SAR images, as they present normal day objects with different geometries and positioning that an everyday person wouldn't recognize easily.

This is in contrast to optical imaging systems, which produce images that are much more intuitive for non-specialists to interpret, as shown in Figure 1.4.

The integration of SAR with optical systems presents a way of combining both systems advantages. By merging the high-resolution, color-rich visual data from optical sensors with the all-weather, day-and-night capabilities of SAR, it is possible to achieve superior monitoring and data collection.

This thesis introduces a novel approach to this integration challenge using a convolutional neural network (Convolutional Neural Network (CNN)) based autoencoder. This method

Figure 1.4 Satellite optical image of ship port

leverages the power of deep learning to extract relevant information from SAR images and reconstruct their optical counterparts. This process aims to simplify the data interpretation process, making it accessible to non-specialized personnel. By enhancing pattern recognition and reducing the need for expert analysis, this approach can significantly broaden the applicability and efficiency of remote sensing systems in various domains, including land monitoring, environmental conservation, and security applications. This work provides a robust tool for the enhanced interpretation of complex imaging data and facilitating better-informed decisions in critical areas.

Chapter 2

Components

2.1 Optical remote sensing Systems

Optical remote sensing imaging systems are sophisticated technologies that take advantage of the properties of light to capture and analyze images from the environment. Spaceborne optical systems provide the ability to capture high-resolution, color-rich images of any given site at any given time, making them highly valuable for situations where ground visualization is impossible [2].

These remote sensing imaging systems function by capturing Electromagnetic Radiation (EMR) in the visible spectrum. These signals are emitted by an illumination source, usually the sun, and propagate through the atmosphere eventually colliding with ground objects. When the light wave, which can be seen as either a wave or a compositions of multiple photons, collides with the objects a portion of it will be reflected. These optical systems capture this reflected light from which they can then convert the backscattered values into a quantized digital image. The frequency of the wave will dictate the band from which the image will be presented, where red will represent the backscattered values with frequency equals to around 440THz. Green encompasses frequencies around 560THz and blue around 640THz.By quantizing each band to a value from 0 to 255 it is then possible to create a computer redable image [3].

The light, a form of EMR, interacts with objects and either gets absorbed, reflected, or transmitted [3]. Optical systems are designed to capture this reflected or transmitted light. The core of these systems is their ability to process visible and near-visible wavelengths, which includes not only the visible spectrum but also the near-infrared and shortwave infrared bands. Figure 2.1 showcases how a satellite optical system captures the radiation, by using the sun as an illuminator of opportunity.

Upon interaction, EMR suffers changes in magnitude, direction, wavelength, polarization, and phase. These changes are then detected by the remote sensor, enabling the interpreter to obtain useful information about the object of interest [3, 4]. The image data contain both spatial information (size, shape, and orientation) and spectral information (tone, color, and spectral signature). The sun's incident electromagnetic radiation can be described by Equation 2.1.

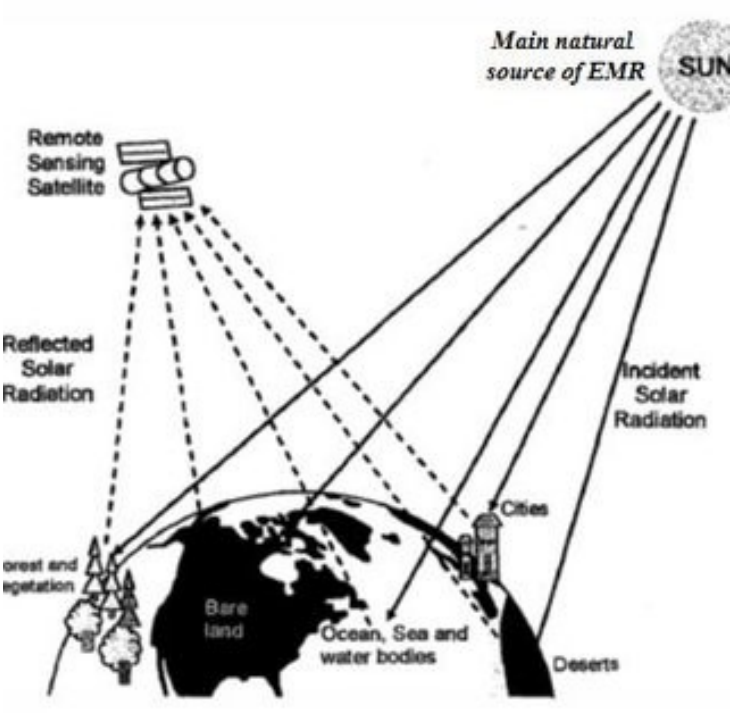

Figure 2.1 Optical remote sensing systems Optical remote sensing systems

$$EI(\lambda) = ER(\lambda) + EA(\lambda) + ET(\lambda) \tag{2.1}$$

Where EI is the sun's Incident energy, ER is the Reflected energy, EA is the absorbed energy and ET is the Transmitted energy [3, 4].

EMR can be understood through two foundational theories: the wave theory and the particle theory [3, 4].The wave theory defines that electromagnetic energy propagates as a wave traveling at the speed of light (denoted as c), which is approximately 3×10^8 meters per second. Each wave is characterized by its wavelength (λ), defined as the distance between successive peaks, and its frequency (ν), which measures the number of peaks that pass a given point per unit of time, as illustrated in Figure 2.2. The relationship between the speed of light, wavelength, and frequency is described by Equation 2.2.

$$c = \nu \times \lambda \tag{2.2}$$

Given that the speed of light is constant, wavelength and frequency are inversely related: as the frequency increases, the wavelength decreases. In remote sensing, electromagnetic energy is categorized based on its position within the electromagnetic spectrum, typically defined by its wavelength. At one end of this spectrum, there are shorter wavelengths like X-rays, while at the other end, there are longer wavelengths used for various communication purposes, including television and radio signals.

The interaction of electromagnetic energy with Earth's atmosphere significantly impacts its transmission and reception. Two primary atmospheric effects, scattering and absorption, modify the incoming solar radiation [3, 5, 2, 4]. Scattering is a process that occurs when radiation is

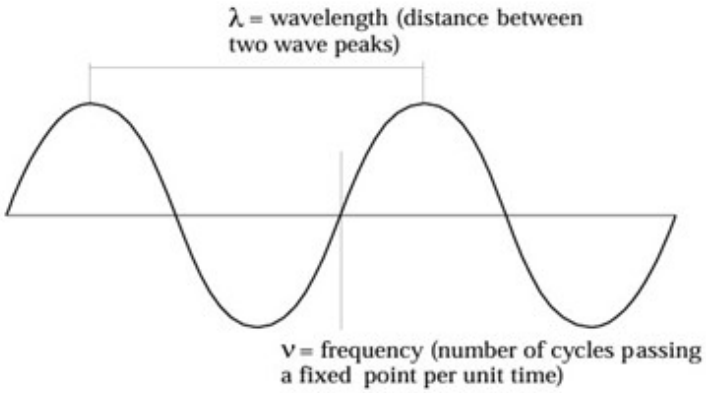

Figure 2.2 Wavelength and frequency

deflected by atmospheric particles, and depends heavily on the size of these particles relative to the wavelength of the incoming energy. Rayleigh scattering, a specific type of scattering, happens when the wavelengths are larger than the atmospheric particles. This scattering is more intense for shorter wavelengths, which explains why the sky appears blue, as shorter wavelengths scatter across the sky more uniformly than longer wavelengths, making blue the most visible color. Mie scattering happens when particles are about the same size as the wavelength of the light and affects all wavelengths more equally than Rayleigh scattering. Rain can cause this type of scattering, however it is usually more impactful during harsher conditions such as dust storms, forest fires, or other events that caused the atmospheric aerosol load to increase [4].

Absorption refers to the process by which the atmosphere particles take some of the energy from the incoming radiation, effectively reducing the intensity of the radiation as it travels through the atmosphere [5, 2, 4]. In remote sensing, atmospheric absorption can weaken the signals enough that it might require correction algorithms to accurately interpret the data.

A key benefit of optical satellite systems is their flexibility. By being mountable on aircraft, they enable the capture of imagery from any location at any given moment. This is crucial for real-time applications and time-sensitive analyses, where quick decision-making requires up-to-date information, making them especially valuable for applications that require accurate visual assessments, such as environmental monitoring, urban planning, and disaster response.

These also offer color rich images, meaning that since optical systems focus on the visible spectrum the images provided are better representations of the real world, when compared to other systems like thermal, or radar imagery.

However, when considering imaging systems, whether they are optical or radars it is important to consider the spatial resolution, as this will affect the maximum size that a target of interest can have without being distorted into the background noise. If an object is smaller than the minimum special resolution, it's representation will be grouped with other elements and might become hidden. This describes how detailed a satellite's imagery can be. It can be

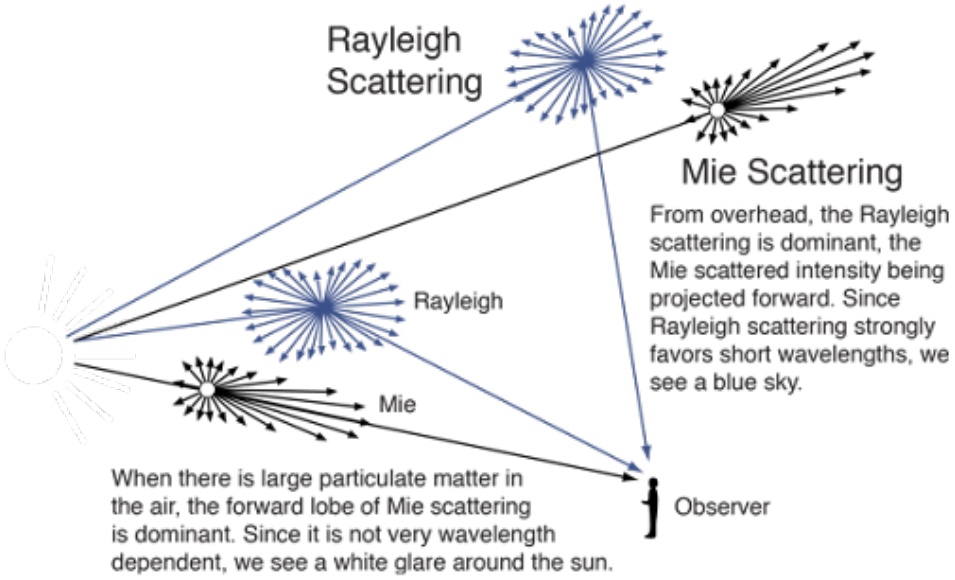

Figure 2.3 Rayleigh scattering

measured as the pixel capability, which refers to the smallest possible ground area that can be displayed in an image. For example, a 10m2/pixel satellite can capture an area of 10 by 10 meters in each pixel, which means that all objects with that area will be displayed in one single pixel.

The chosen ground resolution can affect the performance of Machine Learning (ML) and Neural Network (NN) models used for analyzing satellite imagery. These models often rely on clear and accurate representations of features to detect patterns, classify objects, or identify specific targets. If the spatial resolution is too low, objects of interest may be incorrectly represented or entirely missing, leading to inaccurate or incomplete analysis.

Higher spatial resolution can offer finer details and a clearer distinction between closely situated features. However, it also requires more data storage, processing power, and transmission bandwidth, which are also important considerations in the design and application of satellite imaging systems.

Figure 2.4 Optical Satellite image

2.1.1 Key Components of Optical Systems

- Optical Lens System: The lens system includes one or more lenses that focus incoming light onto the sensor. Lenses can vary, from fixed-focus lenses to complex zoom lenses with variable focal lengths. The quality of the lens affects attributes like depth of field, magnification, and resolution of the final image.
- Sensor: The sensor captures incoming photons and converts them into electrical signals. These sensors differ in their sensitivity, noise levels, and how quickly they can capture images.
- Filters: Optical filters control which wavelengths of light are allowed to reach the sensor. These can be used to enhance contrast, reduce glare, or target specific wavelengths for applications such as vegetation analysis in remote sensing.
- Image Processor: The image processor is a digital system that converts raw sensor data into a usable image. This involves correcting for various distortions, enhancing the image for better visualization, and compressing the image for storage and transmission.

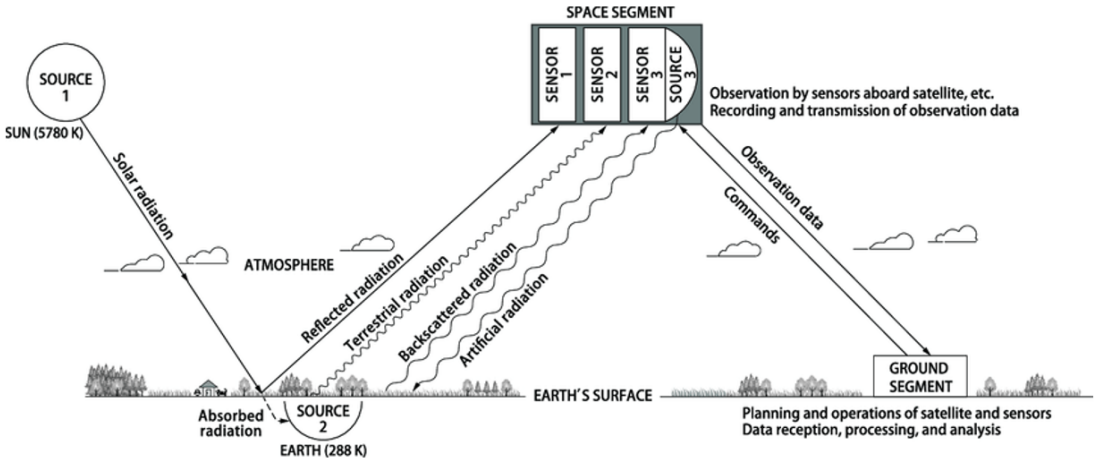

Figure 2.5 Image acquisition cycle

After creating the image based on captured photons, the finished product is an image, that can be analyzed, and information can be extracted. However, these images can suffer from distortions and noise that can decrease the quality of the image. Some post-processing techniques can be applied to mitigate this issue, such as:

- Denoising: Algorithms such as wavelet transforms are used to remove noise while preserving important details.
- Contrast Enhancement: Techniques like histogram equalization or adaptive histogram equalization are applied to improve image contrast, making details more visible.

- Geometric Correction: Corrective transformations are applied to address issues like optical distortion and perspective anomalies, especially in aerial imaging and satellite imagery.

Despite their advantages, optical systems are limited by their dependence on ambient light conditions. Poor lighting, whether due to time of day or weather conditions like fog and clouds, can severely diminish the quality of the captured images. Advances in technology are overcoming these limitations through the integration of optical systems with other types of imaging technologies, such as IR or radar and near visible, as these systems are able to acquire photos independently of weather and lighting conditions due to the frequency that they operate on and due to being active. However these are costlier and increase the complexity of the system and so might not be feasible to employ. To improve cloud coverage correction in satellite imagery, both more traditional and modern techniques have been employed. Traditional methods like adaptative thresholding and connected components algorithms were used for detecting and segmenting cloud-covered areas, as seen in Figure 2.6. These techniques focus on identifying and isolating clouds by evaluating pixel intensity and spatial continuity. On the other hand, advanced generative models, are now employed to reconstruct missing data caused by cloud obstructions, effectively generating the hidden optical details by learning patterns from cloud-free images.

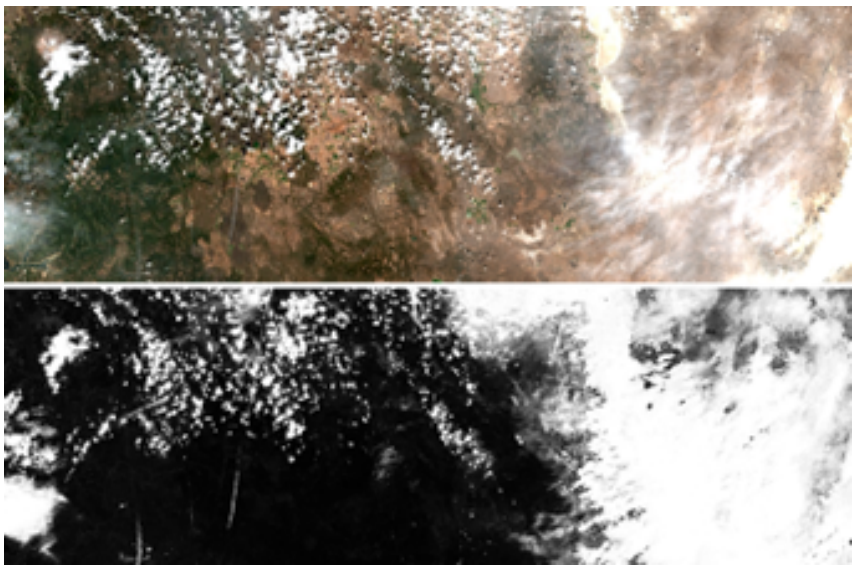

Figure 2.6 Sentinel-2 images with cloud cover above and cloud mask below

2.1.2 Synthethic Aperture Radar

In 1951, Carl A. Wiley developed the first SAR [6]. These radar systems take advantage of the fact that they are mounted on a moving platform to simulate a longer antenna, resulting in better resolutions with fewer resources.

SAR is an active remote sensing technique that uses microwave frequencies to obtain high-resolution images of the Earth's surface by emitting electromagnetic pulses and receiving

their echoes. Based on the delay between the sent signal and the backscattered signals, it determines the distance (range) between the sensor and the surface objects. The forward motion of the platform enables the transmission and reception of signals from the same object from successive sensor positions.

By coherently combining the backscattered signals from the object, it enables the construction of a synthetic aperture [6]. The synthetic aperture is equal to the length of the orbit where the object is detected. The longer the SAR antenna length, the higher the spatial resolution (assuming the signal is kept the same). Yet, due to physical constraints, it might not be feasible to mount a bigger antenna, and so by simulating this longer synthetic aperture, it allows us to achieve a higher spatial resolution using the same resources.

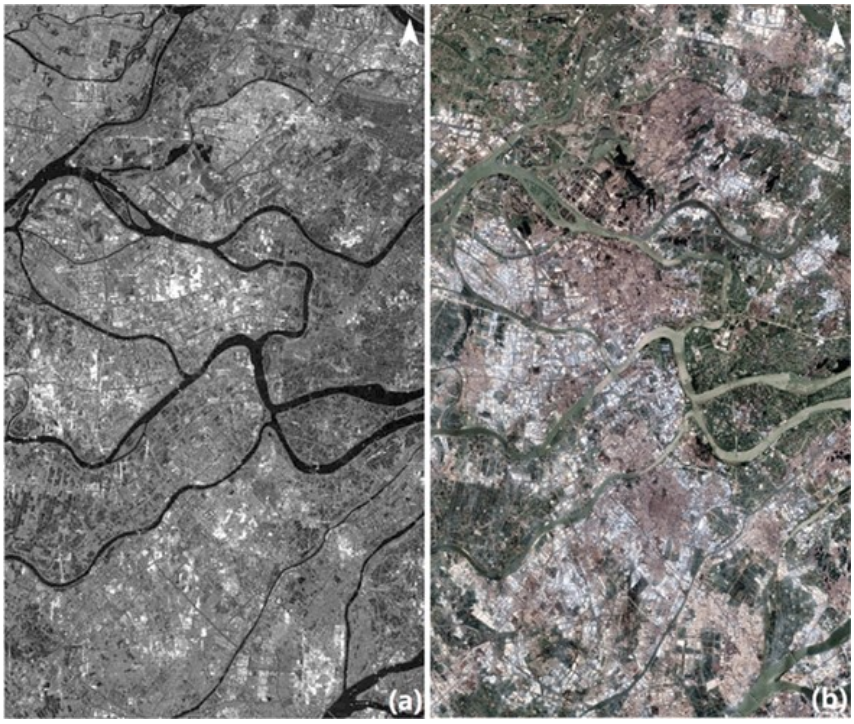

Figure 2.7 SAR and optical images for comparison

Until the development of SAR systems, the commonly used radars for object detection were the Side Looking Airborne Radar (SLAR) systems. These systems emerged during WWII [6] and were used for navigation and reconnaissance, providing a side-looking view of the terrain relative to the flight path of the aircraft. Unlike SAR, these systems don't take advantage of a moving platform to increase the spatial resolution, they just transmit the signal perpendicular to the aircraft's flight path. As a result, SLAR systems disregard the phase information of the received backscatter, and thus, they were not capable of coherently combining the signals, making it impossible to create a synthetic aperture, leading to lower qaulity images as seen in Figure 2.8.

By maintaining the phase information of the returned signal, SAR can effectively integrate these signals over a longer synthetic aperture. This process involves complex signal processing techniques that align and sum the phase and amplitude of the returns from multiple pulses.

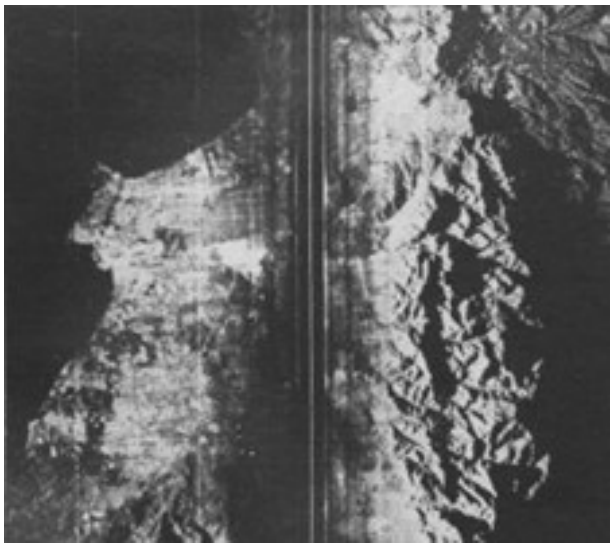

Figure 2.8 Side-looking radar imagery of Los Angeles area around 1965

This coherent integration enables SAR to synthesize an antenna that is much longer than the physical antenna, improving the azimuth resolution.

For SLAR systems, the azimuth resolution can be obtained through Equation 2.3:

$$\delta a = \frac{\lambda}{d_a} r_0 \tag{2.3}$$

Where δa is the azimuth resolution, λ is the wavelength, d_a is the antenna length, and r_0 is the distance. The equation above shows that the azimuth resolution is dependent on the range distance, as with the increase of altitude, the azimuthal resolution decreases. Using a SAR system, the azimuth resolution is improved and is computed by Equation 2.4:

$$\delta a = \frac{d_a}{2} \tag{2.4}$$

While for SLAR the resolution degrades with distance r_0, for SAR the resolution is kept constant. Considering a 3-meter antenna (d_a) and a wavelength of 0.03m (λ) targeting an object 5 km away (r_0), using SLAR it is possible to obtain a resolution of 50m [6]. With SAR, using the same antenna, it is possible to increase the azimuth resolution to 1.5m, which is 30 times better [6]. SAR system resolution is independent of flight height, meaning it can be mounted on a satellite and maintain the same resolution as one mounted on an aircraft, therefore increasing deployment options. Figure 2.9 showcases the differences in resolutions that both systems offered, where it is possible to observe the advancements in resolution and lower noise interference.

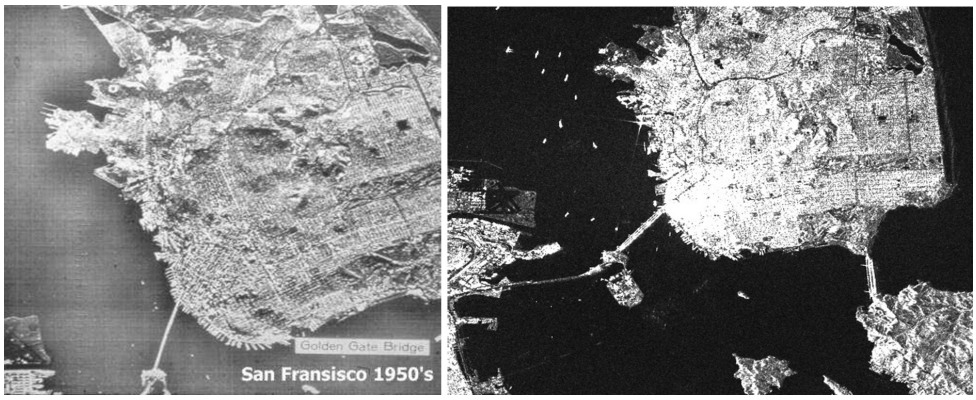

Figure 2.9 SLAR image of San Francisco around 1965) and a current SAR image also from San Francisco)

SAR sensors commonly utilize frequency modulated pulsed waveforms for transmission, known as chirp signals [7]. These signals vary in frequency, either increasing (up-chirp) or decreasing (down-chirp) with time.

The chirp system's key feature is its reversibility, allowing a chirp signal to be converted back into an impulse using an antichirp system. This requires the antichirp system to have a magnitude of one and the opposite phase of the chirp system. Short pulses are used for high-resolution distance measurements. However, generating short pulses typically requires high power, which can be problematic for the power handling capabilities of the radar system. This allows the portions of the system that measure distance to see short pulses, while the power handling circuits see long-duration signals [7].

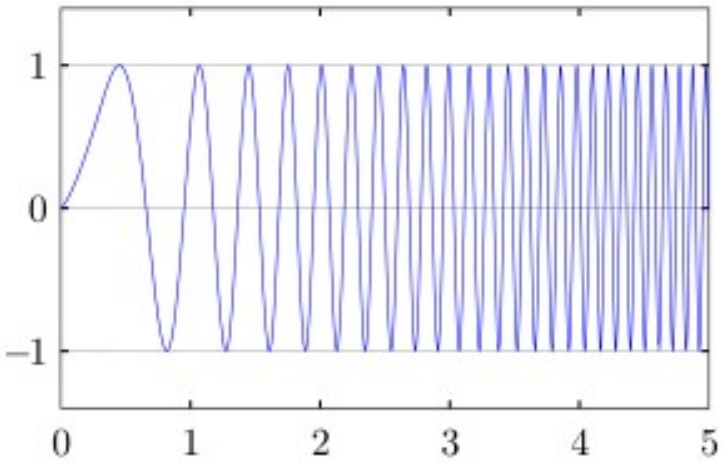

Figure 2.10 Chirp signal frequency

As seen in Figure 2.10, the amplitude of the transmitted waveform is constant during the pulse time x, while the frequency of the signal varies exponentially as a function of time according to Equation 2.5:

$$f(t) = f_0 k^{\frac{t}{T}} \tag{2.5}$$

Where f_0 is the starting frequency (at $t = 0$), and k is the rate of exponential change in frequency, which is defined by Equation 2.6:

$$k = \frac{f_1}{f_0} \tag{2.6}$$

Where f_1 is the ending frequency of the chirp (at $t = T$).

After transmitting the signal, the radar "listens" to the scattered echoes and stores the received signals on-board. When referring to the time in the range direction, it is often denoted as fast time, which is an allusion to the velocity of the electromagnetic waves traveling at the speed of light. The transmission and listening procedure is repeated every Pulse Repetition Interval (PRI) seconds, where the PRI is the reciprocal of the pulse repetition frequency (Pulse Repetition Frequency (PRF)), as shown in Equation 2.7 [7].

$$\text{PRI} = \frac{1}{\text{PRF}} \tag{2.7}$$

At any time t, the distance between the radar, moving at constant velocity v, and a point on the ground, described by its coordinates $(x, y, z) = (x_0, 0, \Delta h)$, is easily obtained by applying Pythagoras' theorem, as shown in Equation 2.8:

$$r(t) = \sqrt{r_0^2 + (vt)^2} \tag{2.8}$$

The received echo signal data form a two-dimensional data matrix of complex samples, where each complex sample is given by its real and imaginary part, thus representing an amplitude and phase value. The first dimension corresponds to the range direction (or fast time). The radar acquires a range line whenever it travels a certain distance, forming the second dimension of the data matrix, known as azimuth or slow time. The return echoes from the illuminated scene are sampled both in fast time (range) and slow time (azimuth), as illustrated in Figure 2.11.

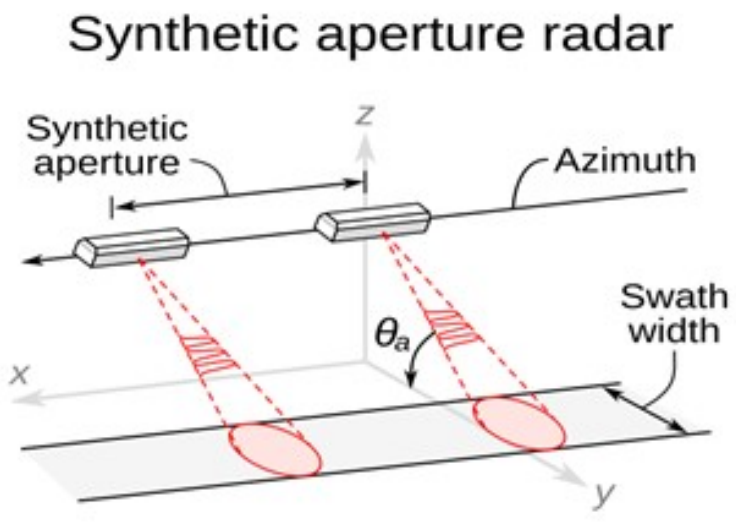

Figure 2.11 Synthethic aperture radar diagram

SAR imagery captures the phase and amplitude of backscattered signals, which can be influenced by antenna properties such as wavelength, polarization, viewing geometry, and transmitted power, as well as by the object it reflects from, based on surface roughness, shape, terrain, humidity, and other factors.

The most common frequencies used by radar are listed in Table 2.1:

SAR Band	Frequency range	Wavelength range	Application examples
X	8–12 GHz	2.5–4 cm	Snow monitoring, urban monitoring
C	4–8 GHz	4–8 cm	Change detection, ice monitoring, vegetation monitoring
S	2–4 GHz	8–15 cm	Vegetation monitoring, ice monitoring, subsidence monitoring
L	1–2 GHz	15–30 cm	Canopy penetration, subsurface imaging, biomass estimation

Table 2.1 SAR Bands and Their Applications

As the frequency goes up, the system manages to obtain finer details in images as the spatial resolution increases. However, higher frequencies are more affected by losses and noise and lose the capability of penetrating vegetation and soil. Due to this, the most commonly used of the four bands is the C band [6], as it offers a good balance between resolution and penetration capabilities. SAR systems provide both 2D and 3D mapping, as well as 4D mapping (space and time), capable of achieving resolutions from a few millimeters to hundreds of meters. This allows us to obtain SAR images over a given area, from which it is possible to extract knowledge, such as controlling crop production or monitoring flood areas.

SAR systems can be active imaging methods if the illuminator of opportunity is the radar itself, or passive if they use external sources as illuminators, like cell towers. SAR is useful for land monitoring and scientific research of both land and space, including oceans. Additionally, it is useful for security applications due to its diverse range of deployment options, from airborne platforms to drones and satellites. One of the biggest advantages of SAR is that, due to the frequencies used, SAR systems are less affected by weather and light conditions, meaning it can still obtain meaningful information even with significant cloud coverage or at night as seen in Figure 2.12.

Inherently, due to the movement of the platform of the radar, the emitted signal will shift in frequency when it is reflected back. This is called the doppler effect and is caused when either the emitting source or the object on which the signal reflects are moving [8]. As the platform travels it is consecutively sending signals that as it gets closer to an object the interval between the sent and received signals is shortened, shifting its frequency up and decreases it as it gets further away [8, 9].

This effect helps the SAR system better distinguish objects on the ground in the azimuth direction as the doppler effect arises because different points on the ground have different relative velocities with respect to the moving radar sensor, leading to different Doppler shifts

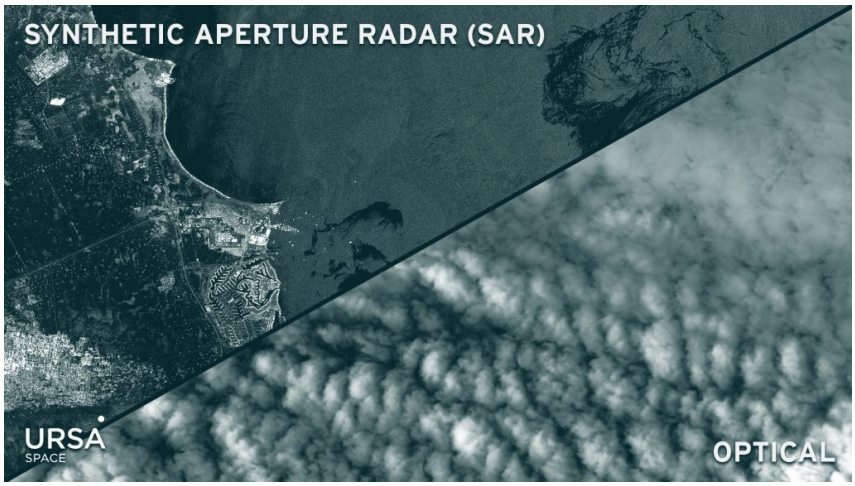

Figure 2.12 SAR vs Optical in cloudy weather

[8, 9] as seen in Figure 2.13.

These shifts allow the SAR system to distinguish between targets that are side by side in the azimuth direction, however, require more processing to compensate for the doppler shifts. By analyzing the frequency shifts (Doppler shifts) associated with each return signal over the synthetic aperture, the system can achieve finer resolution, which helps resolve ambiguities that might arise in the azimuth dimension [8, 9].

In SAR imaging, as the platform moves it induces a Doppler shift in the radar signal reflected backscatter from a stationary target. This can be predicted and compensated by algorithms [10], however, when the target itself is moving, especially at a significant speed, relative to the SAR sensor, standard Doppler compensation techniques, which assume the target is stationary, may not accurately account for the additional Doppler shift caused by the target's own motion. This effect manly manifests itself in the azimuth direction as this direction is obtained through the Doppler shifts of the received signal, while the range direction is obtained through the time delay of the radar signal's return.

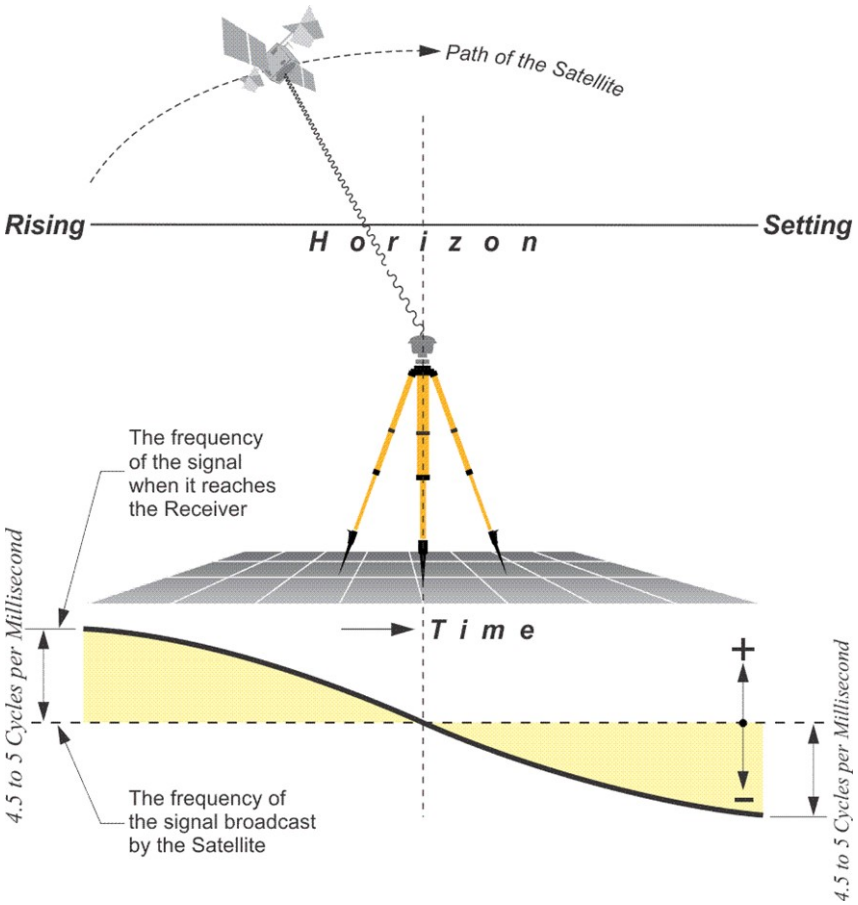

Figure 2.13 Doppler shift in SAR systems

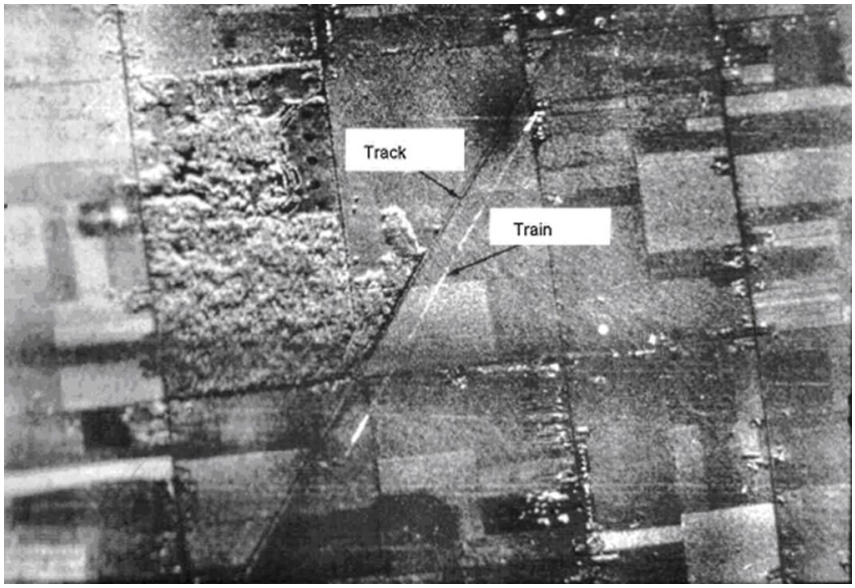

Figure 2.14 SAR misalignment issue due to train velocity

This misalignment results in the target's position appearing shifted in the final image as seen in Figure 2.14, where the train is misplaced to the side, appearing as if it is traveling on the field and in Figure 2.15, where the vessel's wakes are misaligned with the vessel position. When creating automated systems, it is needed to take into account how this effect can degrade

the performance as it might induce false positives and negatives.

A simple example of this is a vessel that is traveling through a river that when observed in a SAR image can appear on top of the land as when doppler shift was compensated it assumed it was a stationary object and incorrectly positioned it, which can lead to incorrect or missing classification of the vessel. Figure Figure 2.14 and Figure 2.15 showcase this issue where, due to the velocity of the ground objects when the doppler shift is compensated it induces errors in the positioning of the objects. This however can also provide information about the object as the speed can be inferred by calculating the difference between the SAR image position and the ground truth position.

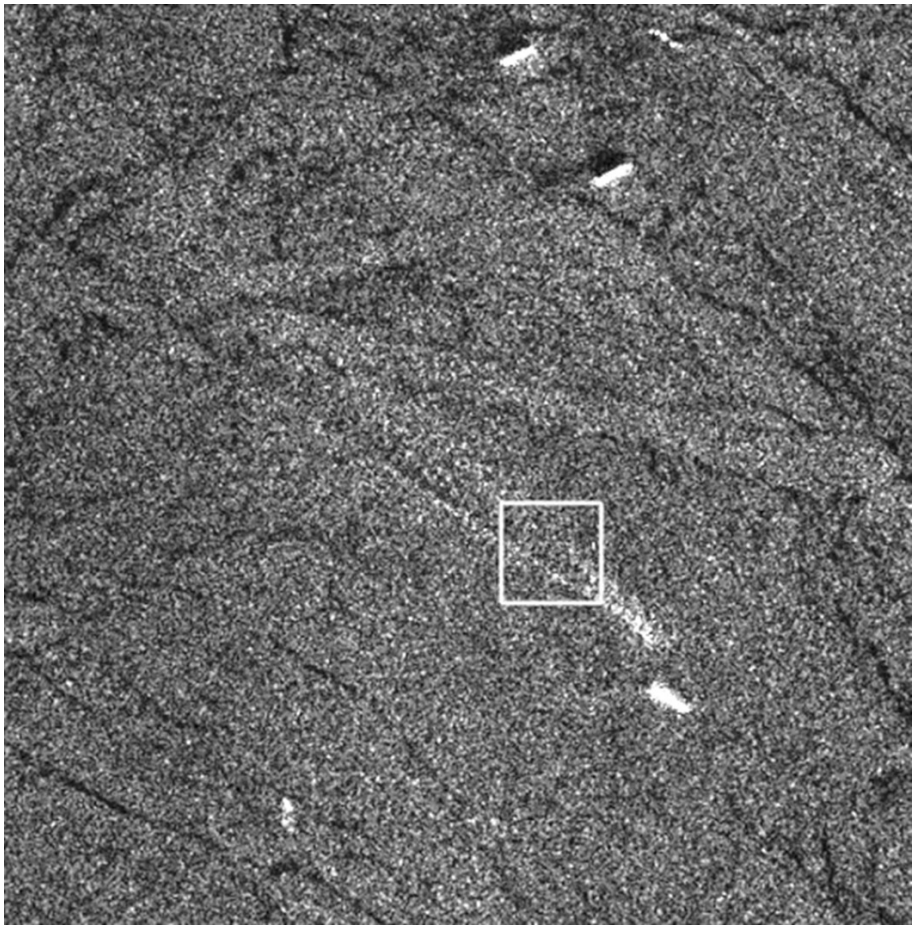

Figure 2.15 SAR misalignment issue due to vessel velocity, observed by the displacement between the wakes and the vessel

Geometric distortions are also an inherent challenge in SAR imaging, largely due to the side-looking nature of SAR systems, that unlike optical sensors that generally look straight down. This diagonal scanning of the ground by radar beams leads to distortions in how ground features are projected in the radar image, including foreshortening, layover, and shadow [11, 12].

Foreshortening occurs when a slope or tall structure faces towards the radar, causing the radar beam to reach the base of the slope (point A) before the top (point B) [11, 12]. As the radar measures distance in slant-range, the radar beam hits and reflects off the base of

the slope before reaching the top. This makes the slope or structure appear compressed and taller than it actually is. The radar signal reflects off point A and returns to the radar before it does from point B, leading to an image where the slope (from A to B) appears shortened and compressed as if it is leaning towards the sensor (A' to B'). Depending on the angle of the hillside or mountain slope relative to the incidence angle of the radar beam, the severity of foreshortening varies.

Maximum foreshortening happens when the radar beam is perpendicular to the slope (from C to D). In such cases, the length of the slope is represented incorrectly and reduced to an effective length of zero in slant-range, making the slope appear extremely compressed (from C' to D'). Foreshortened slopes manifest as bright features in the image, particularly affecting the imaging of steep mountainous terrain. This phenomenon is illustrated in Figure 2.16.

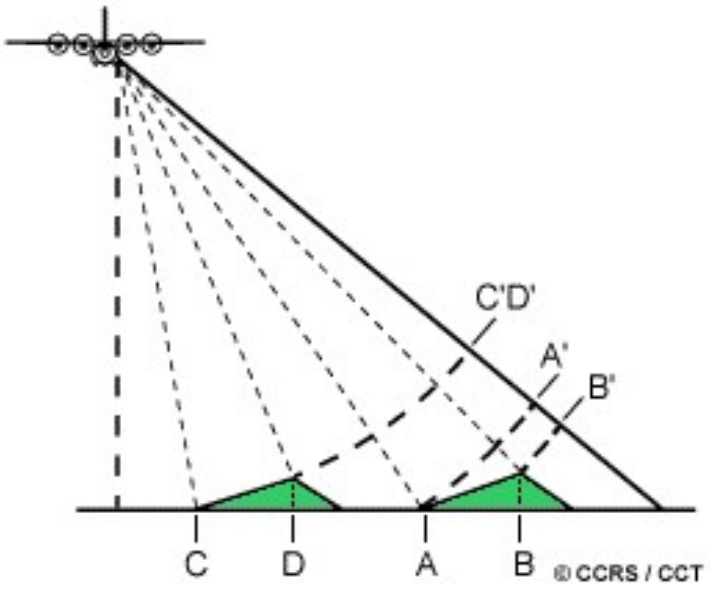

Figure 2.16 Example of radar foreshortening

Layover is another type of geometric distortion encountered in SAR imaging, primarily affecting tall structures or steep terrain. Layover occurs when the radar beam reaches the top of a tall feature before it reaches its base. This happens because the top of the feature is physically closer to the radar when viewed obliquely [11, 12]. The radar signal that reflects off the top of the feature (point B) returns to the sensor before the signal reflecting from the base (point A).

In the resulting SAR image, the top of the feature is displaced towards the radar from its actual position on the ground. This displacement causes the upper portion of the feature to appear as if it is 'laying over' the base. In the image, this is represented by the top (B') appearing closer to the radar than the base (A'), sometimes overlapping or obscuring areas in front of it and is depicted in Figure 2.17.

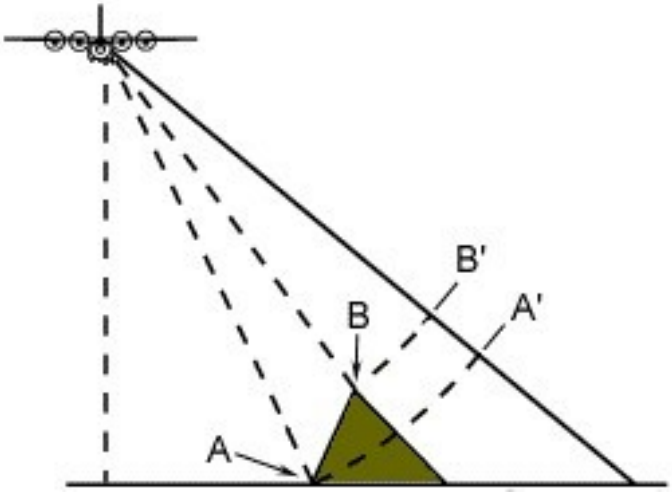

Figure 2.17 Example of radar layover)

Both foreshortening and layover result in radar shadow. Radar shadow occurs when the radar beam is unable to illuminate the ground surface, typically in the down-range dimension or towards the far range, behind vertical features or steep slopes. In areas where shadows occur, no energy is available to be backscattered, resulting in dark areas on the radar image. As the incidence angle increases from near to far range, shadow effects become more pronounced, as shown in Figure 2.18 [13, 12].

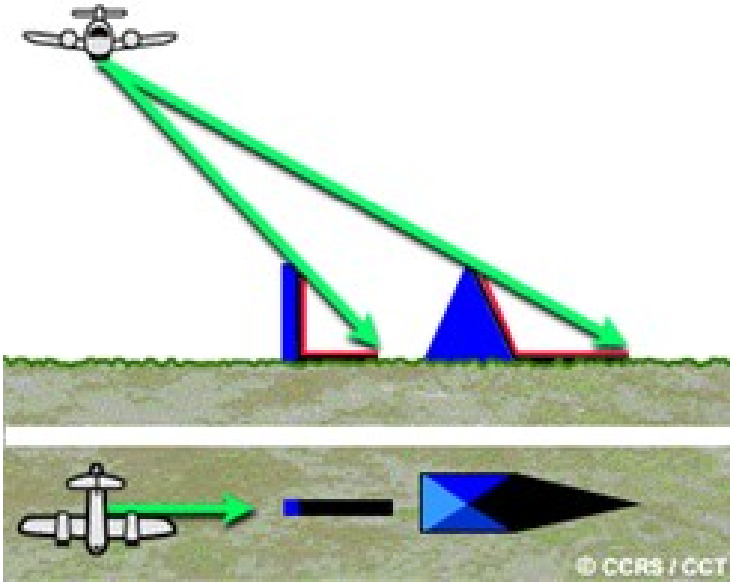

Figure 2.18 Example of radar shadow)

During the process of radiometric corrections of radar images, areas affected by foreshortening can be corrected using digital elevation models [11]. However, for areas impacted by layover or shadow, there are no actual data returns to correct, because these regions do not receive any radar illumination. Consequently, in the corrected images, pixels in these shadowed or layover regions typically have a value of 'No Data', as shown in Figure 2.19.

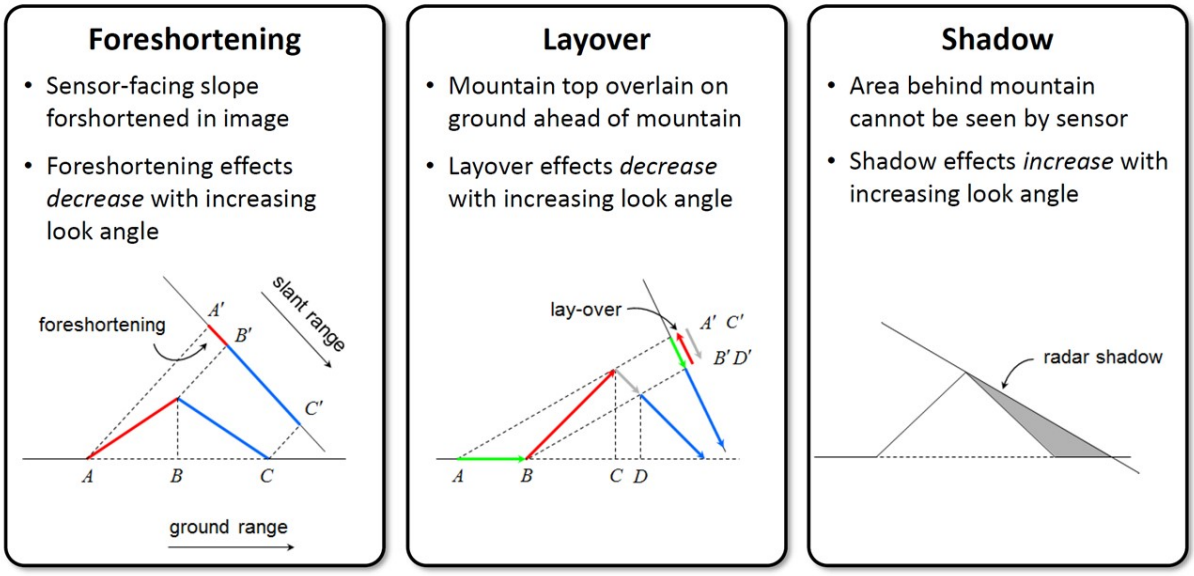

Figure 2.19 Radar geometry distortions comparison)

SAR image analysis by humans is a complex task due to geometric distortions, speckle noise, and most importantly, the absence of color information. As a result, the interpretation of SAR images requires a specialized human operator, turning tasks like ship detection into a slow and difficult process, which can be crucial for time-sensitive missions. To overcome this limitation, SAR systems are often combined with other sensors, such as IR or optical sensors, to provide a more comprehensive overview of the region of interest.

However, with the recent expansion of applications and advancements in NN and ML technologies, new approaches aim to lower the skill floor required for human interpretation of SAR images. Deep learning techniques can be utilized to automate and simplify the image analysis process, providing more intuitive and accessible solutions for non-expert users.

2.1.3 Neural Networks

The topic of NN is not new. In 1943 the concept of the perceptron was created by McCulloch and Pitts [14]. The perceptron is based on the human's neurons where there are electrical impulses that stimulate neurons and based on the combined strength of these impulses the neuron will either activate, propagating the response to others, or if it is not strong enough, it will not propagate anything. Instead of electrical signals, neural networks receive two or more input values, then multiply them by a weight that can change as the network adapts. Based on an activation function or loss function, it will either activate or not. By comparing the obtained result with the expected result, the network can learn how to better optimize its learnable values so that it diminishes the loss function.

A perceptron serves as a foundational component within NN architectures, contributing significantly to information processing and decision-making (as shown in Figure 2.20). Comprised of essential elements, the perceptron's inputs $(\chi_1, \chi_2, \ldots, \chi_n)$ convey external stimuli or features, each associated with a weight $(\omega_1, \omega_2, \ldots, \omega_n)$ indicating its significance. These

weights, that are learnt through training, optimize the perceptron's performance.

The bias term (ω_1), an independent parameter, introduces flexibility by allowing the perceptron to shift its decision boundary. The activation function is a non-linear operation applied to the weighted sum of inputs and bias. This introduces non-linearity, enabling the model to learn intricate relationships in the data.

The perceptron's output (y), determined by the activation function, signifies its decision, meaning the probability of belonging to a particular class. Training involves adjusting weights and bias through backpropagation [15], where the perceptron minimizes the difference between its output and the desired output.

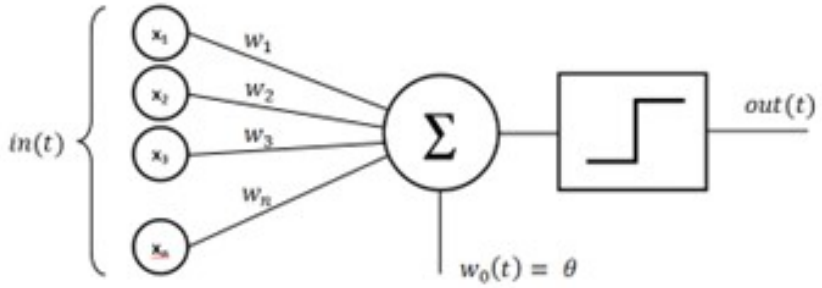

Figure 2.20 Perceptron diagram

The backpropagation algorithm revolutionized NN learning by iteratively adjusting weights based on prediction errors [15]. It propagates the difference between predicted and actual values backward through network layers, updating weights via gradient descent. This iterative process enhances the network's ability to learn and adapt, enabling it to capture complex patterns in data and significantly advances Artificial Intelligence (AI) and ML applications.

It was only in the early 90's that the use of backpropagation was popularized by Yann LeCun [16]. These networks learn by comparing the predicted output with the expected output to calculate an error value, and its error is then back propagated throughout the layers of the networks, allowing each to learn iteretavely. This error is calculated based on the chosen loss function. The Mean Squared Error (MSE) is one of the most common loss functions and calculates the square difference between the real and expected output, as shown in Equation 4.7.

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{2.9}$$

Where y_i represents the true value and $\hat{y}_i$ the predicted value. To update the network's weights, it needs to calculate the gradient of the loss values. This tells the network that for a given set of weights what are the directions that when shifting the weight values, the error is minimized. Considering a single weight w, it is possible to calculate the gradient using Equation 2.10.

$$\frac{\partial L}{\partial w} = \frac{\partial L}{\partial y} \times \frac{\partial y}{\partial w} \tag{2.10}$$

Where $\frac{\partial L}{\partial w}$ represents the loss with respect to the weight, $\frac{\partial L}{\partial y}$ is the derivative of the loss function with respect to the output y, and $\frac{\partial y}{\partial w}$ is the derivative of the output y with respect to the weights W. Once the model calculates its derivative for its weight it is possible to obtain the gradient, which is the combination of the derivatives. With this the model can use the gradients to adjust the parameters of the network. The basic idea behind gradient descent is to update each parameter in the direction that reduces the loss, as shown in Equation 2.11.

$$w_{\text{new}} = w_{\text{old}} - \alpha \times \frac{\partial L}{\partial w} \tag{2.11}$$

Where w_{new} are the new weight values after the update, w_{old} are the weights before the update, α is the learning rate (a hyperparameter that determines the step size), and $\frac{\partial L}{\partial w}$ is the gradient of the loss with respect to the weight w. These steps are done for all the weights in all the layers, starting at the last layer, and the resulting gradients and losses are propagated throughout the network.

A significant milestone in the evolution of NN was marked by the introduction of CNN by Kunihiko Fukushima [13]. CNN offered a new shift in image processing and feature extraction. Until this the proposed state of the art would involve a fully connected deep NN, where each pixel would be independently fed to the network as an input and connect to every neuron in the first layer, where each weight associated to a pixel would determine the importance of the pixel to obtain the desired output. By adjusting the weights the network would be able to prioritize certain patterns in the image from which could label them. This however was not practical as both the images and networks grew in size. Assuming a single neuron, and using an image of 100x100 then the network would have to calculate the weights for 10 000 values.

The core principle of CNN involves convoluting a sliding window kernel across an input image, yielding smaller convoluted images known as feature maps. These feature maps serve as representations of distinct patterns that the network focuses on, capturing important details for accurate classification or image reconstruction (see Figures 2.21 and 2.22).

While these feature maps may appear abstract to human observers, they contain vital information that enables correct classification in the case of outputting labels or reconstruction in the case of image generation, where each pixel can be considered as a label. The training process involves comparing the ground truth image with the recreated image. Subsequently, the network adjusts its kernel values through a learning process, minimizing the loss and therefore enhancing accuracy.

The strength of CNN lies in the stacking of these convolutional layers. By combining multiple layers, each with its distinct kernels, the network can extract hierarchical and meaningful information from an image. These layers, when chained together, create a hierarchical representation of features, enabling the network to distinguish complex patterns at different levels of abstraction (as illustrated in Figures 2.21 and 2.22).

The ability to automatically learn and adapt the convolutional filters during training allows CNN to excel in tasks such as image recognition, object detection, and image generation. This hierarchical feature extraction, facilitated by the convolutional operation and subsequent

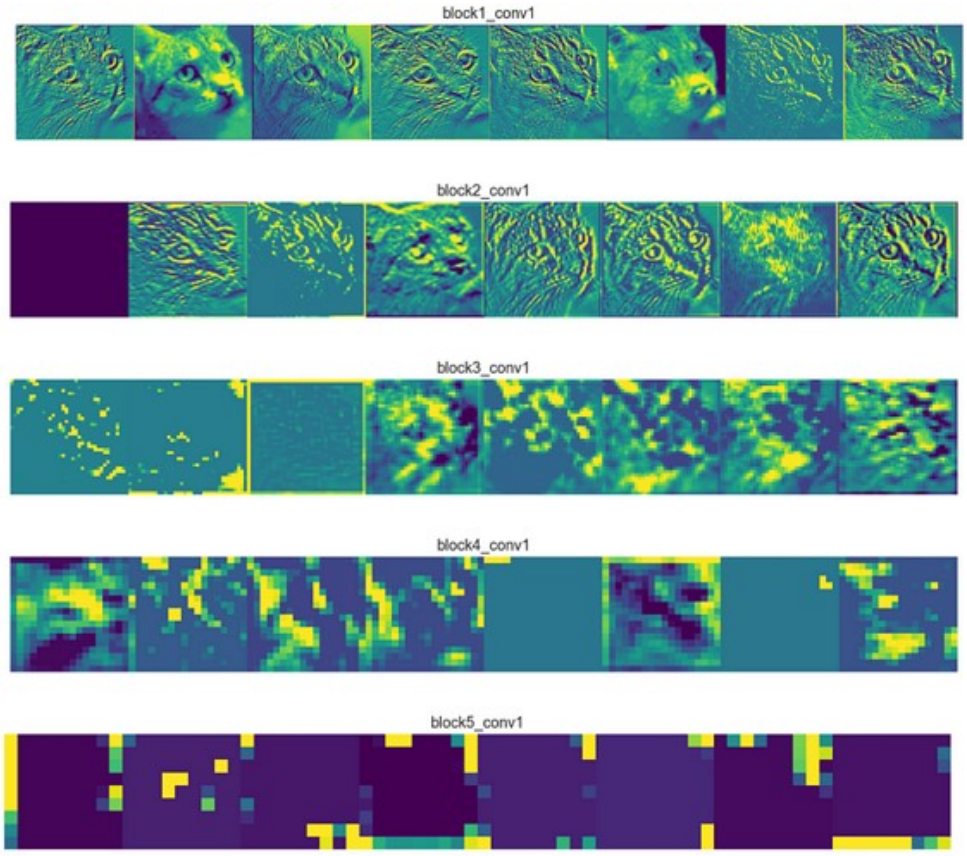

Figure 2.21 Cat feature map as the image passes through the layers

layer stacking, has increased the effectiveness of NN in understanding and interpreting visual information.

With this in mind, this thesis proposes the SAR2optical system that takes advantage of both SAR and CNN by creating a convolutional neural network-based model that is trained to convert from the SAR image domain to the optical image domain, as illustrated in Figure 2.23.

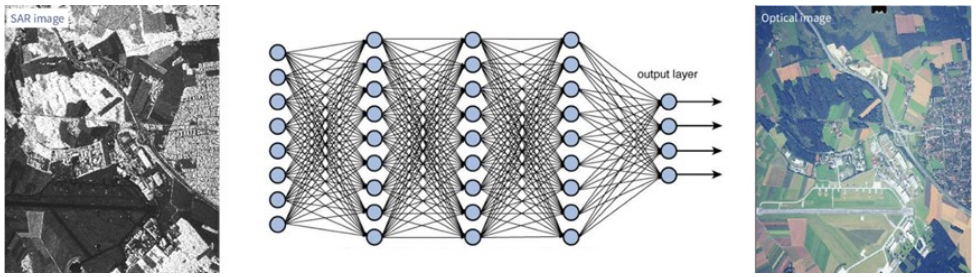

Figure 2.23 SAR2optical system

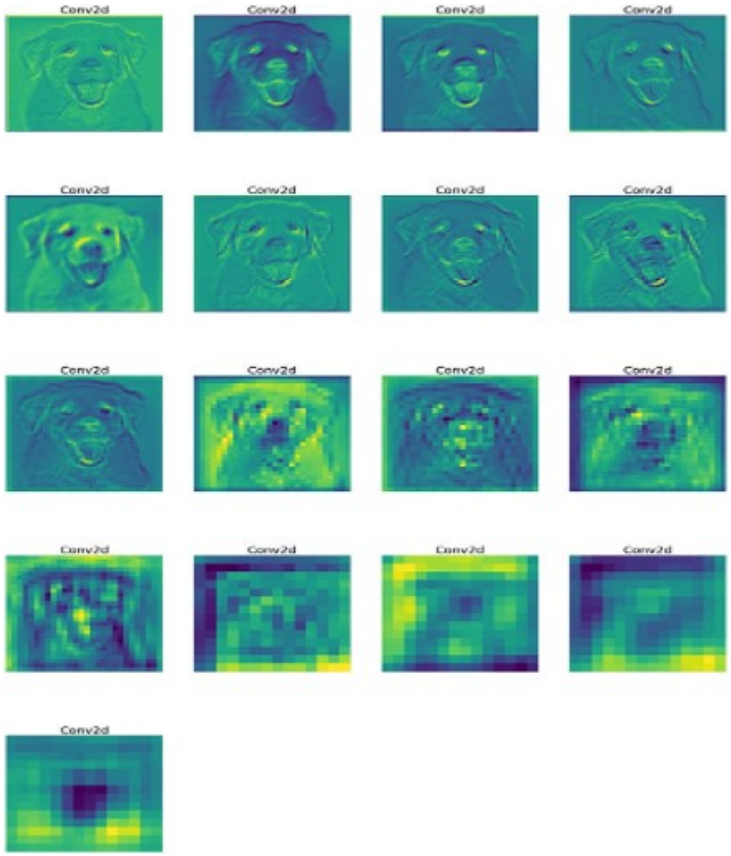

Figure 2.22 Dog feature map as the image passes through the layers

Chapter 3

State of the art

Due to the need to color SAR images, many techniques have been developed to provide a more reliable and easier-to-analyze image. Until the developments in ML and NNs, most techniques focused on color domain transformations based on the intrinsic aspects of SAR systems.

Due to the importance of augmenting color information in SAR images, many fake coloring techniques have been developed [1, 17]. These techniques are usually based on pseudo-coloring the image, where a color can be attributed to a specific range of backscatter values [1] or by combining multiple polarizations [17]. While these methods provide some visual information, they still fall short in providing accurate color information.

A simple yet effective pseudo-coloring technique was to map different colors to different backscatter values. For this, the grayscale images are divided into different intensity ranges. Each range is then assigned a unique color from a color spectrum. For instance, lower intensities might be mapped to blue shades, middle intensities to green, and higher intensities to red. This mapping transforms the single-channel grayscale image into a three-channel color image, as shown in Figure 3.1.

While these methods don't offer "realistic" colors, they still provide pseudo-colored images that make it easier to detect slight variations in backscatter values that for SAR images would mean slight gray variation. This method works by converting grayscale pixel values to a color domain like HSI or CIELab [1]. SAR systems are inherently noisy, where speckle noise can drag pixel values, creating the variations that become more noticeable when converting to color, requiring adjustments to color values by smoothing the variations.

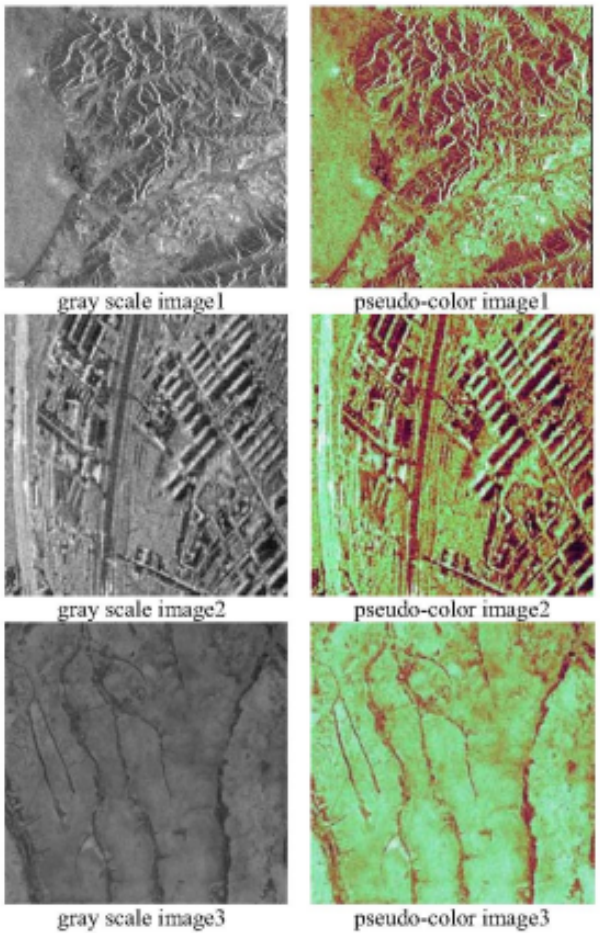

Figure 3.1 Pseudo-coloring technique based on CIELab color domain [1]

The selection of colors can be based on various criteria depending on the application. For some cases, colors are chosen to enhance contrast and highlight specific features of interest, such as distinguishing between water bodies, vegetation, and urban areas. By using pseudo-coloring, features that might be difficult to distinguish in a grayscale image become more apparent and easier to analyze, yet none offer a good representation for all objects present in the image.

Other methods take advantage of the fact that SAR systems can produce images in multiple polarizations (HH, VV, HV, and VH) to create pseudo-colored images [17, 18]. Each polarization captures different scattering characteristics of the Earth's surface. When visualized, these channels are typically in grayscale and are not very intuitive or visually appealing. By combining these channels, it is possible to augment the SAR image with color information based on the multiple polarization interaction.

This usually involves two main procedures. First, the method involves a fusion procedure that combines the four polarimetric channels into three derived Red Green Blue (color space) (RGB) decomposed bands [17, 18]. This step retains the correlation properties typical of natural color images. Next, the RGB image is converted into the CIELab color space, known for its perceptual uniformity. In the CIELab space, color equalization is performed to distribute the color information evenly. Finally, the image is converted back to RGB, resulting in a visually

intuitive representation that closely resembles natural color images, as shown in Figure 3.2 [17]. The channel decomposition captures the relationships between the polarizations that then can derive a set of values that can be mapped to the RGB color channels to create a pseudo-colored image.

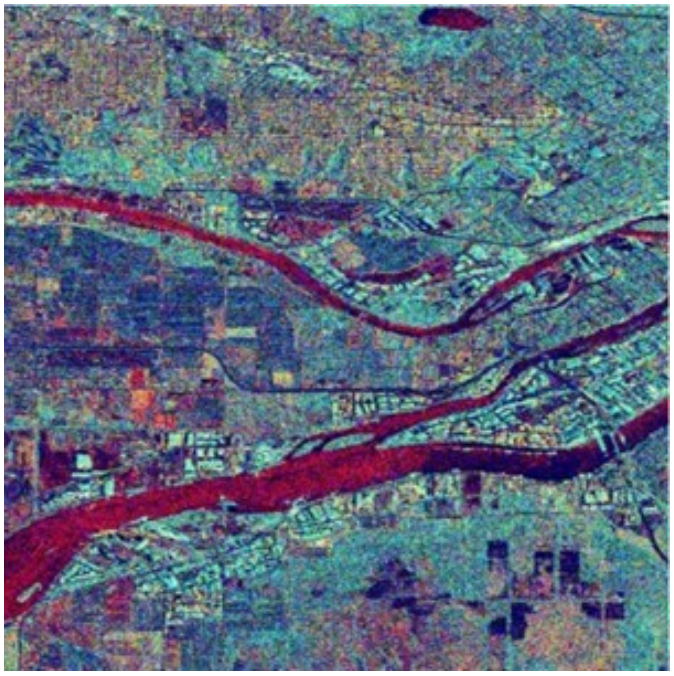

Figure 3.2 The RGB image obtained using the HH, HV, and VV polarizations

Another approach is to assign the most relevant components of the SAR image and map them to different colors. For this, Principal Component Analysis (PCA) can be used [19]. PCA is a statistical method used in the pseudo-coloring of SAR images to enhance visual interpretation by transforming multi-channel polarimetric data into orthogonal components. This technique identifies the principal components that capture the most variance in the data, which are then mapped to color channels (RGB), producing a composite image that highlights different features and structures.

PCA is applied to the multi-polarization data to extract the most informative components. Each polarimetric channel (e.g., HH, HV, VV) captures different scattering mechanisms from the Earth's surface. By applying PCA, these channels are transformed into a new set of uncorrelated components, each representing a significant pattern or feature in the data [19].

The first principal component typically contains the most significant information about the backscatter intensity, while the subsequent components capture additional details and variations that might not be as prominent in the original channels. By mapping these principal components to the red, green, and blue color channels, a pseudo-colored image is created, which enhances the visual differentiation of various surface features, as shown in Figure 3.3.

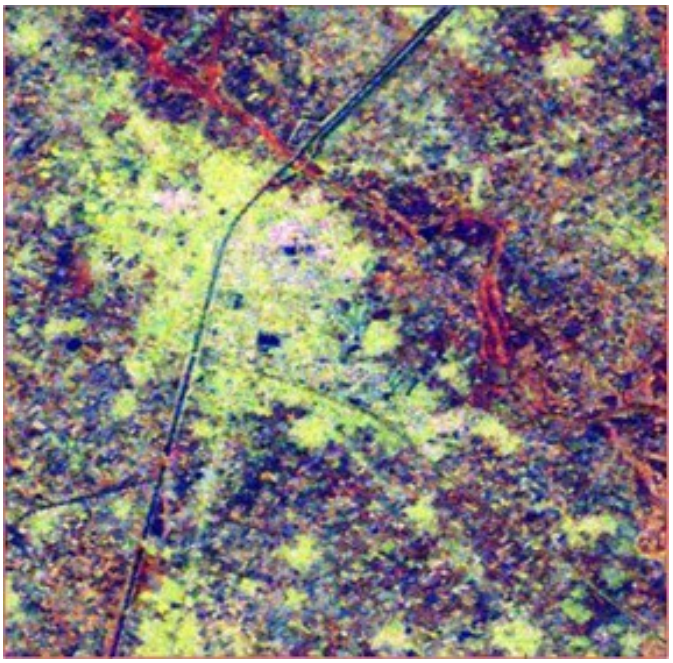

Figure 3.3 Pseudo-color RGB image obtained using PCA applied to the SAR image

Other approaches are based on classifying every pixel before coloring and creating a lookup table based on the classified pixels to assign a color [20]. Based on the statistical properties of the backscatter pixel, such as noise, speckle, and texture estimation, it is then possible to assign a label to the pixel such as urban, forest, ocean, agricultural field, etc. With this, it is possible to then create a lookup table that defines for each class what the correct pixel color is, as demonstrated in Figure 3.4.

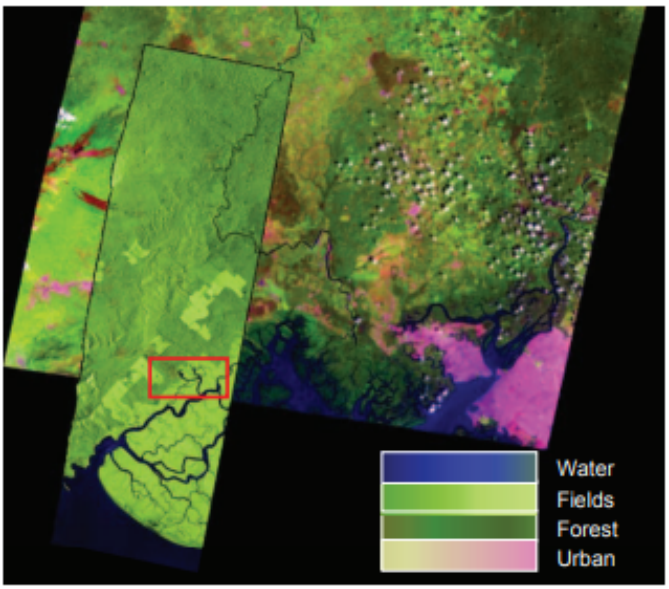

Figure 3.4 Overlay of the colorized TerraSAR-X image and the colorized SAR image

Despite these advancements, traditional techniques often lack the precision required for detailed analysis. They provide some visual insights, but their ability to convey accurate color information is limited. This limitation is particularly evident when the goal is to reproduce the

true optical appearance of the scene or to enhance the detection of specific features that are critical for applications such as environmental monitoring, urban planning, and disaster management.

More recently, with the developments in deep NNs, the approach evolved into searching for features that, while too complex to be used by more traditional techniques, offer possible clues as to the color information of the area of interest [18, 21, 22, 23]. The introduction of CNNs expanded the range of approaches by allowing the network to "interact" with the image, learning its patterns and features.

These features can then be combined to create a feature pool that together allows the network to “hallucinate” the correct colors as if an optical system was used to capture the region. By combining the strengths of multiple specialized networks, it is possible to create a pipeline that extracts hierarchical multi-scale spatial features from the grayscale SAR image, using CNNs, and map these spatial features to other meaningful representations that then would allow us to either classify or create new images.

Current approaches utilize models such as GANs coupled with models such as U-net loosely based in autoencoder architectures, yet with expanded latent space bottleneck as to capture an higher number of features [21, 22, 24, 23, 25]. This comes with the trade off of an exponential increase in computation needed. Some other approaches utilize NN to apply the segmentation and look-up coloring technique, where the network classifies the pixel class and applies an uniform color to the RGB image [26]. Other diffusion based models, that are probabilistic models that start with noise (randomly placed pixels) and gradually refine the image until it reaches the desired output [27, 28, 29].

Chapter 4

Methodology

As programming languages and overall tools got better, the knowledge entry level for a user got lower, requiring less technical skills, and abstracting overall aspects where behind the curtains the programming languages/frameworks/algorithms do the heavy lifting. This, however, sometimes creates a fog around these black boxes that, if there isn't a deeper understanding of the mechanisms, can hinder progress and require extensive trial-and-error life cycles.

This chapter aims at covering some of the techniques that are abstracted by tools such as some of the algorithms behind frameworks like Tensorflow for AI NN model training and other general intermediary steps.

4.1 Convolutional Neural Networks

CNNs are a specialized type of NN extensively employed in computer vision tasks, such as image recognition, pattern detection, and object identification. These networks leverage the spatial structures inherent in images to efficiently extract relevant features.

4.1.1 CNN Core

At the core of CNNs are layers known as convolutional layers. These layers function by applying a convolutional operation to the input image. This operation involves a kernel or filter, which is a small matrix of weights, that slides over the image spatially (often referred to as a sliding window). At each position over the image, the kernel performs an element-wise multiplication with the values of the image pixels it covers. The results of these multiplications are then summed up to produce a single value. This value represents the response of the filter at that specific location, indicating how strongly the features detected by the filter are present in that segment of the image.

The kernel is moved across the entire image, usually moving a few pixels at a time (this movement is specified by what is called stride). At each stop, the convolution operation is performed, and an output value is generated and recorded in the output feature map. This feature map effectively represents a filtered version of the original image, where each value

indicates the presence or strength of a feature at different locations in the image, as shown in Figure 4.1.

Suppose f represents the input image and g represents a kernel (or filter). The convolution of f and g is given by Equation 4.1:

$$(f * g)(i,j) = \sum_m \sum_n f(m,n) \odot g(i-m, j-n) \tag{4.1}$$

Where m and n iterate over the kernel, and i, j represent the spatial location in the output feature map. This slides the kernel g over the image f, multiplying and summing up products of the values in the overlapping entries at each position. Each element in the resulting matrix from the convolution operation, $(f * g)(i,j)$, represents how much the kernel g matches the part of the image at position (i,j). Thus, the entire output matrix forms a "feature map" that highlights patterns or features in the image as detected by the kernel.

In essence, these networks can be summed up to matrix multiplication and additions, where the input is convolved with the values of a kernel. The size of the kernel and the stride parameter influence the dimensionalities of the resulting feature maps. The kernel size determines the spatial block that the model will use for pattern analysis, while the stride controls the step size during convolution. Padding can be applied to maintain the convolution product's size. The values within this kernel define the features that the CNN focuses on, representing what the network seeks to learn to better capture essential input structures.

As shown in Figure 4.1, this process results in the creation of a feature map, which represents how well the kernel's features match various parts of the image.

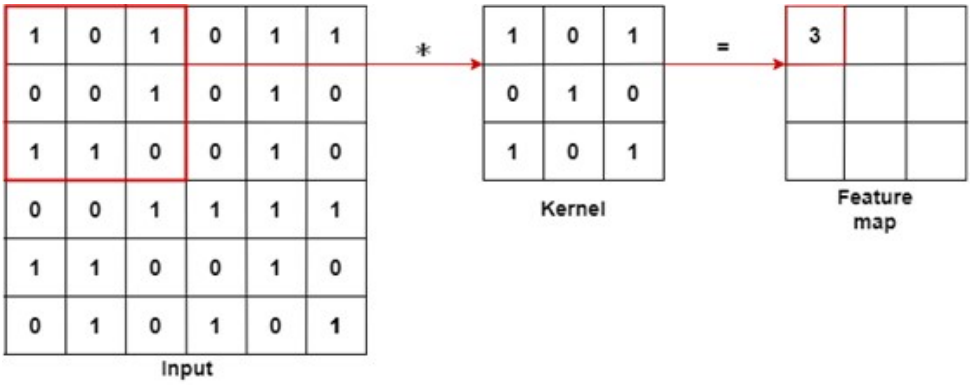

Figure 4.1 Convolution of an input image with a kernel resulting in a feature map.

4.1.2 Activation functions

After the convolution layer applies a filter to an input image and creates a feature map, an activation function is applied to introduce non-linearity into the model. This non-linearity allows the network to capture more complex patterns and interactions in the data, which are not possible to model with just linear operations like convolutions. It does not matter how large the network is, without an activation function to introduce non-linearity it would behave as if it was a single-layer perceptron. Neural networks operate by processing inputs sequentially, with each step's

output depending on the previous step's state. This process involves updating the state of the network that takes the previous state, the current input, and a set of parameters. The network parameters include weights for the inputs and the recurrent connections, as well as biases, all of which are adjusted during training to minimize error. In the context of training these networks the calculation of gradients, which are used to update the parameters, are based on backpropagation. When a network is composed of multiple layers the backpropagation will calculate the current layer's error based on the previous layers, starting from the output layer, and moving through until the input layer, usually called chain rule. To calculate the weights for a given layer the network it must first calculate and multiply the gradients of all the following layers. A problem appears when the gradients for a given layers start to get increasingly smaller or larger.

4.1.3 Activation functions

Assuming a simple NN, where the loss function L depends on the output y, which in turn is a function of intermediate layer outputs $z_1, z_2, \ldots, z_n$ and these layers are functions of the weights $W_1, W_2, \ldots, W_n$. The chain rule allows us to compute the gradient of L with respect to any weight W_i by multiplying the gradients through each layer from the loss back to that weight.

Mathematically, if $y = f_n(z_{n-1})$ and each $z_i = f_i(z_{i-1}; W_i)$ for a set of functions f defined by the network architecture, the gradient of the loss L with respect to a weight matrix W_k is given by Equation 4.2.

$$\frac{\partial L}{\partial W_k} = \frac{\partial L}{\partial y}\frac{\partial y}{\partial z_{n-1}}\frac{\partial z_{n-1}}{\partial z_{n-2}} \cdots \frac{\partial z_{k+1}}{\partial z_k}\frac{\partial z_k}{\partial W_k} \tag{4.2}$$

Each term in the product is a partial derivative that propagates the error back from the output towards the weights. The term $\frac{\partial z_k}{\partial W_k}$ directly measures how the k-th layer's output changes with its weights, indicating where to adjust weights to decrease the loss.

Gradient vanishing occurs due to the multiplicative effect of gradients. As the network iterates through the data, it might reach certain situations where one of the gradients for a given layer converges to small values. For such cases, where the term $\frac{\partial z_{i+1}}{\partial z_i}$ converges to 0, a cascade effect will occur where, due to the multiplicative effect, the previous layers will also tend towards smaller gradients, which will hinder training, causing gradient vanishing [30]. When this occurs, the network will have difficulties converging towards the optimal solution, as the weight updates become less significant.

Conversely, considering that each derivative term $\frac{\partial z_{i+1}}{\partial z_i}$ might have values greater than 1. For such cases, as the derivatives are multiplied over the many layers, as in a deep network, or over many time steps, the product can grow exponentially. This means that for every update to the weights, their variations become more pronounced, making the network unable to converge towards the optimal solution, as it will "jump" more randomly through the loss function, causing gradient explosion, as illustrated in Figure 4.2.

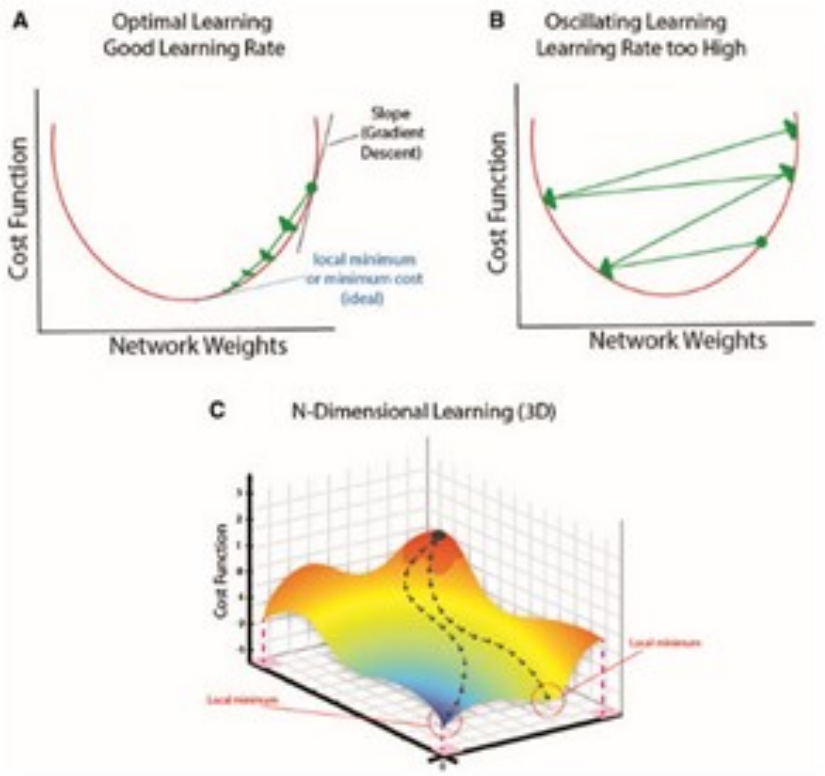

Figure 4.2 Gradient vanishing (left) and explosion (right)

Sigmoid function

The sigmoid activation function, often represented mathematically by Equation 4.3, and it effectively compresses its input range, which can be any real number, to an output range between 0 and 1. This function is inherently non-linear, allowing the network to learn more complex features that a linear model wouldn't be able to [31]. This function is characterized by a smooth derivative, which facilitates the computation of gradients. The sigmoid's curve transitions outputs from the negative input domain towards zero and positive inputs towards one.

However, the sigmoid function has a drawback, especially in deep NN with many layers. Because the output is limited to the range [0,1], the outputs of neurons using sigmoid are also "squashed" into this narrow range [31]. This squashing effect significantly impacts the gradients during the backpropagation process. Specifically, for very high or very low inputs, the sigmoid function's slope (derivative) approaches zero, leading to very small gradients, which in turn leads to the problem of gradient vanishing, as illustrated in Figure 4.3.

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{4.3}$$

Hyperbolic tangent

The Hyperbolic tangent (tanh) function shares a structural resemblance with the Sigmoid function but differs in the range of its output, which spans from -1 to +1 [31]. This characteristic allows it to center its output around zero, which is often beneficial during the training of neu-

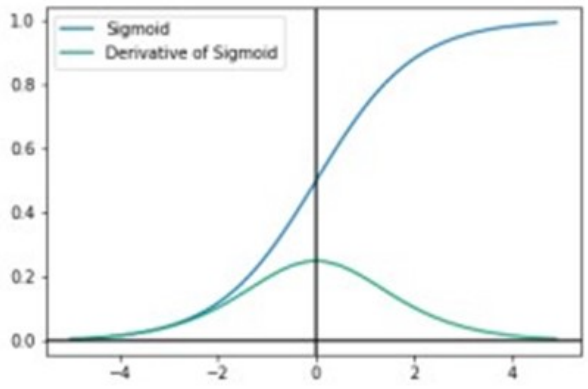

Figure 4.3 Sigmoid function and its derivative

ral networks as it tends to lead to faster convergence. Similar to the sigmoid, tanh is also a smooth, continuously differentiable function, which is advantageous for gradient-based optimization methods. One difference of the tanh function is that it has a steeper derivative compared to the sigmoid.

This steeper slope can be advantageous because it allows for larger gradient values early in the training process, potentially accelerating the learning speed. However, despite these larger gradients, the tanh function can still lead to the vanishing gradient problem. This issue arises because, like the sigmoid, the outputs of tanh are dependent on previous layers. As inputs become large or small in magnitude, the function saturates, meaning the gradient approaches zero. Thus, during backpropagation, these small gradients can multiply through several layers of a network, compounding into even smaller values that minimally adjust the weights in earlier layers of the network. The tanh function can be defined as in Equation 4.4.

$$\tanh(x) = \frac{1}{1 + e^{-2x}} \tag{4.4}$$

The behavior of the tanh function and its derivative is depicted in Figure 4.4, showing how it saturates for large positive or negative inputs.

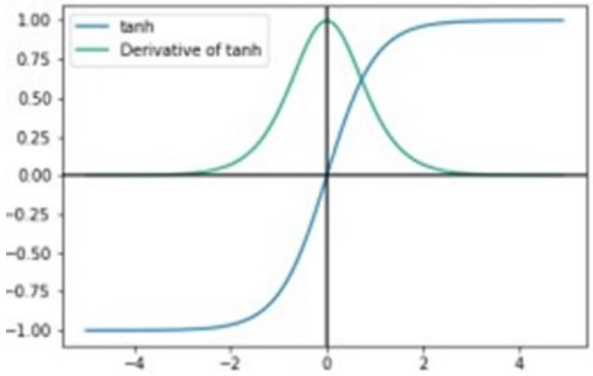

Figure 4.4 Hyperbolic tangent and its derivative

Rectified Linear Unit (ReLU)

The Rectified Linear Unit (ReLU) has become the most widely used activation function among researchers in the field of deep learning [31]. The popularity of ReLU can largely be attributed to its superior training performance compared to traditional activation functions like the logistic sigmoid and the hyperbolic tangent. ReLU is defined by Equation 4.5:

$$\text{ReLU}(x) = \max(0, x) \tag{4.5}$$

As seen in Equation 4.5, it outputs the input directly if it is positive; otherwise, it outputs zero. For positive input values, ReLU acts as an identity function, ensuring that the gradient is neither amplified nor diminished during backpropagation, which helps in mitigating the vanishing gradient problem that is prevalent with sigmoid or tanh functions.

One significant advantage of using ReLU is its ability to introduce non-linear properties without affecting the magnitude of the input for positive values, as shown in Figure 4.5. This characteristic allows for efficient backpropagation and convergence during training by maintaining strong gradients when the neuron is active [31].

ReLU also contributes to creating sparse representations, which refers to the idea of having a reduced number of non-zero values in the feature maps compared to the total number of values. This is beneficial as having fewer non-zero values means less storage space is required and computation becomes more efficient during the forward and backward passes of the neural network. Sparse representations can capture the most discriminative features of the input data while filtering out irrelevant or redundant information, leading to better generalization performance, especially when dealing with high-dimensional data.

ReLU's unbounded output for positive values can sometimes lead to issues for certain types of networks. For instance, in Recurrent Neural Network (RNN)s, the use of ReLU might not be advisable due to the potential for very large outputs, which can increase the risk of exploding gradients.

Additionally, another issue known as the "dying ReLU" problem occurs when inputs to a ReLU neuron are negative, causing the outputs to be zero. In such cases, the neuron stops learning entirely, as the gradient through the function is zero [Review of Activation]. This phenomenon is a particular instance of the vanishing gradient problem, where once a neuron gets "stuck" in this way, it is unlikely to recover, as it no longer contributes to the network's adaptation during training.

Leaky ReLu

Leaky ReLU is a variant of the ReLU activation function, designed to address the issue of the dying ReLU problem [31]. The dying ReLU problem occurs when ReLU neurons become inactive and only output zeros because their inputs are negative, and so the network stops learning. The Leaky ReLU function modifies the standard ReLU by allowing a small, non-zero gradient when the unit is not active, i.e., when the input is less than zero. Mathematically, the

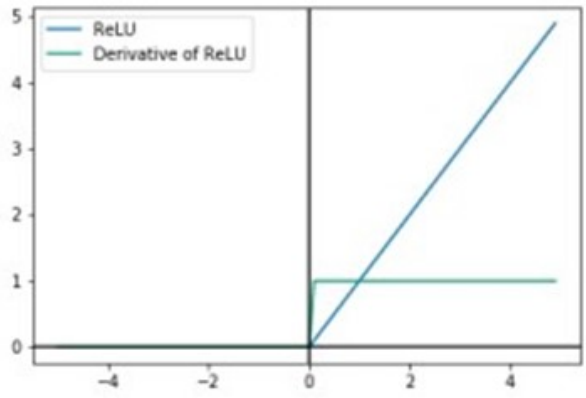

Figure 4.5 ReLU and its derivative

Leaky ReLU is defined by Equation 4.6.

$$f(x) = \max(\alpha x, x) = \begin{cases} x, & x > 0 \\ \alpha x, & x < 0 \end{cases} \tag{4.6}$$

As shown in Equation 4.6, x is the input to the neuron, and α is a small coefficient (typically a small positive number like 0.01). This small slope α ensures that the gradient is not zero even when the neuron's input is negative. By allowing a small gradient when the input is negative, Leaky ReLU keeps the weight updates alive during the backpropagation process, which helps to prevent neurons from dying. By preventing the neurons from dying, all of these can potentially update and adapt during training, leading to more robust learning. Like standard ReLU, Leaky ReLU does not saturate in the positive domain and is computationally efficient. It retains the non-linear properties that help neural networks learn complex functions. The comparison of both ReLU and Leaky ReLU activation functions can be observed in Figure 4.6.

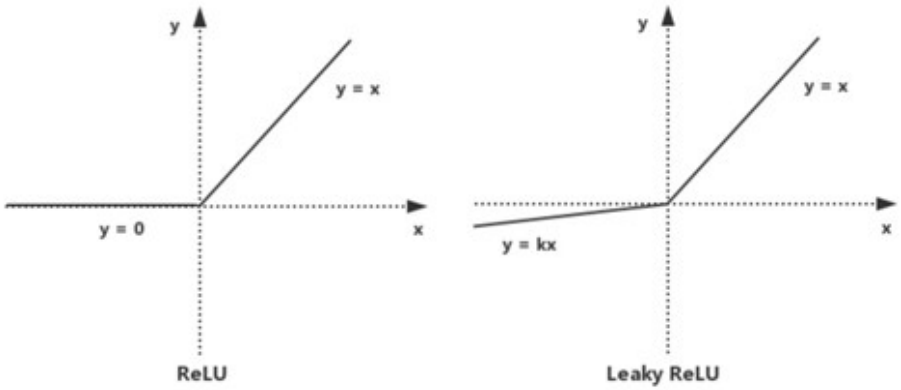

Figure 4.6 ReLU (left) and Leaky ReLU (right) activation functions

Other activation functions

The Exponential Linear Unit (ELU) [31] was initially introduced as an enhancement to the ReLU, aiming to address some of its problems. Like ReLU, the ELU function retains the identity for non-negative values (i.e., $x \geq 0$), but it adopts a different behavior for negative inputs. For $x < 0$, ELU outputs values based on an exponential curve, specifically $\alpha(e^x - 1)$, where α is a constant that defines the value to which the ELU converges towards for negative inputs. This allows ELU to output negative values, unlike ReLU, which can help mitigate the vanishing gradient problem by maintaining a more robust gradient flow during training. Additionally, the smooth transition of ELU to its asymptote helps improve learning dynamics by making the activation function more continuous and differentiable throughout its domain.

Swish is another innovative activation function that emerged from empirical experimentation using reinforcement learning [31]. It is defined as $\text{Swish}(x) = x \cdot \sigma(\beta x)$, where β is either a constant or a trainable parameter. This function is notable for being smooth and non-monotonic, meaning it does not consistently increase or decrease. Swish retains the unboundedness above like ReLU, which can be advantageous for maintaining active gradients during deep network training. However, unlike ReLU, Swish is also smooth, which provides the benefits of derivatives that are well-behaved across the entire range of input values.

Mish is a relatively new activation function that shares similarities with Swish in terms of its smooth, non-monotonic profile [31]. Mish is mathematically defined as $\text{Mish}(x) = x \cdot \tanh(\text{softplus}(x))$, where $\text{softplus}(x) = \log(1 + e^x)$. The key distinction of Mish lies in its behavior in the negative domain, where it exhibits a concave curve. This subtle modification allows Mish to potentially outperform both ReLU and Swish by facilitating even better gradient propagation through deep networks, albeit at a higher computational cost due to its more complex mathematical operations.

This behavior of the ELU, Swish, and Mish functions, as well as their derivatives, can be observed in Figure 4.7.

By introducing these non-linearities, activation functions help the network learn complex patterns across large and varied datasets. They make it possible for each layer in a CNN not only to perform template matching (via the filters in convolution layers) but to model hierarchical patterns. For instance, initial layers might detect simple edges or textures, while deeper layers might use these inputs to detect higher-order features such as small patterns, hidden information or complex shapes.

Therefore, the choice of activation function and its properties (like non-linearity and the ability to avoid gradient problems) significantly influences the training dynamics and performance of a CNN. Different functions may be more suitable for different tasks or datasets, affecting how effectively a network can train and generalize from its training data to real-world applications.

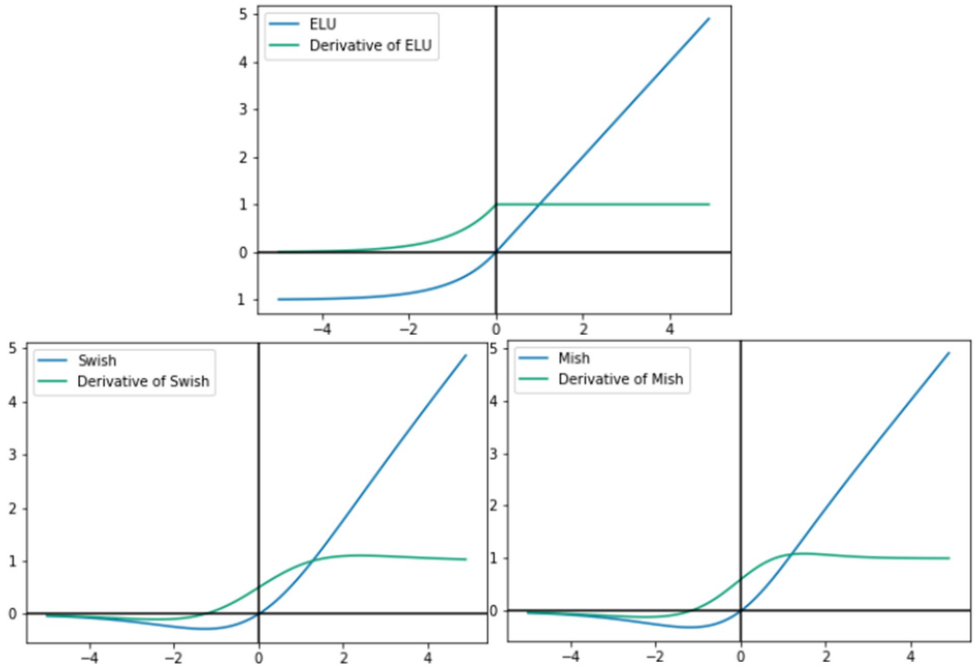

Figure 4.7 ELU (top), Swish (left), and Mish (right) activation functions and derivatives.

4.1.4 Loss functions

In the context of neural networks, error refers to the difference between the predicted output of the network and the actual ground truth output. It is a measure of how well the model's predictions match the expected outcome. Minimizing this error is the primary goal of training a neural network, as it directly impacts the model's performance and its ability to generalize to new, unseen data.

To measure this error a loss function can be applied. Loss functions provide a measure that the optimization algorithm can use to adjust the model's parameters (weights and biases). During training, the model makes predictions on the input data, and the loss function calculates the loss based on these differences.

MSE

The MSE is a widely used loss function in regression tasks, particularly in training NN models. It is preferred for its simplicity and its direct relationship to the Euclidean distance between the predicted and true values. The MSE measures the average of the squared differences between the predicted output values and the actual (ground truth) output values. This provides a clear, interpretable metric for how well a model is performing, especially when dealing with continuous data.

Mathematically, MSE is defined by Equation 4.7.

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{4.7}$$

where n represents the number of data points, y_i is the ground truth value (the actual

target), and $\hat{y}_i$ is the predicted value by the model for the i-th data point.

The term $(y_i - \hat{y}_i)$ is referred to as the residual, which quantifies the error between the prediction and the actual value for each data point. By squaring this residual, the MSE penalizes larger errors more heavily than smaller ones. This is useful when the model wants to prioritize minimizing large deviations. Visually MSE can be described by Figure 4.8.

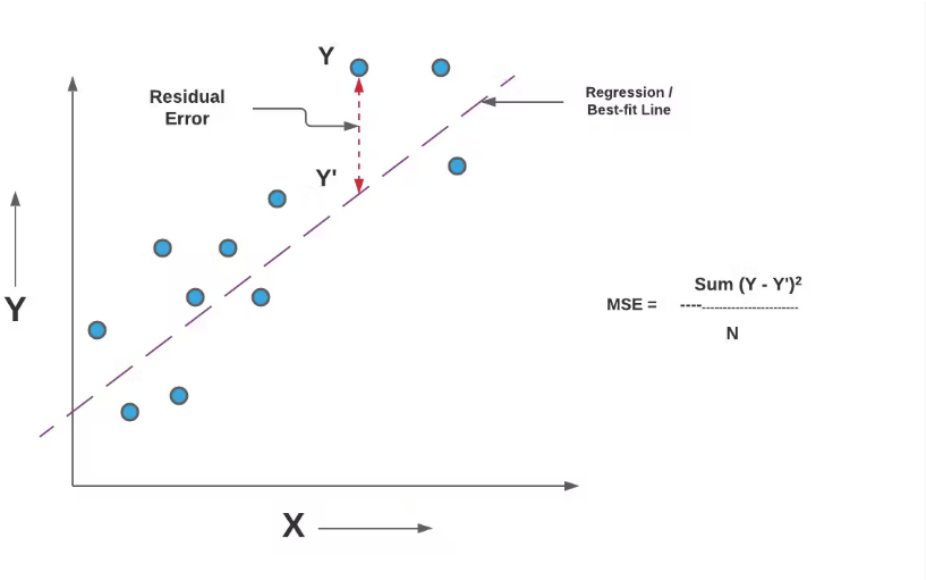

Figure 4.8 Mean square error function

The key idea is that during the training process, the NN attempts to minimize the MSE by adjusting its internal parameters (weights and biases) using a method like gradient descent. This is done by computing the gradient of the loss function with respect to the model's parameters and updating them in the direction that reduces the MSE. The gradient with respect to each model parameter w can be computed using Equation 4.8.

$$\frac{\partial \text{MSE}}{\partial w} = \frac{2}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i) \cdot \frac{\partial \hat{y}_i}{\partial w} \tag{4.8}$$

This equation calculates the gradient of the MSE with respect to a particular weight w in the network. The partial derivative $\frac{\partial \hat{y}_i}{\partial w}$ represents how the model's prediction $\hat{y}_i$ changes with respect to the weight w.

The learning process works by iteratively updating the weight w using the gradient Equation 4.9.

$$w_{\text{new}} = w_{\text{old}} - \alpha \cdot \frac{\partial \text{MSE}}{\partial w} \tag{4.9}$$

Here, α is the learning rate, a hyperparameter that controls the step size during the update. By repeating this process over many iterations, the model learns to reduce the MSE, and consequently, its predictions become more accurate.

The squaring of the errors means that even small errors are amplified, making the MSE highly sensitive to outliers.

For tasks such as image reconstruction, MSE can be problematic. By focusing on individual pixel values without considering neighboring pixels, MSE fails to account for the spatial structure or patterns that exist in the data. This lack of contextual understanding means that even if the pixel-wise errors are minimized, the resulting image may still look distorted or unrealistic leading to chromatic aberrations.

For example, in image reconstruction, using MSE alone might produce a solution where the pixel values are close to the target values, but visually, the image could appear blurry or lack sharp edges. This because MSE doesn't prioritize the preservation of local structure or texture. It simply aims to reduce pixel-wise differences.

SSIM

There are other functions that take into account the surroundings, one example of which is the Structural Similarity Index (Structural Similarity Index (SSIM)). SSIM is a perceptual metric used to measure the similarity between two images. Unlike MSE, which only considers pixel-wise differences, SSIM is designed to account for changes in structural information, luminance, and contrast that are more aligned with human visual perception.

The formula for SSIM can be expressed as:

$$\text{SSIM}(x, y) = \frac{(2\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)} \tag{4.10}$$

Where:

- μ_x and μ_y are the mean intensity values of images x and y, respectively (the luminance component),
- σ_x^2 and σ_y^2 represent the variance (or contrast) of images x and y,
- σ_{xy} is the covariance between the two images, representing their structural similarity,
- C_1 and C_2 are small constants used to stabilize the division when the denominator is close to zero.

SSIM effectively combines three components:

Luminance: Measures how similar the brightness values are between the two images, and is defined by Equation 4.11.

$$l(x, y) = \frac{2\mu_x\mu_y + C_1}{\mu_x^2 + \mu_y^2 + C_1} \tag{4.11}$$

Contrast ($c(x, y)$): Compares the contrast between the two images, which is based on their standard deviations, defined by Equation 4.12.

$$c(x, y) = \frac{2\sigma_x\sigma_y + C_2}{\sigma_x^2 + \sigma_y^2 + C_2} \tag{4.12}$$

Structure ($s(x, y)$): Evaluates how similar the structures of the images are by comparing their correlation. This can be calculated through Equation 4.13.

$$s(x, y) = \frac{\sigma_{xy} + C_3}{\sigma_x \sigma_y + C_3} \tag{4.13}$$

The full SSIM index is the product of these three terms:

$$\text{SSIM}(x, y) = [l(x, y)]^{\alpha} \cdot [c(x, y)]^{\beta} \cdot [s(x, y)]^{\gamma} \tag{4.14}$$

where α, β, and γ are used to adjust the relative importance of each component, and typically $\alpha = \beta = \gamma = 1$.

SSIM ranges from -1 to 1, where 1 indicates perfect similarity between the two images, 0 means no similarity, and -1 indicates complete dissimilarity. However, for practicality for training NN models, the value of SSIM is converted to the range between 0 and 1, where higher values indicate a stronger similarity.

One of the strengths of SSIM over MSE is that it takes into account the structure of the image. For example, MSE only calculates pixel-wise differences and treats each pixel independently, while SSIM evaluates local patches of the image, considering the relationships between neighboring pixels. This makes SSIM a more effective metric for images because it reflects changes in perceptual quality that MSE might miss. For example, SSIM is sensitive to differences in structure and texture, which are essential for understanding real-world images.

As illustrated in Figure 4.9, MSE stays consistent (0.04 in both cases), while SSIM improves more naturally (from 0.15 to 0.85) when blur and a constant are added to the image respectively, reflecting the fact that SSIM considers more than just pixel-wise differences. By considering the local mean, variance, and covariance of pixel intensities, SSIM provides a more accurate measure of perceptual image quality.

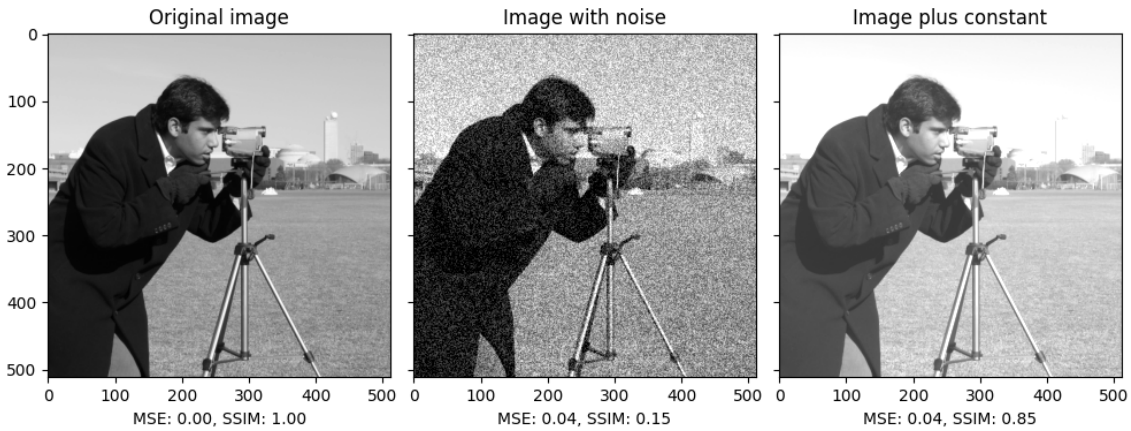

Figure 4.9 MSE stays consistent (0.04 and 0.04) while SSIM improves more naturally (0.15 and 0.85) when adding blur and a constant

PSNR

Other loss functions like the Peak Signal-to-Noise Ratio (PSNR) quantifies the difference between the original and the processed image by measuring the ratio of the maximum possible power of a signal (the original image) to the power of the corrupting noise (the error introduced by compression or reconstruction). PSNR is derived from the MSE, where it applies a domain conversion of the MSE error to the logarithmic scale, converting it to decibels (dB). Higher PSNR values indicate better image quality, as they signify that the error between the original and processed image is low. PSNR however faces the same challenges as MSE, where these do not correlate well with human visual perception. A high PSNR value does not always correspond to perceived high quality because it does not consider perceptual factors like structural similarity and contrast sensitivity. Both can also be significantly affected by a few large errors (outliers), which may not accurately reflect overall image quality as shown in Figure 4.10.

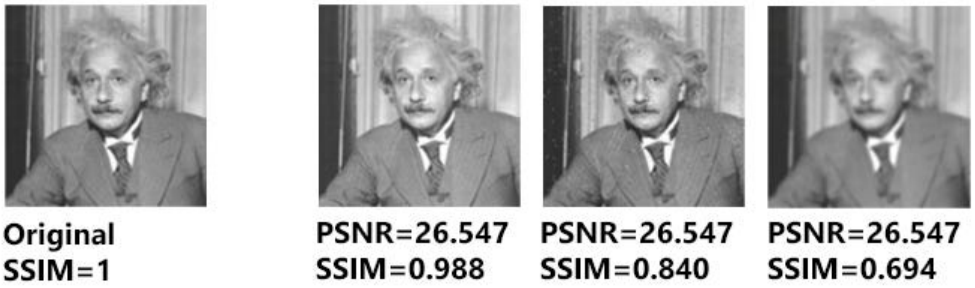

Figure 4.10 PSNR stays constant while SSIM decreases more naturally)

4.1.5 Hyper parameters

In ML models, there are two main types of parameters: model parameters and hyperparameters [32, 33]. Model parameters are values that are learned and updated through the data learning process, such as the weights of neurons in NNs. These parameters are adjusted by the algorithm during training to minimize the loss function and improve the model's performance. On the other hand, hyperparameters are predefined before the training process begins and are not learned from the data. Instead, they define the overall architecture and functioning of the ML model.

These include settings that determine the structure of the model, such as the number of layers and nodes in a deep NN, as well as parameters that affect the learning process itself, such as learning rates, step sizes, initialization conditions, and momentum decay parameters. These settings must be carefully chosen, as they can significantly impact the model's ability to learn from data and generalize to new, unseen data.

Hyperparameters can be categorized based on their nature: they can be discrete, con-

tinuous, categorical, or binary [32, 33]. Discrete hyperparameters include values such as the number of clusters in a clustering algorithm, while continuous hyperparameters might include the learning rate, which can take any value within a range. Categorical hyperparameters refer to selections among a set of options, such as the type of optimizer used in training (e.g., Adam, Stochastic Gradient Descent (SGD), RMSprop).

Properly chosen hyperparameters can lead to better model quality, both in terms of training performance and the model's ability to generalize to new data when deployed in real-world applications. Therefore, optimizing hyperparameters is an important step in the ML pipeline, often involving techniques such as trial and error, grid search, random search, or more advanced methods like Bayesian optimization. These techniques aim to find the best combination of hyperparameters to maximize the model's performance and robustness. While there are no rules or right solutions when choosing hyperparameters, through know-how it is common for AI researchers to fine-tune these hyperparameters through a trial-and-error approach.

Other approaches include a more thought-out approach, ranging from other AI models such as genetic algorithms, that by assigning parameters as genetic characteristics are able to mutate through possible combinations, keeping the best ones. However, due to their simplicity the most common approach to finding optimal hyperparameters is through grid search combined with trial and error, where the algorithm tests multiple configurations of hyperparameters to find the optimal matches.

The Table 4.1 shows the most commonly used hyperparameters [32, 33].

Hyperparameter	**Common Techniques for Tuning**	**Typical Values**	**Description**
Number of Layers	Manual tuning, grid search	1-100+	Defines how many layers the neural network will have. More layers can capture more complex patterns but may lead to overfitting.
Number of Nodes/maps per Layer	Manual tuning, grid search	10-1000+	Specifies the number of neurons/maps in each layer. More neurons/maps can increase model capacity but also computational complexity.
Activation Function	Manual selection	ReLU, Sigmoid, Tanh, Leaky ReLU	Determines the function applied to the output of each neuron. It influences how the network learns nonlinear relationships.

Learning Rate	Grid search, random search	0.0001 - 0.5	Controls the step size during gradient descent. A higher learning rate can speed up training but may overshoot minima; a lower rate ensures convergence but may be slow.
Learning Rate Decay	Manual tuning	0.1 - 0.9 (decay factor)	A technique to decrease the learning rate over time, helping the model settle into a minimum more precisely.
Learning Rate Schedule	Manual selection	Step decay, exponential decay, adaptive	Predefines changes in the learning rate at certain epochs, which can help the model converge.
Batch Size	Grid search, random search	16, 32, 64, 128, 256, 512	Defines the number of samples processed before the model is updated.
Epochs	Manual tuning	10-1000+	The number of complete passes through the training dataset. More epochs can improve learning but may lead to overfitting.
Momentum	Grid search, random search	0.5 - 0.99	Used to accelerate gradient vectors in the right directions, thus leading to faster convergence.
Weight Initialization	Manual selection	Random, Xavier, He	The method used to initialize the weights of the network, which affects the starting point of learning.
Dropout Rate	Grid search, random search	0.1 - 0.5	The fraction of neurons to drop during training, which helps prevent overfitting by ensuring the network does not rely on specific neurons.
Optimizer Type	Manual selection	SGD, Adam, RMSprop	Specifies the optimization algorithm used. Different optimizers can affect the convergence speed and stability.
Train/Test/Validation Split	Manual tuning	0.6/0.2/0.2	The proportion of training data set aside for training, testing, and validation.

Early Stopping	Manual tuning	5 - 50 (epochs)	The number of epochs with no improvement after which training is stopped.

Table 4.1 Most commonly used hyperparameters for CNNs

4.1.6 Upsample

Traditional CNNs encounter a limitation as they progress through layers, with the consecutive applications of convolutions and pooling operations to extract finer details it results in a reduction of the feature map size. This reduction leads to a loss of localization context, hindering the network's ability to precisely reconstruct spatial relationships.

Because of this, other techniques can be employed to increase the spatial resolution of the feature maps. Upscaling, or upsampling, are techniques that can yield good results while not having trainable parameters, which will not increase the computational resources needed. Upscaling is a process of increasing the resolution of an image, typically by a factor of two or more. It is often necessary when dealing with low-resolution images or when transitioning between different layers of a NN with varying spatial dimensions.

When upsampling, it performs the upscaling operation by using interpolation to fill in the new values between existing ones. The most common interpolation methods include nearest-neighbor interpolation, bilinear interpolation, or bicubic interpolation.

Nearest-neighbor interpolation assigns the value of the nearest pixel to the new pixel location [34]. It is the simplest interpolation method and involves no calculations beyond finding the nearest neighbor. However, it can produce blocky artifacts, especially when upscaling images by a large factor. Bilinear interpolation considers the four nearest neighbors of the new pixel location and computes the weighted average of their values. It provides smoother results compared to nearest-neighbor interpolation and is computationally more intensive. Bilinear interpolation works well for most general-purpose image upscaling tasks [34].

Assuming that there is a set of defined data coordinates (X_k, Y_k), where $k = 1, 2$. These coordinates define the position of the points Q_{11}, Q_{21}, Q_{12}, and Q_{22}. For any given x and y coordinates, which are located in between the x_k and y_k points, by applying the bilinear interpolation technique, it is possible to find the point P as shown in Figure 4.11.

To find point $P(x, y)$ through bilinear interpolation, a two linear interpolations along the x-axis can be applied, finding intermediate points R_1 and R_2. Then, after performing one linear interpolation along the y-axis it is possible to find point P [34, 35].

R_1 and R_2 can be found using Equation 4.15 and Equation 4.16, respectively:

$$R_1(x, y) = \frac{Q_{11} \times (x_2 - x)}{(x_2 - x_1)} + \frac{Q_{21} \times (x - x_1)}{(x_2 - x_1)} \tag{4.15}$$

$$R_2(x, y) = \frac{Q_{12} \times (x_2 - x)}{(x_2 - x_1)} + \frac{Q_{22} \times (x - x_1)}{(x_2 - x_1)} \tag{4.16}$$

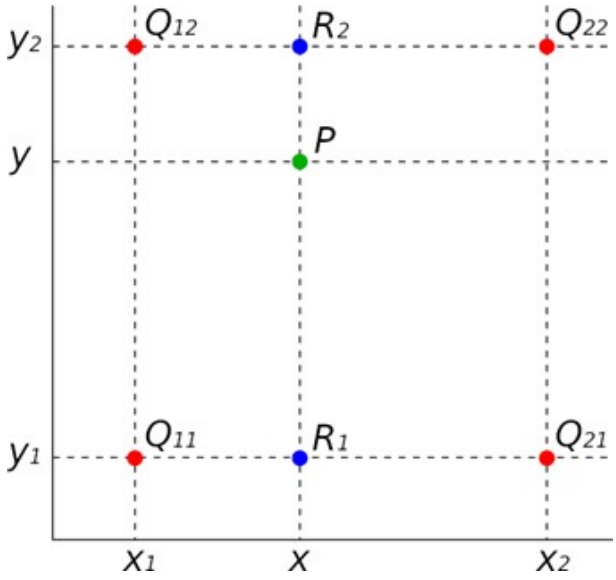

Figure 4.11 Bilinear interpolation example.

The interpolated point $P(x, y)$ is then defined as in Equation 4.17:

$$P(x, y) = \frac{R_1 \times (y_2 - y)}{(y_2 - y_1)} + \frac{R_2 \times (y - y_1)}{(y_2 - y_1)} \tag{4.17}$$

Bicubic interpolation is an extension of bilinear interpolation that considers a larger neighborhood of 16 pixels surrounding the new pixel location. It computes the weighted average of these 16 pixels using a cubic function. Bicubic interpolation tends to produce smoother and more visually appealing results compared to bilinear interpolation. However, it is computationally more expensive.

There are other less known techniques like the Lanczos interpolation [34] that uses a windowed sinc function to compute the weighted average of neighboring pixels. It provides high-quality results with less blurring compared to bilinear and bicubic interpolation. Lanczos interpolation is particularly effective for upscaling images while preserving sharp edges and fine details. However, it is computationally more demanding than other interpolation methods.

4.1.7 Pooling

Pooling is the opposite of upsampling. When applying pooling the resulting image is a smaller sized image of the original. This technique isusually applied to feature maps and reduces the spatial dimensions of these, which helps in minimizing computational complexity, preventing overfitting, and capturing the most important features.

A common type of pooling is maxpooling. In maxpooling, the input feature map is partitioned into a set of regions. Afterwards a sliding window, usually with sizes of 2x2 or 3x3, passes through the image. For each region, the maximum value is selected and forwarded to the next layer, compressing the feature in that region regardless of its exact location. This operation retains the most prominent features detected by the convolutional filters while discarding less significant activations, thus making the network more robust to variations and distortions

in the input data. These pooling layers also help reduce the number of learnable parameters, as by reducing the feature map, the amount of convolutions with the kernel will be lower. There are three common types of pooling operators. The maxpooling, that as described selects the feature map pixel with the highest value as seen in Figure 4.12. The average pooling calculates the mean of the feature map pixels inside the filter and substitutes it with this new mean as seen in Figure 4.13. Lastly global pooling is a technique taht compresses every pixel value in a feature map to a single pixel value. Global pooling can also employ global maxpooling, where the highest value in the feature map is selected as seen in Figure 4.13, or a global average pooling, where a mean is performed over the feature map pixels and the single mean value is forwarded to further layers.

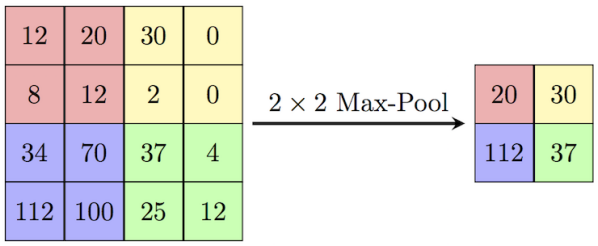

Figure 4.12 Maxpooling example

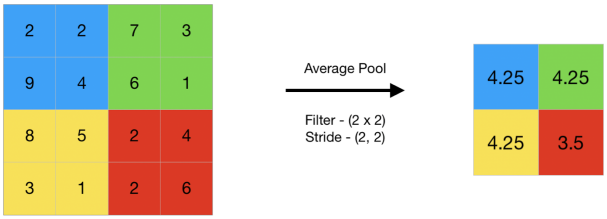

Figure 4.13 Average pooling example

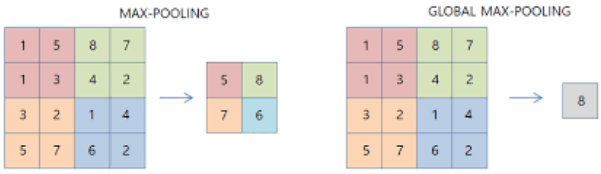

Figure 4.14 Max pooling (left) comparison with global max pooling (right)

4.2 Architecture

To utilize the advantages of both SAR widely usability and the ease of recognition that optical satellite systems offer, the model aims to create a network capable of extracting the main patterns present in SAR images and recreate the optical counterpart of what the same scene would look like. For that, CNN were opted for due to their optimized performance when applied to images.

A common challenge encountered in neural network development is the computational cost associated with training and running the model. Regardless of the hardware resources available, as the model size increases, it eventually reaches a computational ceiling due to limitations such as Video Random Access Memory (VRAM) constraints. This limitation can pose significant scalability and practical deployment challenges for the model. Consequently, it's crucial to design the model with computational efficiency in mind to ensure its feasibility for real-world applications. One effective technique for mitigating computational costs is applying max pooling, a downsampling operation that decreases the feature map resolution, enabling the layers to process smaller data more efficiently.

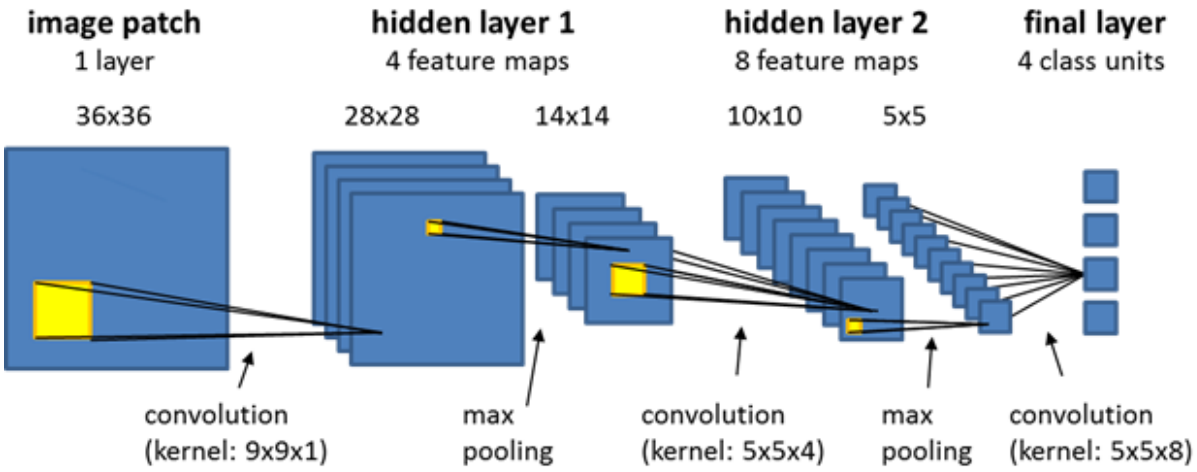

Figure 4.15 Different Pooling Layers for CNN

As the network progresses through successive convolutional layers, the resulting feature maps progressively reduce in size as the resolution decreases with each convolution operation. However, since the goal is to generate an optical image from SAR data, it is needed to maintain the output size equal to the input to preserve crucial details and spatial information. Achieving this balance requires employing upsampling techniques, which restore the spatial dimensions lost during downsampling as seen in Figure 4.16. However, maintaining a constant feature map size throughout the network can lead to increased computational overhead as there are more data points to process. Consequently, it may be necessary to optimize the model architecture by either reducing the number of feature maps per layer or decreasing the number of layers, though this could compromise the model's ability to generalize and capture the features in the data, reducing accuracy.

Due to these constraints, the CNN model opted for was based on the autoencoder architecture. Autoencoders are specialized CNN architectures designed to learn a compact and meaningful lower dimension latent representation, usually called latent space, of images through an encoding and decoding process. By utilizing autoencoders, it can effectively reduce the computational burden of the model while retaining its capacity to extract essential features from SAR images and reconstruct their optical counterparts with high fidelity. It is composed of an encoding layer, which reduces the dimensionality of images, reducing the number of learn-

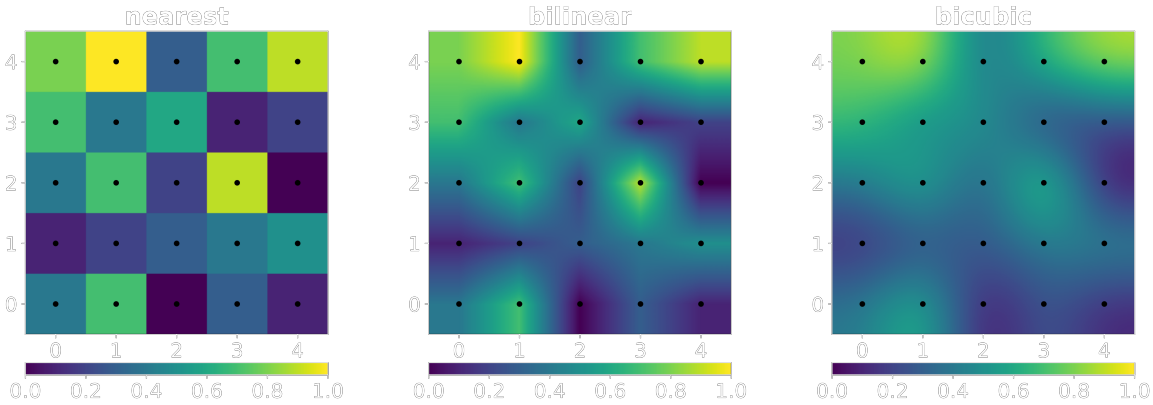

Figure 4.16 Nearest Neighbour (left), Bilinear (middle) and Bicubic (right) upscalling

able kernel values for a given deeper layer, thereby lowering computational costs as operating over these matrices requires operations with fewer values, while capturing essential encoded features, and a decoding layer, which reconstructs the original image from the latent representation.

These models learn to map input data to a continuous and low-dimensional latent space. From this latent space, it becomes possible to generate new samples that share similar characteristics with the training set. By adding information to this latent space, it is possible to manipulate the features of the generated samples, allowing for the addition or alteration of information.

The basic structure of an autoencoder consists of two main parts: the encoder and the decoder. The encoder receives the input and maps it to the latent space, through multiple convolutional layers and maxpooling operations. The decoder, in turn, takes the latent representation and attempts to reconstruct the original input using layers that may mirror those of the encoder as seen in Figure 4.17. The latent space that these architectures create becomes a condensed yet informative representation, that while they might not mean much for humans when visually observing their values, these have all the information needed for reconstruction. This capability has broad applications, ranging from image generation in computer vision to feature manipulation in generative models.

One of the challenges inherent in traditional CNN is the loss of localization context as the data progress through the network layers. While convolutions and pooling layers are effective at extracting fine-grained details, the sequential application of these operations can lead to a reduction in spatial information, hindering the network's ability to precisely localize features within the input data.

An innovative solution to address this challenge involves replacing traditional pooling operators with up-sampling operators [36]. This modification preserves the localization context while still enabling the extraction of high-level details. In the context of an encoder/decoder architecture, this approach entails an encoding block that utilizes convolutions and pooling to extract features in multiple spatial resolutions. And a decoder block that is tasked with reconstructing the image by employing upsampling and convolutions on the encoded latent space

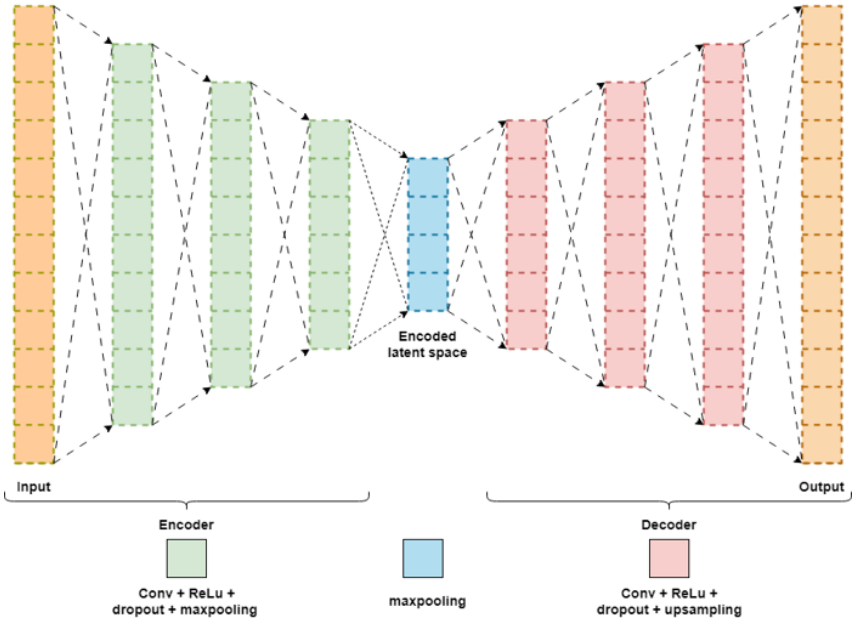

Figure 4.17 Autoencoder architecture

features to reconstruct the image or for the case of SAR to optical domain transformation alter the original latent space SAR image into features belonging to the optical image, making it capable of recreating the optical image by "hallucinating" the color and texture information of the SAR image.

This process is illustrated in the architecture diagram of our proposed system, as shown in Figure 4.18.

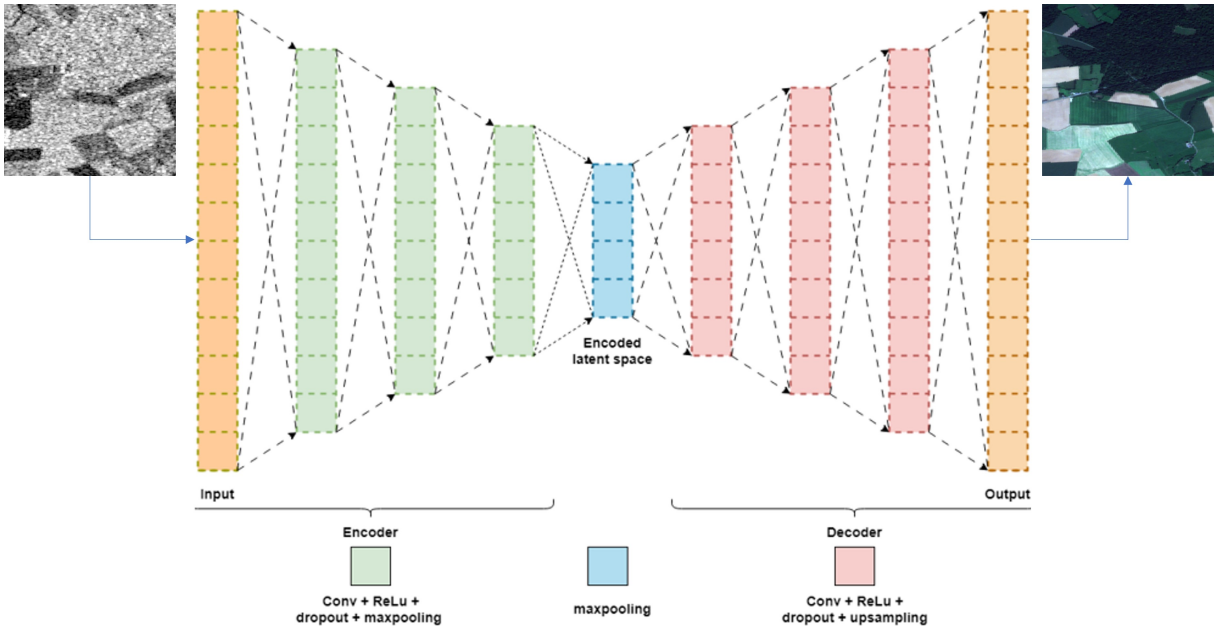

Figure 4.18 SAR2Optical architecture

However, even when applying upsampling operators in the decoding block, a lot of the context is still lost as the network is constructing the optical image only based on the SAR latent space, which is a compacted encoded image represented by its feature maps. This is a

problem that applies to many different types of neural networks, from Long short-term memory (LSTM)s to RNNs and CNNs, and so some techniques were created to combat this. One of these was the development of skip connections [37]. These are connections created between different layers in the network in order to facilitate the transmission of information through the network. This helps pass context in the form of feature maps to different layers, controlling the information that each layer uses to produce a more faithful result.

U-net [36, 37] popularized the application of skip connections in autoencoders for the purpose of biomedical image segmentation, where by connecting the encoding layer with the decoding layer of the same spatial resolution (i.e., same input and output dimension), the network is capable of always retaining a combination of not only the encoded latent space of the original image but the image itself through the decoding blocks. This allows the decoder to access a more contextualized view, allowing a more faithful reconstruction of the original input. These connections help address the challenge of information loss during the encoding process. By providing shortcut connections between layers, the network can effectively transmit detailed information from earlier stages to later stages of the decoding process without increasing the number of trainable parameters. This not only promotes a richer contextual understanding but also enables the decoder to produce reconstructions that better preserve the key spatial features present in the input data.

This technique is illustrated in Figure 4.19, which demonstrates the integration of skip connections in the SAR2Optical architecture.

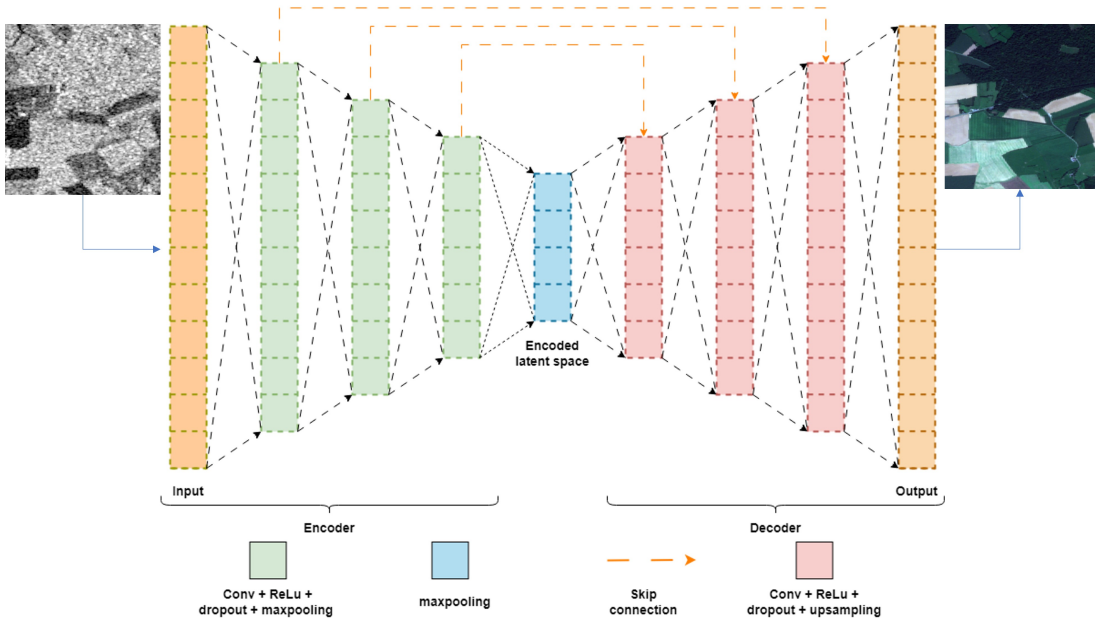

Figure 4.19 SAR2Optical architecture with skip connections

When creating neural network models, it's good practice to create benchmark models to which one can compare the current results. For this, the aim is to create the simplest model possible. Due to the impact of each hyperparameter when comparing results, the minimum number of changed hyperparameters should be used. For this, it was necessary to select what the benchmark model would be.

As for the activation function, the model uses Leaky ReLU for the encoding and decoding layers and a sigmoid activation function for the last layer to normalize the values from 0 to 1. Some testing revealed that the different activation functions did not visually change the results that the model produced, and due to the possibility of gradient vanishing/explosion, Leaky ReLU was chosen as the intermediary activation function.

The applied upsampling applied was bi-linear interpolation. The number of layers was fixed at 4 encoding layers, 1 layer representing the encoded latent space, and 4 decoding blocks. The number of feature maps generated per layer varied to test if the network was underfitting or overfitting. The learning rate varied but was usually between 0.001 and 0.005, with a decay rate of 0.92 per 1000 steps (multiplying the current learning rate by 0.92 every 1000 iterations). The batch size was constrained byVRAM limitations, meaning the model could not handle more than 32 images per batch when using the deeper models tested, so for comparison, the batch size was kept at 32 for all tested models.

During training, the network undergoes iterative adjustments to minimize the difference between the reconstructed image and the input image using various loss functions. The network was trained using MSE as a baseline metric due to its simplicity (see Equation 4.7). In addition to MSE, the network was also trained using SSIM, which captures not only color comparisons but also structural elements such as brightness and contrast (shown in Equation 4.10).

In addition to traditional loss functions, recent advancements in deep learning have led to the development of loss functions tailored to specific applications and data modalities. For instance, perceptual loss functions use pre-trained deep neural networks, such as VGG or ResNet, to measure perceptual similarities between images at higher semantic levels. By incorporating high-level features extracted from deep neural networks, perceptual loss functions offer improved perceptual quality and robustness to variations in low-level image characteristics. Similarly, adversarial loss functions, commonly used in Generative Adversarial Networks (GAN)s, introduce a discriminator network to distinguish between real and generated images, encouraging the generator network to produce more realistic outputs. These loss functions can complement the traditional techniques as they use different approaches to the training problem. By using a pre-trained network and fine-tuning it allows us to save resources as the costly part of training the model was already done, however these networks were not trained with the current problem in mind, and so they did not offer increased results. If these pre-trained networks are large, these might require more computational resources, when the objective is to fine-tune a pre-trained model. GANs are a good solution for cases where the dataset is not heavily populated, where by having a discriminatory network acting as a teacher it can generate more realistic images with a lesser dataset, yet these models also require a greater amount of processing power as there are two networks to train simultaneously, and so were not feasible to apply.

A common approach to programming problems, particularly in the domain of AI generative models, involves breaking down complex tasks into smaller, more manageable sub-problems.

One strategy approaches the problem by developing smaller, more specialized models that can individually address these sub-problems, leading to improved overall performance when combined. One possible approach to the current problem is to split it into smaller chunks. By splitting the RGB color channels of the optical image it is possible to create 3 different networks each specialized in a single channel (red, green, or blue). It is then possible to combine the resulting color channels produced by the 3 networks to create a single RGB image that represents the optical image. By splitting the networks, it now hase one channel as input and one as output, as illustrated in Figure 4.20.

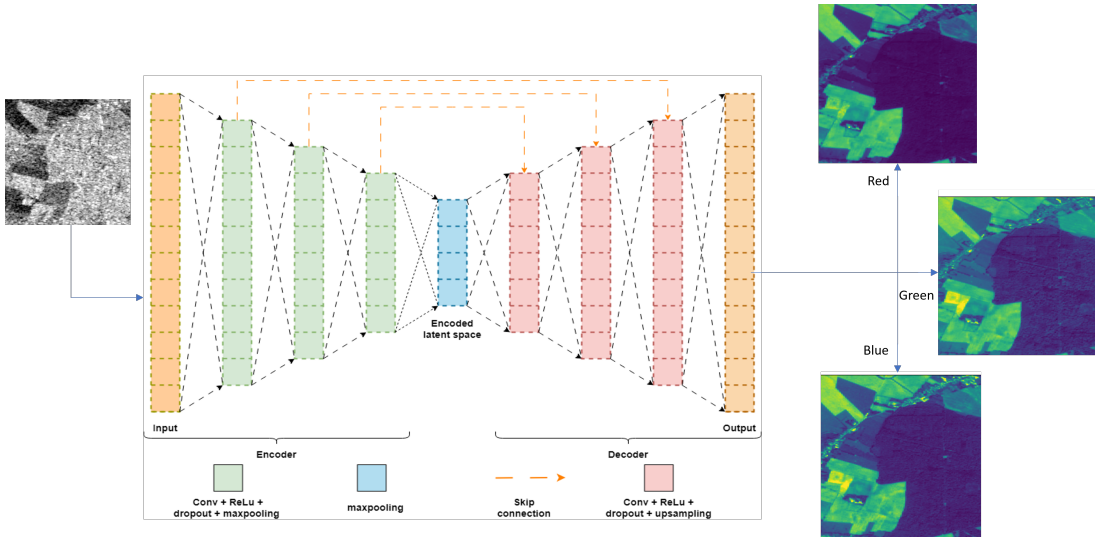

Figure 4.20 Splitting the RGB output into 3 seperate channels

Since three different networks are being used, one might get more overfitted than the others, resulting in some color deformations. To mitigate this a fourth network can be used, where it receives the resulting channels of each RGB network and fuses them together to homogenize the channels. With this approach, when the model receives a SAR image, the three specialized networks each generate the RGB channels, and once these are done, a fourth network receives the three color channels and converts them into a single image, as seen in Figure 4.21. This requires the training of an extra network, which might require aditional computational resources as it receives 3 input images instead of a single SAR image.

One potential solution for generating optical RGB images from grayscale SAR data involves dividing the task into two distinct steps. First, the texture and composition of the optical image can be generated, resulting in a grayscale image capturing the edges and features of the optical image extracted from the SAR image. This grayscale representation, often referred to as the luminance or brightness component, lacks any color information. In the second step, the color information is generated based on this extracted texture information. By decoupling the color information from the luminance, a two-step approach is established: first, the luminance aspect of the optical image is recreated using the SAR image as input, followed by the coloring of this recreated luminance image to match the colored optical image.

The problem is that for RGB images the luminance is derived from the color channels

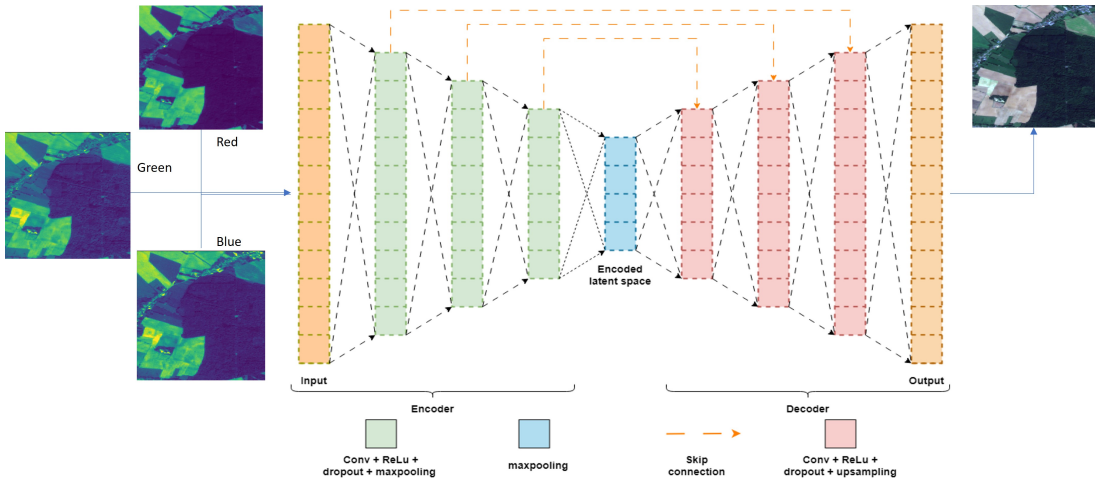

Figure 4.21 Combining the 3 seperate channels into a single RGB image

(Red, Green, Blue) using a weighted sum, where each channel contributes differently to the overall perceived brightness so the model can not extract the luminance component directly from the RGB image. Luminance in an RGB image can be calculated using a weighted sum of the individual color channels (Red, Green, and Blue). The weights are chosen to mimic the human eye's sensitivity to different colors. One common formula to calculate luminance from RGB values is the luminance formula is defined by Equation 4.18.

$$Y = 0.2126 \times R + 0.7152 \times G + 0.0722 \times B \tag{4.18}$$

Here, Y represents the luminance, while R, G, and B denote the intensities of the red, green, and blue channels, respectively. These coefficients (0.2126, 0.7152, and 0.0722) represent the relative contributions of each color channel to the perceived brightness, based on the sensitivity of the human eye to different wavelengths of light. The coefficients are chosen to reflect the eye's greater sensitivity to green light, followed by red and then blue.

The CIELAB color domain, also known as the Lab color space, offers a distinct advantage over the RGB color space by separating color information from luminance information. Similar to RGB, CIELAB comprises three channels: L, a, and b. However, in CIELAB, the L channel represents luminance, while the a and b channels encode color information. This separation facilitates independent manipulation of luminance and color components, enabling more precise control over image properties without affecting each other.

The L channel contains information about the perceived brightness of the image, ranging from 0 (black) to 100 (white). The a and b channels represent color information: the a channel encodes green to red hues, with negative values indicating green and positive values indicating red, while the b channel encodes blue to yellow hues, with negative values representing blue and positive values representing yellow. This orthogonal representation of color and luminance allows for easier adjustment of image characteristics, making CIELAB optimal for tasks such as image enhancement and color correction.

To convert a RGB image into a CIELAB color space image it must first be converted to the

XYZ color space, which is an intermediate space for color transformations that is a standard defined by the International Commission on Illumination (CIE). This transformation to the XYZ color space can be defined as by Equation 4.19.

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} R \\ G \\ B \end{bmatrix} \times \begin{bmatrix} M_{11} & M_{12} & M_{13} \\ M_{21} & M_{22} & M_{23} \\ M_{31} & M_{32} & M_{33} \end{bmatrix} \tag{4.19}$$

Where RGB are the real linear values, XYZ are the RGB equivalent in the XYZ color space and M is the transformation matrix. There have been many proposed transformation matrices, but CIE standards specified the values presented in Table 4.20.

$$\begin{bmatrix} 0.49 & 0.31 & 0.20 \\ 0.17 & 0.81 & 0.01 \\ 0.00 & 0.01 & 0.99 \end{bmatrix} \tag{4.20}$$

Once the values are in the XYZ color space, the next step is to transform these into the CIELAB color space, which better separates color information from brightness (luminance).

The transformation to CIELAB is done using a reference white point, which is typically defined as the brightest point in the image. This reference white point ($X_{\text{ref}}, Y_{\text{ref}}, Z_{\text{ref}}$) is used to normalize the XYZ values.

The luminance component L in CIELAB is calculated using Equation 4.21.

$$L = 116 \times f\left(\frac{Y}{Y_{\text{ref}}}\right) - 16 \tag{4.21}$$

In this equation, the luminance is based on the Y value from the XYZ color space and is scaled using the reference white luminance Y_{ref}. The constant 116 is applied to scale the result, and the subtraction of 16 is done to shift the luminance values so that they range from 0 to 100 for most real-world images.

The chromaticity components, a (obtained using Equation 4.22) and b (obtained using Equation 4.23), in the CIELAB color space are calculated by comparing the relative color differences of X, Y, and Z.

$$a = 500 \times \left[f\left(\frac{X}{X_{\text{ref}}}\right) - f\left(\frac{Y}{Y_{\text{ref}}}\right)\right] \tag{4.22}$$

$$b = 500 \times \left[f\left(\frac{Y}{Y_{\text{ref}}}\right) - f\left(\frac{Z}{Z_{\text{ref}}}\right)\right] \tag{4.23}$$

Here, a represents the color difference between the red/green channels, and b represents the color difference between the blue/yellow channels. The use of the function $f(t)$, detailed in Equation 4.24, ensures that the transformation is perceptually uniform, meaning that changes in lightness, color, or contrast are treated in a way that corresponds more closely to how the human eye perceives those changes. The function $f(t)$ applies a nonlinear transformation that adjusts the color space based on the magnitude of the value being transformed.

$$f(t) = \begin{cases} t^{1/3} & \text{if } t > \left(\frac{6}{29}\right)^3 \\ \frac{1}{3}\left(\frac{29}{6}\right)^2 t + \frac{4}{29} & \text{otherwise} \end{cases} \tag{4.24}$$

This function ensures that the transformation is smooth and continuous. When the value of t is large enough (i.e., above the threshold $\left(\frac{6}{29}\right)^3$), it applies a cubic root. Otherwise, it applies a linear transformation with a specific slope and offset. The goal of this function is to model the nonlinear behavior of human vision, which is more sensitive to small differences in dark areas and less sensitive to small differences in bright areas.

4.3 Dataset

Both SAR and optical images were obtained using Sentinel-1 and Sentinel-2 respectively. The Sentinel-1 is the European Radar Observatory for the Copernicus joint initiative of the European Commission (EC) and the European Space Agency (ESA). Copernicus is a European initiative for the implementation of information services dealing with environment and security. It is based on observation data received from Earth Observation satellites and ground-based information.

The Sentinel-1 SAR phased array antenna supports four imaging modes providing different resolution and coverage: Interferometric Wide Swath Mode (SAR) (IW), Extra Wide Swath Mode (SAR) (EW), StripMap Mode (SAR) (SM), and Wave Mode (SAR) (WV). All modes, except the WV mode can be operated in dual polarization. Both the IW and EW provide large swath width of 250 km at ground resolution of 5m x 20m and 400 km at ground resolution of 20m x 40m, respectively, as seen in Table 4.2 [38, 39].

Mode	**Incidence Angle**	**Chirp Bandwidth [Hz]**	**Single-Look Ground Resolution (rg x az) [m]**
SM	20-43	87.6-42.2	5×5
IW	30-42	87.6-42.2	5×20
EW	20-44	22.2-10.4	20×40
WV	23 & 36.5	74.5 & 48.2	5×5

Table 4.2 Key parameters of the Sentinel-1 SAR imaging

A single Sentinel-1 satellite can map the entire world every 12 days. With two satellites in the constellation, this cycle is reduced to 6 days. This means the satellites pass over the same spot on Earth every 6 days. The coverage frequency varies depending on location, with a repeat frequency of 3 days at the equator, less than 1 day at the Arctic, and 1-3 days over Europe, Canada, and major shipping routes, regardless of weather conditions. Radar data will be available to Copernicus services within an hour of being captured [38, 39].

As seen in Figure 4.22, the Sentinel-1 coverage map demonstrates the global reach and revisit capabilities of the satellite, enabling frequent observation over high-priority areas such

as Europe, polar regions, and maritime routes.

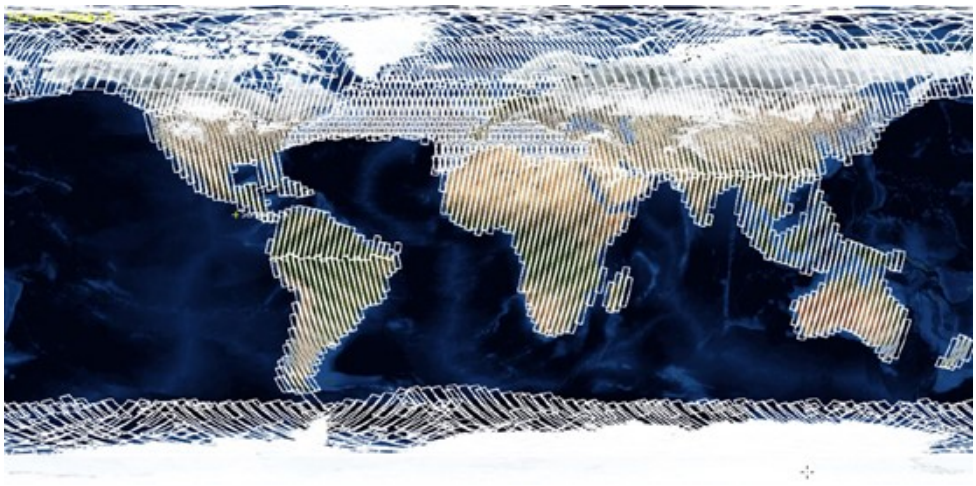

Figure 4.22 Sentinel-1 coverage map

Sentinel-1 offers a few data products, each with different levels of pre-processing. These are the SAR Level-0, Level-1 Single Look Complex (SAR data) (SLC), Level-1 Ground Range Detected (SAR data) (GRD), and Level-2 Ocean Component Network (OCN). Data products are available in single polarization (VV or HH) and dual polarization (VV+VH or HH+HV).

The SAR Level-0 products consist of compressed, unfocused SAR raw data. For the data to be usable, it needs to be decompressed and processed using a SAR processor. Level-0 products serve as the basis from which all other higher-level products are produced. Level-0 data includes noise, internal calibration, and echo source packets, as well as orbit and altitude information [40].

Level-1 data are the generally available products intended for most uses. Level-1 products are produced as SLC and GRD. The processing steps involved in producing Level-1 data products include pre-processing, Doppler centroid estimation, single-look complex focusing, and image and post-processing for the generation of the SLC and GRD products [40].

Level-1 SLC products consist of focused SAR data geo-referenced using orbit and altitude data from the satellite, and provided in zero-Doppler slant-range geometry. These products include a single look in each dimension using the full transmit signal bandwidth and consist of complex samples that preserve the phase information [40].

Level-1 Ground Range Detected (GRD) products consist of focused SAR data that has been detected, multi-looked, and projected to ground range using an Earth ellipsoid model. Unlike SLC, the phase information in GRD products is lost. The resulting product has approximately square spatial resolution pixels and square pixel spacing, with reduced speckle at the cost of worse spatial resolution [40].

As seen in Figure 4.23, SLC products retain higher resolution, while GRD products, despite losing phase information, offer a more noise-reduced representation with lower spatial resolution.

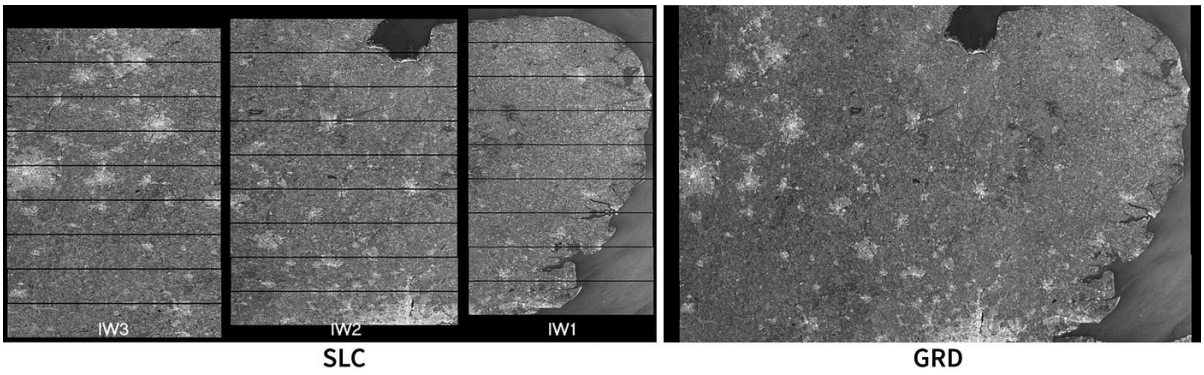

Figure 4.23 SENTINEL-1 SLC vs GRD

Level-2 OCN products include components for Ocean Swell Spectra (OSW), along with two new components: Ocean Wind Fields (OWI) and Surface Radial Velocities (RVL). The OWI component can be observed in Figure 4.24, while the OSW component is showcased in Figure 4.25

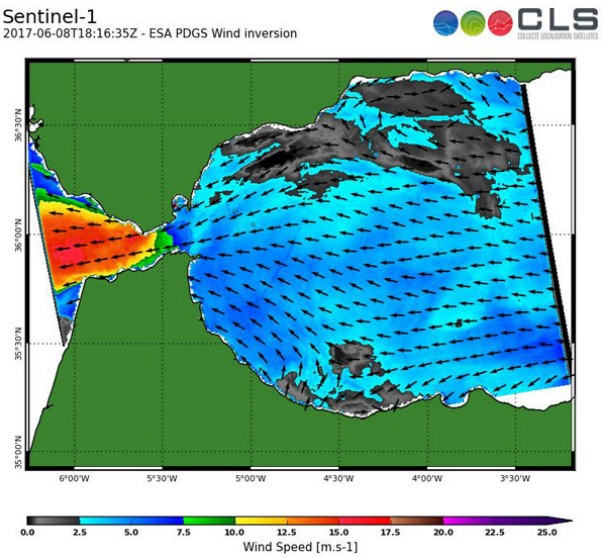

Figure 4.24 Surface Wind measurement (OWI): Example of SAR wind measurement over Gibraltar (funnelling effect) - Copernicus Sentinel data

The OSW is a two-dimensional ocean surface swell spectrum and includes an estimate of the wind speed and direction per swell spectrum. The OWI is a ground range gridded estimate of the surface wind speed and direction at 10 m above the surface derived from internally generated Level-1 GRD images.

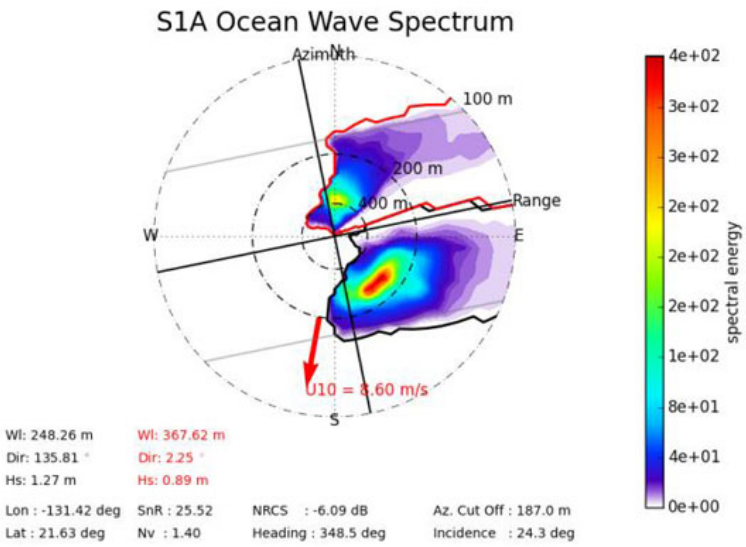

Figure 4.25 Swell inversion (OSW): Example of ocean wave spectrum presenting 2 swell partitions - Copernicus Sentinel data

The Copernicus SENTINEL-2 mission comprises a constellation of two polar-orbiting satellites placed in the same sun-synchronous orbit, phased at 180° to each other. It aims at monitoring variability in land surface conditions, and its wide swath width (290 km) and high revisit time (10 days at the equator with one satellite, and 5 days with 2 satellites under cloud-free conditions) will support monitoring of Earth's surface [41].

Sentinel-2 also offers data products in multiple levels of pre-processing. Sentinel-2 Level-0 data consists of raw unprocessed data acquired by the satellite's sensors. These data include imagery captured in multiple spectral bands and at different spatial resolutions. Level-0 data require further processing to correct for sensor artifacts and convert them into usable formats. Level-0 products also offer a 25 km across track by 23 km along track coverage area.

The Level-1B product provides radiometrically corrected imagery in Top-Of-Atmosphere radiance values and in sensor geometry. Additionally, this product includes the refined geometric model which is used to generate the Level-1C product. Like level-0 data Level-1 also cover the same area of 25 km by 23 km [42].

The Level-1C product is composed of 110x110 km2 tiles. Earth is subdivided on a predefined set of tiles and each tile has a surface of 110x110 km² in order to provide large overlap with the neighboring. The Level-1C product results from using a Digital Elevation Model (DEM) to project the image in cartographic geometry [42].

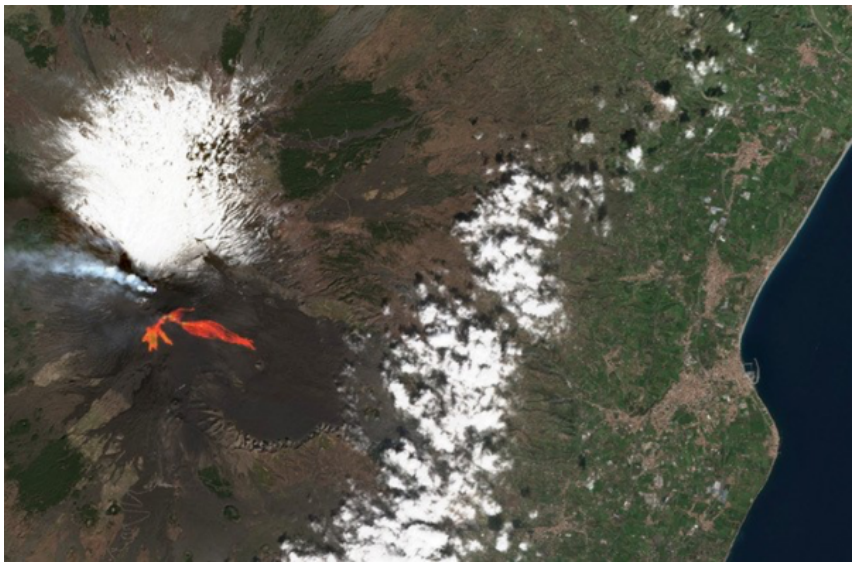

Figure 4.26 Extract of a Sentinel-2 Level-1C Product

The dataset used to train the neural network was obtained by Technische Universität München combined with manually obtained images and consists of aligned images of both SAR and optical systems [43]. It contains 4 classes of images: grassland, barrenland, agricultural, and urban. For each type of terrain, there are optical images in RGB format as well as synthetic aperture radar images. Each class contains 4000 images of size 250x250 pixels. The final set of the captured scenes can be observed in Figure 4.27. For Sentinel-1 the images were acquired in GRD mode using IW swath using VV polarization and at a pixel spacing of 5 m in azimuth and 20 m in range. For Sentinel-2 the images were acquired with a maximum cloud coverage of 1%. If no cloud-free Sentinel-2 image or no VV-IW Sentinel-1 image is available within the corresponding season, the data was discarded [43].

The process of acquiring the dataset images, with its flowchart presented in Figure 4.28, can be described in a few steps [43]:

- **Random ROI Sampling**: Some scenes were sampled globally using Google Earth Engine (GEE) function, ensuring representation across different regions.
- **Data Selection**: Filtering Sentinel-1/Sentinel-2 imagery for the year 2017, images were categorized into four meteorological seasons. The dataset was structured into four sub-groups based on seasonal ROIs, discarding ROIs with no cloud-free Sentinel-2 or VV-IW Sentinel-1 images.
- **Image Mosaicking**: Selected image data were mosaicked into single images for each ROI, creating RGB images from Sentinel-2 bands 4, 3, and 2.
- **Image Export**: Images were exported as GeoTiffs with a scale of 10m. Pre-processing included cutting gray values, scaling to $[0; 1]$ range, and contrast-stretching.
- **First Manual Inspection**: All downloaded scenes were visually inspected for issues such as large no-data areas, strong cloud coverage, and distorted colors.

Figure 4.27 Coverage map for the final set of scenes

- **Tiling**: Patches of 256×256 pixels were generated with a stride of 128 to minimize overlap between neighboring patches. This resulted in 298,790 Sentinel-1/Sentinel-2 patch-pairs.
- **Second Manual Inspection**: Another visual inspection removed suboptimal patches containing small clouds or mosaicking seamlines, ensuring dataset quality.
- **Data Augmentation**: Generating new training examples from existing data through transformations such as rotation, mirroring, amplification, cropping, or shifts. These transformations expand the training set and introduce variations in the data, improving the model's generalization ability by learning the same dataset with features of interest in different positions within the image.

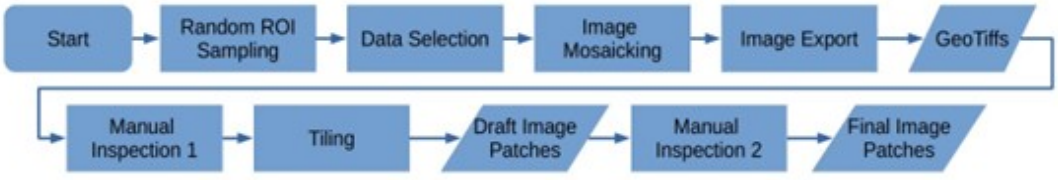

Figure 4.28 Flowchart of the semi-automatic, Google Earth Engine-based patch extraction procedure

Due to the nature of the topic, there are few open-source datasets available, and existing ones are often limited in size. To artificially increase the dataset size, data augmentation techniques can be employed.

A common data augmentation technique is to rotate the input image so that it is possible to generate multiple images from a single source. This way the network not only gets more images to train from, but it also learns to better generalize the input by receiving it in multiple angles. Shifting, flipping, and zooming are other techniques that can help the network to better generalize concepts. These techniques can be observed in the Figure 4.29. Brightness and contrast adjustment can also be employed, these techniques include histogram equalization, data normalization and gamma correction. These methods help in adjusting the overall brightness and contrast levels of images, ensuring consistent illumination across the dataset, improving the model's ability to learn from varied lighting conditions. Yet these may not always yield better results as by transforming the original values it might make us lose information. Adding noise to the image can also help it better generalize and make it more robust when dealing with noisy images.

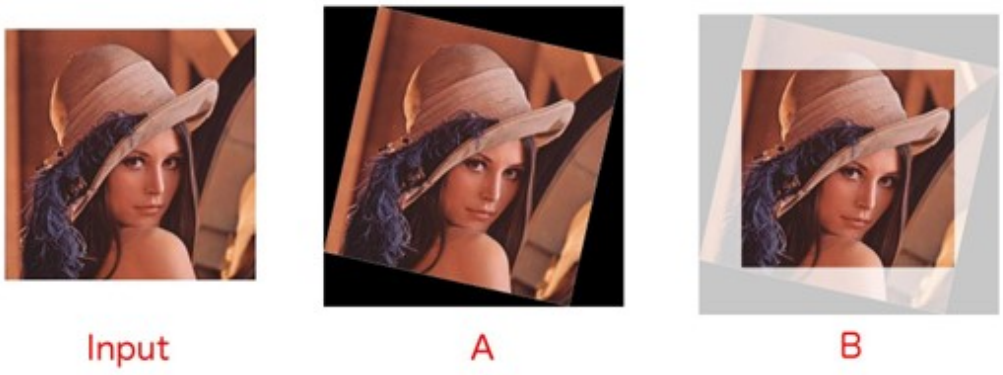

Figure 4.29 Data augmentation techniques

Data normalization is also a relevant technique, transforming the data to a scale between 0 and 1. This approach works by ensuring that each feature contributes equally to the learning process, regardless of its original scale. When input and output images have different ranges, the optimization algorithms used to train models, such as gradient descent, can be influenced disproportionately by features with larger values. For instance, if one input variable has a significantly higher range than others, it can dominate the error signal, leading to unbalanced updates to model parameters. This can hinder learning, resulting in suboptimal model performance.

Normalization helps mitigate this issue by scaling all inputs to a consistent range, which makes the optimization process more stable and ensures more consistent convergence. When the inputs are normalized, the optimization algorithm doesn't have to deal with variables of vastly different magnitudes, which improves the efficiency of parameter updates and makes it easier for the model to reach convergence. By keeping the input data values smaller and consistent, normalization reduces the risk of such instability.

Additionally, normalization can also improve the performance of activation functions, particularly in deep neural networks. Many activation functions, such as the sigmoid and tanh, are sensitive to the input value range. If inputs are not normalized, they may fall into regions where the activation function has very low gradients—essentially saturating the function and leading

to vanishing gradients [30], where the model stops learning effectively. Normalizing the data ensures that inputs to these activation functions are distributed in a range where gradients are more useful, thereby helping the model to learn better representations.

Another common normalization technique is mean subtraction and standard deviation division. This technique involves subtracting the mean value from each data point and dividing by the standard deviation. It works by transforming the data to have a mean of zero and a standard deviation of one, which ensures that the data is centered and scaled appropriately. When the data is centered around zero, optimization becomes more efficient because it helps the model adjust weights in a balanced manner—meaning positive and negative updates to the parameters are not inherently biased by an off-centered dataset. Centering the data also helps prevent the introduction of bias in the gradients, which could otherwise lead the model to favor certain parameter updates over others.

Dividing by the standard deviation, on the other hand, scales the data to have unit variance, ensuring that all features are treated equally during training. This is particularly important in optimization algorithms such as gradient descent, where the magnitude of updates is proportional to the value of the gradients. By having features that all have a similar range and variance, the gradients will also have similar magnitudes, leading to more consistent parameter updates across different features. This consistency helps the model converge faster and reach a solution that generalizes well to unseen data.

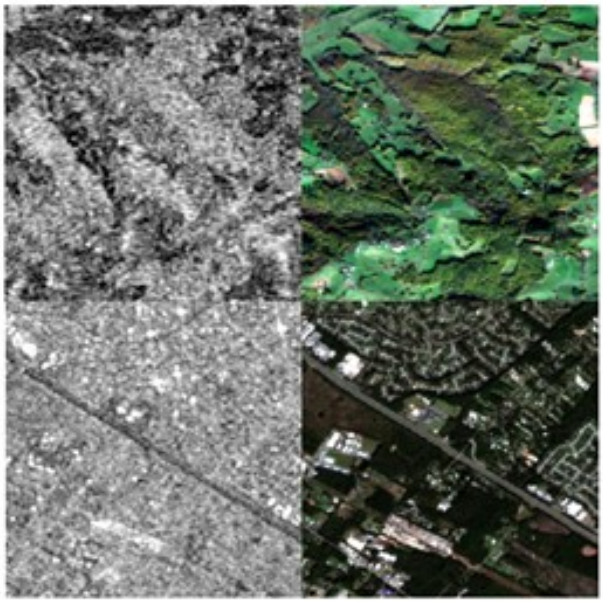

Figure 4.30 Examples of SAR images and their optical counterpart

4.4 Software and tools

When developing a project some care must be put into choosing the available tools, weighting each of its advantages and the best use cases for the given technology and what might be their constrains, either due to scalability problems or hard limitations, when compared to others. The

following chapter goes over the software and tools used, what other alternatives there were and where it was used in the project.

4.4.1 SNAP

Due to the scarcity of the data surrounding the topic the dataset can be complemented by manual extraction of SAR Sentine-1 and optical sentinel-2 data to augment the dataset. Copernicus open access hub offers, either through a Graphical User Interface (GUI) or by Application Programming Interface (API) requests, both Sentinel-1 and 2 data without charge. For Sentinel-1 it offers level 0, 1 and 2 images and for level 1 in both SLC and GRD. Due to the nature of the project level-1 images were chosen as they provide the necessary level of information without the overhead of level 2 images and without the need of extra processing that a level 0 images would need. As for SLC and GRD the main concerns are if phase information is needed and if there is a need to obtain less processed amplitude images. Phase information can be used, and it could help the network better generalize situations where moving objects are present, where due to the object shifting there might be situations where for example a ship is not correctly aligned between sentinel-1 and 2 images which will degrade the colorization of the image. In a later chapter this is expanded as this is a limitation to the network. Another aspect to consider is the file size, while SLC images offer phase information, they also have a considerable size before processing of around 10GB per image. After processing it is possible to reduce the produced image size, yet a sizable dataset of pre-processed SLC images requires a considerable amount of storage space. On the other hand, by transforming the SAR image into a real image, essentially removing the phase dimension, it is possible to reduce the GRD image size to around 2GB before processing.

SAR images inherently suffers from noise, distortions and other anomalies and so some extra processing is needed to improve the image quality. There are a few commercially available tools for SAR image processing, namely the ESA SNAP, which is free of cost. It was developed by the European space agency and offers both sentinel 1 and 2 toolboxes. The PolSARpro,was also created by the European Space Agency but is best suited for polarimetric SAR data, which was not the aim of the project so was discarded. Gamma and NV5 SAR processing software also offer a wide variety of tools but due to their cost were also discarded. Because of this SNAP desktop was chosen, some alternatives also included some python and MATLAB frameworks, where the heavylifting done by ESA SNAP's would require manual implementation of SAR image processing. This would be increasing the complexity of the project without the need for it so it was also discarded for the SAR image processing (data augmentation techniques still need to be implemented using one of these tools, as SNAP does not offer those features).

SNAP offers a sentinel-1 toolkit with which it is possible to use to process the SAR images to denoise, filter and transform the images. The first step is to calibrate the image. When sentinel-1 SLC SAR images are provided these are raw radar reflectivity values as captured by the sensor, where each pixel has both the real and imaginary components yet also contain interference caused by the sensor's characteristics, such as the sensor's sensitivity, antenna

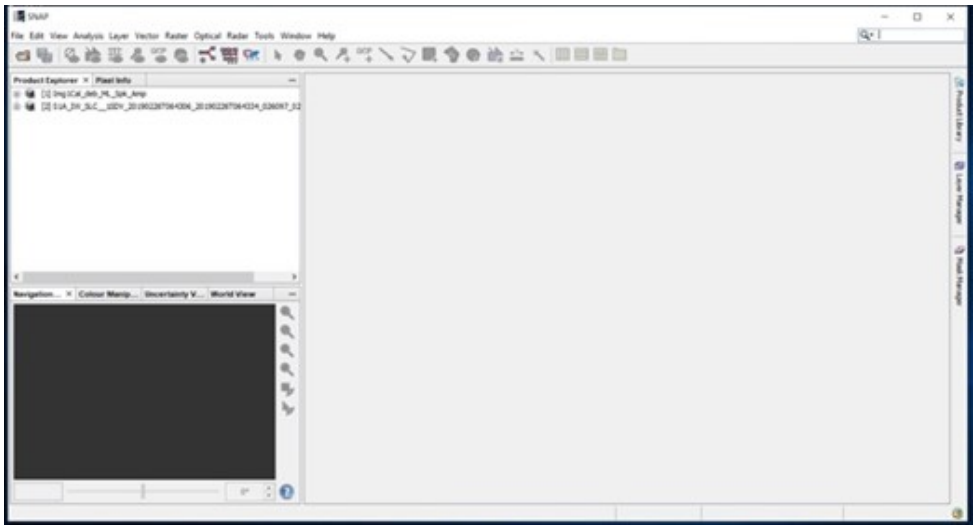

Figure 4.31 SNAP desktop GUI

patterns or other sensor biases. The Figure 4.32 shows the result of image calibration.

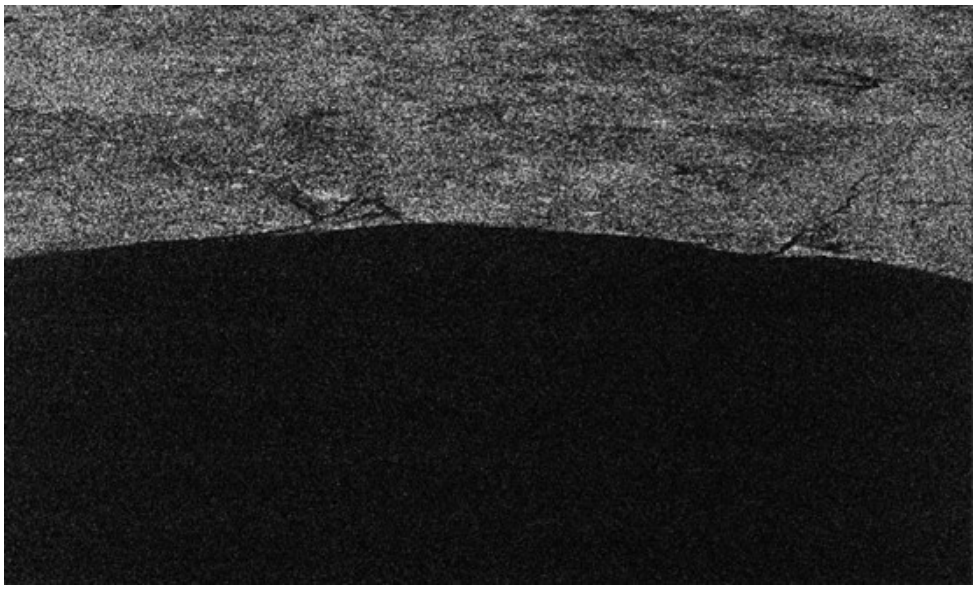

Figure 4.32 Sentinel-1 SLC calibrated image

Speckle noise is an inherent characteristic of SAR images, arising from the coherent processing of radar signals [44, 45]. As the radar signals reflect off the Earth's surface, they scatter in multiple directions. Each radar resolution cell captures the combined backscatter from multiple objects within its area. The coherent nature of the radar means that all reflected signals retain their phase information, leading to constructive and destructive interference among them. This interference results in variations in the amplitude of the received signals within each cell. The multipath reflections, where signals bounce off multiple objects before reaching the sensor, can increase these phase differences, further contributing to speckle noise. As a result, the SAR images display a granular texture, with each pixel representing the integrated effects of these interactions. There have been multiple approaches to speckle noise reduction [44, 45],

the simplest approach one could have is to replace each pixel values with the average or median of its surrounding neighbors pixels. The mean box filter can be implemented by applying a linear smoothing filter which is a kernel composed of '1' (see Table 4.3). As this filter applies an average to the image this results in the loss of edge information which can deteriorate the image shapness, due to edges becoming more blurry, with the benefits of reducing the image noise, as see in Figure 4.33.

1	1	1
1	1	1
1	1	1

Table 4.3 Mean box filter kernel

Figure 4.33 Average filter

The Lee Filter [44, 45] was developed specifically for speckle reduction and is an adaptive filter that preserves details and edges by varying the amount of smoothing applied based on the local variance within the image. If an area has high variance, it is likely to be an edge or detail, and so less smoothing is applied. The Frost filter [Frost filter] is similar to the Lee filter, as it is also an adaptive filter that varies the kernel coefficients based on the local variance. It uses an exponential decay function to adjust the degree of filtering, which helps in better preserving important spatial details. When uniform regions are filtered the Frost filter acts as a mean filter. When high contrast regions are filtered, the filter acts as a high-pass filter with rapid decay of elements away from the filter center. Thus, large uniform areas will tend to be smoothed out and speckle removed, whilst high contrast edges and other objects will retain their signal values and not be smoothed [44, 45]. The results of this filter are shown in Figure 4.36

The mentioned techniques are called spatial filters as they use spatial information about the pixels to remove the speckle noise. It is possible to also use domain transformation filters to extract different domains of the image and apply filters only to these domains. The Fourier

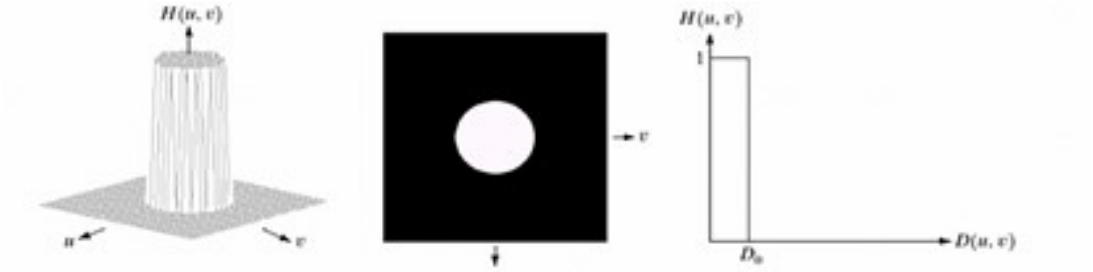

Figure 4.34 Low pass filter for the Fourier transform

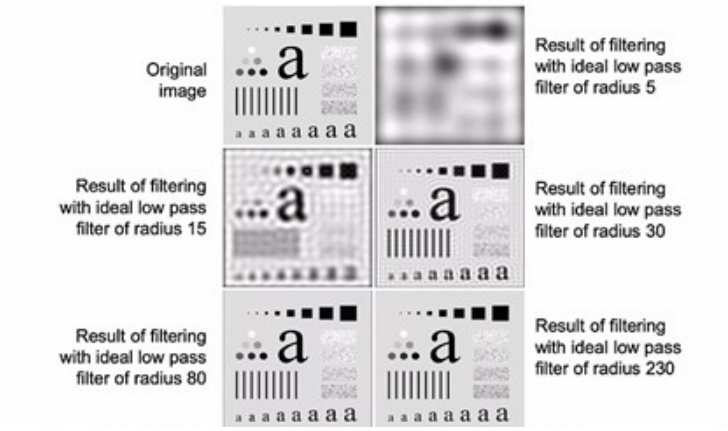

Figure 4.35 Low pass filters with varying sizes

transform is a domain transformation that decomposes the signal into its sine and cosine components. The resulting image after the Fourier transform is the frequency image, where each pixel represents a single frequency in the original spatial image. Values centered around the axis of the image represent the lower frequencies of the signal and further pixels higher frequencies. The average box filter can also be implemented for the Fourier domain by creating a low-pass filter that when applied to the frequency image will cut off the values centered around the axis that when converted back to the spatial domain will smooth the image.

Using Lee Sigma with 3 number of looks, a window size of 7x7, the filter is able to remove additional noise . It is particularly observable in the ocean regions, where the mean filter lacked. This comes at the expense of more processing required and more parameters to fine-tune. Refined Lee is an improvement to the Lee Sigma algorithm that requires no parameters, where a K-Nearest Neighbour (KNN) algorithm is employed to adjust the number of neighbour pixels used within the sliding window [44, 45]. This further improves the speckle noise reduction, removing outliers and noise as to increase the network performance. Frost offers similar results as the refined lee, making it preferential and situational. The results of refined Lee and Frost can be observed in Figure 4.37 and Figure 4.38, where it is possible to notice that while the overall noise in the image was reduced, it was less impactful than the noise reduction done by Lee Sigma . However in both refined Lee and Frost the advantages is that the edge information was less downsampled, retaining a lot of its information, translated into higher pixel values and more contrast.

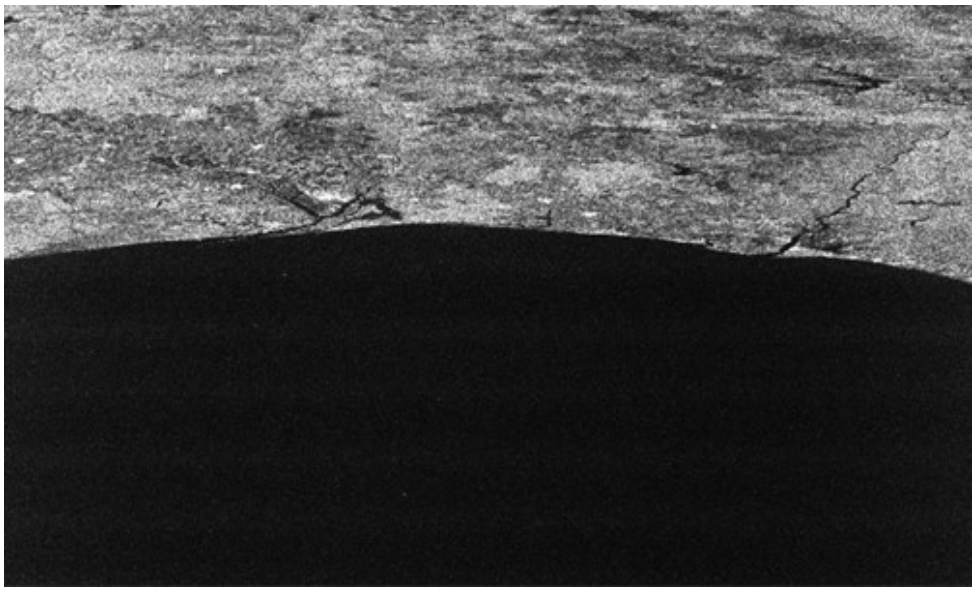

Figure 4.36 Lee Sigma

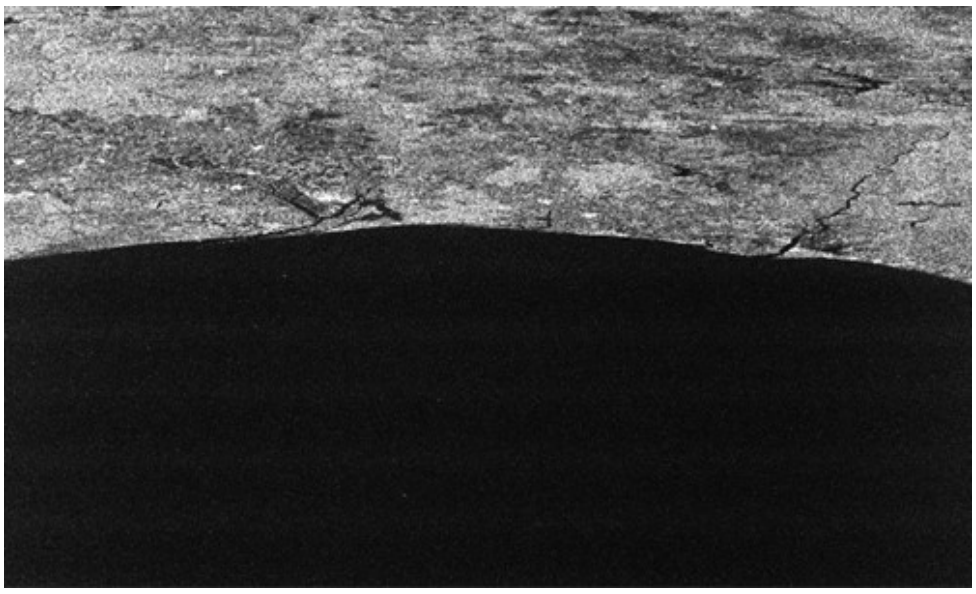

Figure 4.37 Refined Lee

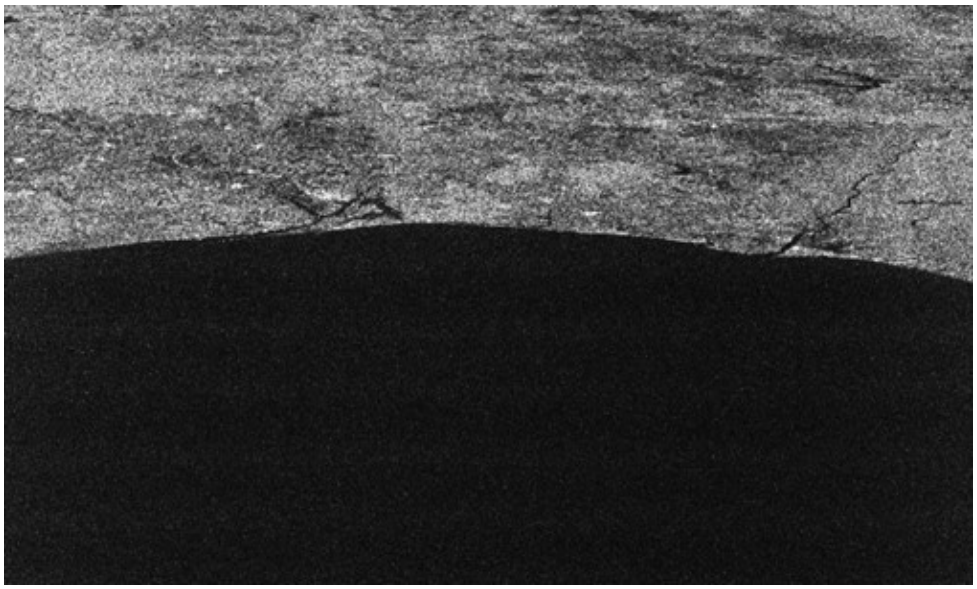

Figure 4.38 Frost

Speckle filtering is helpful in removing the random noise introduced by interference, yet it is not able to remove all the noise present in an image. A common approach is to then apply multilook to the image. Multilooking involves averaging multiple independent images of the same area, each obtained from slightly different viewing angles. These multiple images are called "looks" and are be performed in both the range and azimuth directions. The range multilooking involves averaging the radar returns received at slightly different times as the radar pulse travels back and forth. By splitting the radar pulse into shorter segments and averaging them, the speckle effect can be reduced. The azimuth multilooking takes advantage of the motion of the SAR platform (satellite or aircraft). As the platform moves, it captures new data from slightly different angles. By averaging these data points, the noise can be reduced. This technique also has the drawback of decreasing the resolution of the image, which will result in the loss of edge information. Multilooking can be applied to each dimension (range and azimuth) independently but by applying more looks the image will get blurrier and lose edge information. By comparing the result (present in Figure 4.39), of 3 looks in the range dimension and 1 on the azimuth direction with 10 looks in the range dimension and 3 looks in the azimuth it is possible to observe that the speckle noise impact on the image is lower at the cost of loss of information.

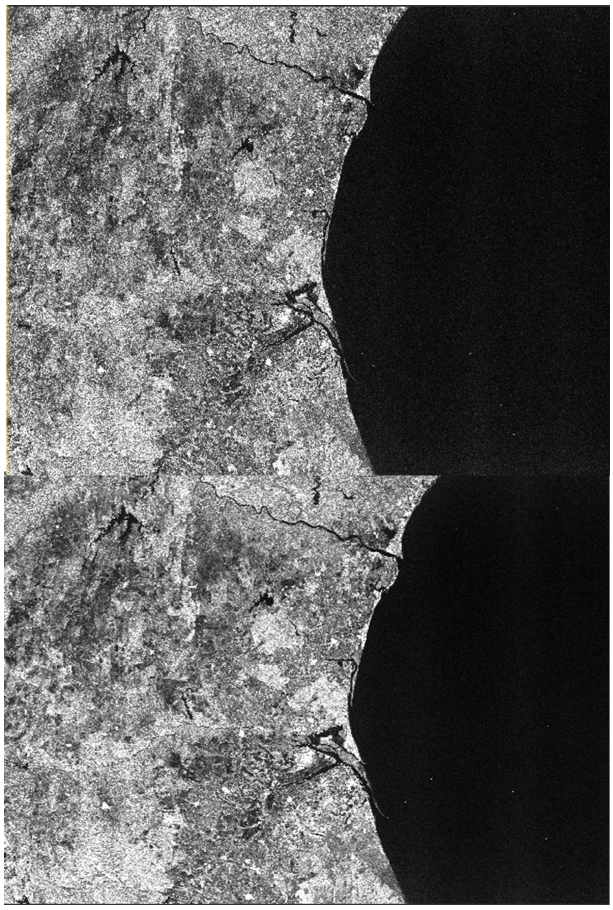

Figure 4.39 Multilooking with 3 by 1 looks (above) and 10 by 3 looks (below) for the range and azimuth dimension respectively

Multilooking can result in images with different terrain orientation. This is due to the fact that SAR systems capture data from multiple look angles to create a single image. SAR systems inherently suffer from speckle noise, which is a granular noise that arises due to the coherent nature of the radar signal. To mitigate this, multilooking is often applied, where multiple looks or acquisitions of the same area are averaged to reduce the speckle noise and enhance the interpretability of the image.

During multilooking, these multiple look angles, each capturing the terrain from a slightly different perspective, are combined into a single averaged image. While this averaging process is effective at reducing noise, it can also lead to discrepancies in the appearance of terrain features. If the angles are significantly different, the resulting image may exhibit shifts in the apparent direction or orientation of certain features. This effect is more prominent in areas with pronounced topography, where variations in incidence angle can cause features like slopes, valleys, or ridges to be perceived differently across different looks.

Because of these shifts, there is a need for terrain correction to ensure that the images are accurately aligned. Terrain correction involves correcting the geometric distortions introduced by varying look angles and terrain relief, so that the final image has consistent geolocation and accurately represents the terrain.

SAR images also acquired images using side-looking geometry, which means that areas

closer to the sensor appear compressed compared to areas further away, causing geometric distortions like foreshortening, layover, and shadowing as previously mentioned. By performing terrain correction, such distortions can be mitigated, ensuring that features on the ground are represented accurately and consistently, irrespective of the viewing geometry. For this SNAP offers functions that automatically apply terrain correction and overlays it over an optical image, which can be seen in Figure 4.40. This correction typically uses a DEM to account for the variations in terrain height and adjust the image geometry accordingly.

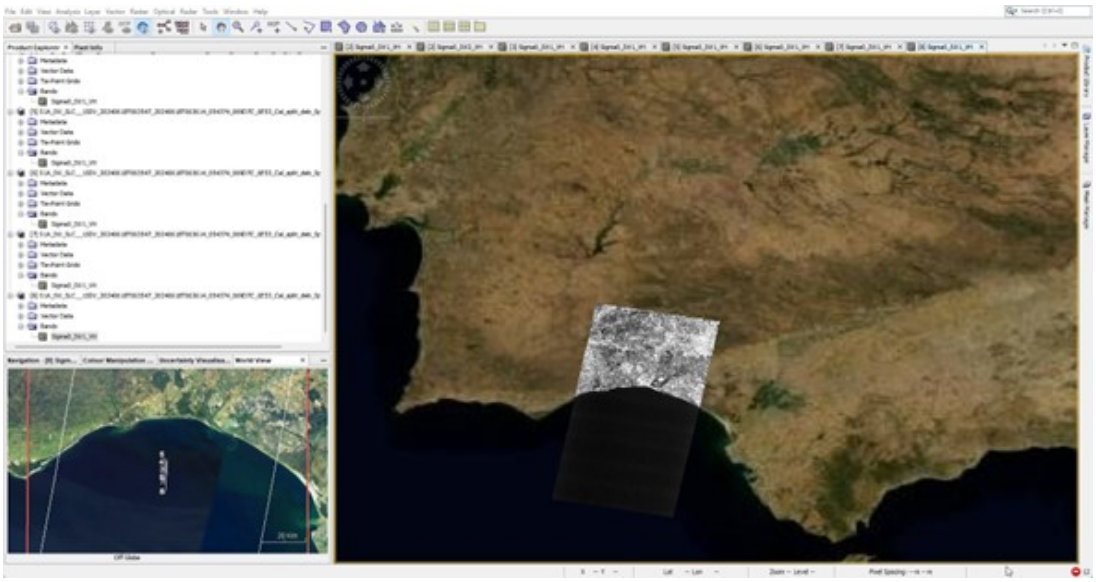

Figure 4.40 Terrain correction of SAR image overlayed over an optical image of the scene

After this is done it is possible to extract sub swaths of the area of interest. This is done to decrease the size of the resulting image as SAR images can reach 15K pixels by 15K pixels or bigger. At the current scale no network can ingest inputs with such sizes of subsets of the SAR image were created with resolution 256x256. The final results of the pre-processing can be observed in Figure 4.41, where it shows the raw level-1 SLC image to the left, and the post-processed SAR image, ready to be added to the dataset.

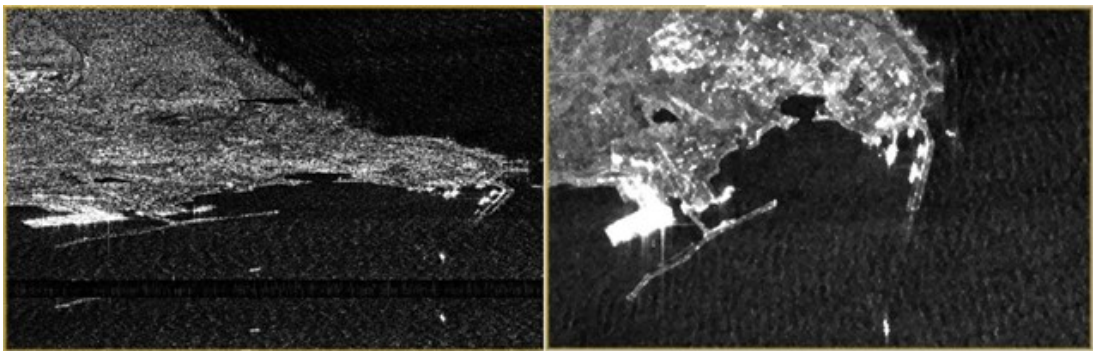

Figure 4.41 Pre-processed image of Sines harbor (left) and post-processing of SAR image using SNAP (right)

4.4.2 MATLAB

There are many programing languages with many frameworks focused on AI models. MATLAB is a licensed programming tools designed for quick matrix calculations. This feature is critical because, at their core, AI models, especially deep learning networks, rely heavily on matrix multiplications. The efficiency of MATLAB in handling such calculations makes it a compelling option for developers working with complex, deep models that are composed of many layers.

The performance of AI models during training is often constrained by factors such as the availability of VRAM and the speed of each training iteration. A language that can execute these operations faster can significantly reduce the time it takes to train models, thus enhancing the overall efficiency of the development cycle.

MATLAB also offers a comprehensive toolbox kit specifically designed for AI development. These toolboxes are integrated into the MATLAB environment, reducing the complexity and "entropy" in setting up and managing AI projects. This integration facilitates a smoother workflow, as developers have access to advanced algorithms and functions without the need for additional installations or configurations.

However, while MATLAB provides powerful tools for AI development, it also presents certain limitations. Being a licensed product, MATLAB requires a financial investment, which is a barrier. Additionally, while many basic features are available, some specialized AI functionalities are gated behind paid toolboxes. This pricing model can make MATLAB less attractive compared to open-source alternatives that offer extensive libraries and frameworks for free, such as Python with TensorFlow or PyTorch frameworks.

4.4.3 Python

Python is an open-source, high-level interpreted programming language known for its simplicity and readability. Created by Guido van Rossum in 1991, Python has since become one of the most popular programming languages. As an interpreted language, Python executes code line-by-line, facilitating quick testing and debugging. It is dynamically typed, meaning variables do not require explicit declaration to reserve memory space. Python holds an extensive standard library that supports many common programming tasks, while offering a vast collection of third-party modules and libraries for all types of tasks. Being an open-source project also facilitates in the creation and maintenance of new tools and the increased community helps when trying to look for possible solutions to bugs.

Python has also become a standard in AI development due to its collection of libraries and frameworks specifically designed for AI and machine learning. Libraries such as TensorFlow and PyTorch provide powerful tools for building and training neural networks. These libraries simplify the implementation of deep learning models, making it easier to develop advanced AI applications. Additionally, scikit-learn offers a wide range of algorithms for traditional machine learning tasks, from classification and regression to clustering and dimensionality reduction. Python's strong integration with data analysis and manipulation libraries, such as pandas and

NumPy, further enhances its utility in AI development. These libraries allow developers to efficiently handle large datasets, perform complex data transformations, and conduct data analysis, which are crucial steps in AI development. The combination of these tools streamlines the entire process of preparing data for machine learning models. Python's use in AI development extends across various domains, including Natural Language Processing (NLP), computer vision, robotics, and predictive analytics. For instance, libraries such as NLTK and spaCy are tailored for NLP tasks, while OpenCV and imageio are popular for computer vision applications. OpenCV is a library for computer vision, providing tools and functions for applications such as image and video analysis, face detection, object tracking, and feature extraction.

For this thesis the main frameworks used were TensorFlow for creating and training models as it offers a more direct implementation of the models which can speed up the training process. Instead of creating each of the many functions needed to train a model such as the loss functions, optimizer, etc. Scikit-learn was used to create an abstraction layer and to facilitate the hyperparameter selection. OpenCV was used for pre-processing task, in conjunction with numpy for array manipulation.

4.5 Hardware

To train a neural network, there are two primary hardware options: Central Processing Unit (CPU) and Graphics Processing Unit (GPU). Each type of hardware has distinct advantages and limitations, influencing the efficiency and effectiveness of neural network training. Understanding these differences can help in optimizing the training process and achieving the best performance from neural network models. When training neural networks, computing power is often the bottleneck that hinders the model's ability to learn and generalize patterns effectively. This computing power can be split into three main limiting factors: the amount of Random Access Memory (RAM), the clock speed of the hardware (whether CPU or GPU), and the amount of VRAM available on the GPU.

RAM plays a crucial role in the data handling process during neural network training. When loading a dataset the best practice is to apply preprocessing steps, this data is then stored inRAM. Sufficient RAM allows the entire dataset or large portions of it to be loaded simultaneously, facilitating faster and more efficient preprocessing and training. If theRAM is insufficient, data must be processed in smaller batches, which can significantly slow down the training process. This is particularly important when preprocessing steps depend on the statistics of the entire dataset, such as normalization, standardization, or data augmentation. Large datasets, especially in fields like image recognition such as this thesis or for other applications as natural language processing, require substantial memory for efficient handling. Insufficient RAM can lead to frequent data swapping between RAM and disk storage, which introduces latency and slows down the entire training process.

CPUs contain multiple cores, which can be a combination of physical and virtual cores. Modern CPUs typically have between 4 and 18 physical cores, with additional virtual cores

made possible through simultaneous multithreading or hyper-threading. Each core is capable of executing instructions from software applications and handles the general-purpose processing needs of a computer system.

Virtual cores enhance parallel processing capabilities by splitting a single physical core into multiple threads. These threads can share the computational load, allowing for more efficient multitasking. These virtual cores are possible due to simultaneous multithreading or hyper-threading, to split a core into virtual cores. When hyper-threading is applied, a single core maintains two separate execution contexts (threads). While one thread waits for data from memory, the other can utilize the CPU's execution units, improving overall efficiency. However, this advantage diminishes when the physical core is fully utilized, as the additional threads cannot exceed the core's total capacity. While virtual cores help optimize the speed of running applications, they can become a bottleneck for training neural networks. If a physical core is being utilized to its full capacity, it cannot effectively support additional virtual cores. Therefore, the limiting factors for training neural networks on a CPU include the number of physical cores and their clock speed. Although higher clock speeds can reduce training time, it affects only the training time as it will still learn the same patterns, but it is going to take longer to train/predict, meaning that they do not affect the final results of the model. This training latency can be a critical consideration for applications requiring real-time processing. CPUs can be effective for smaller models or simpler neural networks where the computational demands are lower. They are also well-suited for running the final trained model in production environments.

GPUs are specialized hardware designed to accelerate the processing of images and videos. They consist of thousands of smaller, simpler, and more efficient cores designed to handle multiple mathematical operations simultaneously. Unlike CPUs, which are optimized for sequential processing, where the CPU performs small and quick operations of multiple applications sequentially, GPUs are optimized for massive parallel processing and are usually optimized for quick matrix operations. This architectural difference allows GPUs to perform many operations concurrently, making them ideal for tasks that can be broken down into smaller, independent operations. Therefore, GPUs are particularly well-suited for training neural networks because the multiple gradient and weight calculations required during backpropagation can be distributed across the many cores of the GPU. This parallel processing capability significantly speeds up training times compared to CPUs. The main limiting factors for GPUs in neural network training are the clock speed and the amount of VRAM.

Similarly to CPUs, the clock speed of aGPU affects how quickly it can perform calculations. However, the clock speed does not impact the accuracy or results of the model, it only influences the duration of the training process. VRAM, on the other hand, is a critical factor that can directly affect the model's performance. VRAM limits the size of the batch that can be processed simultaneously and the size/complexity of the model itself as the model needs to be loaded into theGPU, where is stored in the VRAM. Insufficient VRAM can lead to smaller batch sizes, which means the model has fewer examples to learn from in each iteration, and so the impact of each of the data inputs will be greater. As an example, if training a model on images

with a batch of 4 images the network will give greater attention to those 4 images, so if one of the images is poisoned it will degrade the network more than if it was using a batch size of 100 images, where the network will learn more general patterns found in those 100 images and so the poisoned image will have less of an impact.. Smaller batch sizes can result in less stable training and a greater impact from individual data points, which can be problematic if any of the data points are noisy or corrupted.

For this thesis the model was trained on two different hardware, a weaker computer with a Intel Core i7-8750H CPU with a clock speed of 2.20 GHz and 6 physical and 6 virtual cores with a max clock of 4.10 GHz, and a NVIDIA GeForce RTX 2060 GPU with a clock speed of 1365 MHz and 6 GB of VRAM, reaching a maximum of 1680 MHz clock speed. It also contained 16GB DDR4 GB of RAM with a clock speed of 2400 MHz, which was enough to load the entire dataset. For this weaker computer the training process was much slower with around 5 minutes per iteration when trained on the CPU and around 1m30 per iteration when using the GPU. The model was then trained on a more powerful hardware, where a i7-13700K CPU was available with a clock speed of 3.40 GHz and 8 physical and 8 virtual cores with a maximum of 5.40 GHz clock speed. This hardware also possessed a RTX3060 GPU with a clock speed of 1320 MHz and 12 GB of VRAM, reaching a maximum of 1777 MHz clock speed. It also contained 32 GB of RAM with a clock speed of 4800MHz, which was enough to load the entire dataset. When training on this hardware using the CPU the average time per iteration decreased from 5 minutes to around 2 minutes, and when using the GPU, the average timer per iteration decreased from 1 minute to around 30 seconds. This shows how helpful having a powerful hardware can change the training process, where, considering that the model needs 100 iterations to fully be trained, for the weaker hardware using the CPU it would take around 8 hours, and for the GPU it would take around 2.5 hours. Meanwhile for the more powerful hardware, using the CPU, it would only take about 3 hours when compared to the 8 hours of the older CPU, and using the GPU would further decrease the training time to around 0.8 hours.

Chapter 5

Results

When optimizing a network, several parameters come into play, each of which can significantly influence the outcome. The process of selecting these parameters often involves a combination of systematic testing and trial and error. Some parameters, such as the kernel size, directly impact the number of trainable parameters within the network. In contrast, others can enhance the network's accuracy without altering the count of trainable parameters. To effectively observe the impact of any adjustments made during optimization, we established a baseline for comparison.

For this baseline we used a kernel size of 3x3. This allows the network to see finer details when extracting features, yet it increases the number of parameters. The chosen optimizer was the exponential decay with a learning rate of 0.001, a decay step of 1000 and a decay rate of 0.92. This can be adjusted based on the evolution of the loss when it shows signs of overfitting/underfitting. The chosen loss function was the mean square error as it is the simplest and gives us a direct comparison of pixel values. This baseline network used 9 convolutional layers, the first being an input layer, followed by 4 encoder layers with 32, 24, 16 and 8 kernel maps using a kernel size of 3x3. To subsample the image, we applied max pooling after each convolution. The max pooling after the last encoding layer represents the latent space and has a dimensionality of (32,32,8), meaning we obtain 8 feature maps with sizes 32x32. The decoder is the mirrored version of the encoder, using the same number of kernel maps per layer, but instead of max pooling it uses the upsampling as this allows us to obtain finer details while still maintaining contextual information. The final decoding layer generates 3 feature maps that each correspond to a channel of the RGB Optical image. For this first approach we opted to transform a 1 channel SAR image into a 3 channel RGB image reconstruction of the Optical image without using any extra techniques like the skip connection. This is the most direct approach as it generates the image in a single step having to do only one conversion. The activation function in the hidden layers is LeakyReLU and the output layer applies a sigmoid function. Skip connections were not applied for this first approach as this allows us to compare the results and if it is worth increasing the computational costs.

The result of autoencoding shows better reconstruction for grassland and agriland classes, as they contain simpler patterns for the network to learn. For agricultural fields, we can observe

that the network retains structural and edge information but struggles to preserve color information, especially if it deviates significantly from shades of green/brown. In urban cases, due to the complexity of patterns and rapid variations in pixel color and intensity, the network struggles to capture relevant information. It is limited to coloring the areas with green/brown and tends to ignore the urban features. We can observe what the network is focusing on through the feature maps, ranging from texture to the luminosity and color of pixels. It is also evident that, due to the considerable diversity and inherent differences in land types, some feature maps may not contain much information for one class. However, the same feature map may contain a wealth of relevant information for another class. Using MSE as the loss function for training and after training we calculate the SSIM of the outputted results we get a MSE loss of 0.022 and a SSIM of 0.43. For SSIM the objective is to maximize it. The results are observable in Figure 5.1.

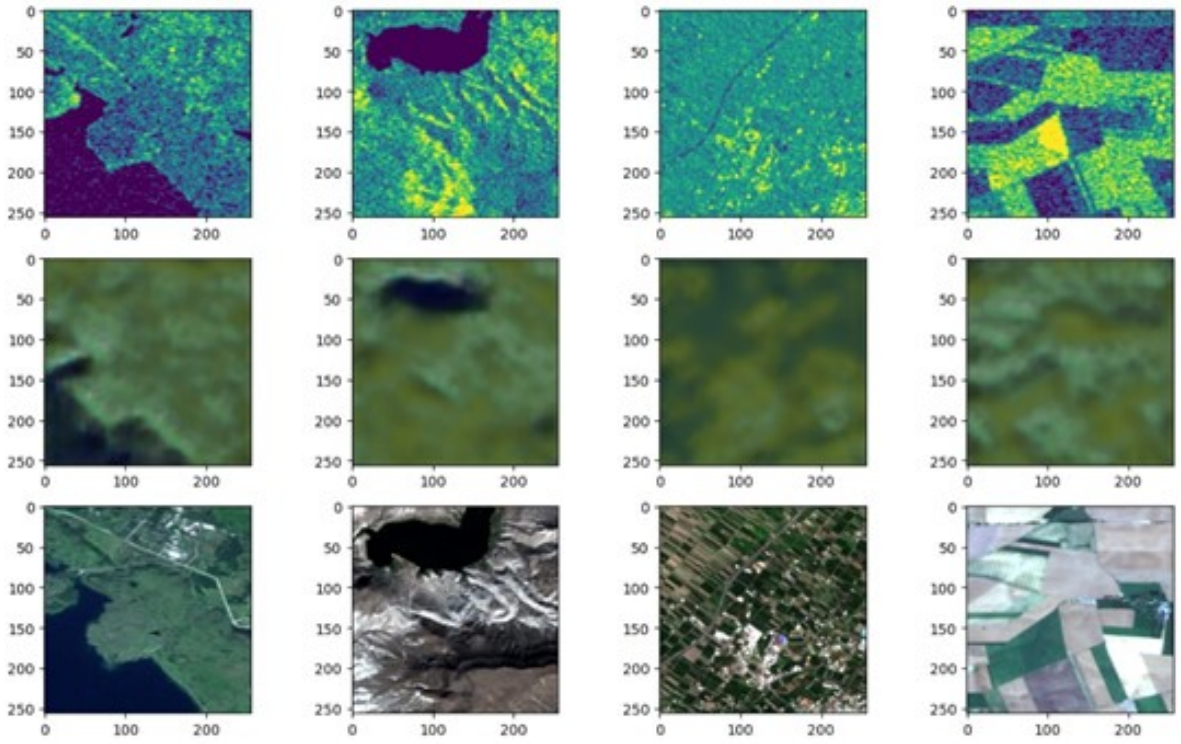

Figure 5.1 Reconstruction of an optical image using MSE for training. SAR image (top), generated image(middle), optical image(bottom)

While some color information is kept, we can observe that the predominant color is green. This is due to the fact that for most images, there is a large portion of green areas, such as forests, and other natural landscapes. This occurs because the model tends to be influenced heavily by the most common features in the training data, known as bias. In this case, since green regions are predominant in the dataset, the model learns to associate and recreate this color more often, leading to an overrepresentation of green in the output. This can occur due to an imbalanced dataset, where some classes or features are more prevalent, leading the model to be "biased" toward those features and less effective at representing less common elements.

A solution to this problem is to create specialized networks, each focused on a specific type of terrain. By creating terrain-specific networks, such as one for forested areas, another

for urban environments, etc, each network can be trained to better capture the characteristics and features of that particular terrain type. This approach allows for more tailored results, as each specialized model is optimized for a more specific set of landscapes, which reduces bias towards overrepresented classes. For instance, a network trained specifically on urban areas would be more adept at capturing fine details such as buildings and roads, while another trained on forested areas would be better at capturing varying shades of green and textures typical of vegetation.

A challenge is the need for a classifier network that detects the terrain type in the input image and assigns it to a specialized network. If the classifier network misclassifies the terrain type, such as when seeing new types of conditions like floods, forest fires, etc, the image may be processed by the wrong specialized network, leading to poor-quality results.

Instead of using multiple specialized networks, another approach to improve output quality is to use tailored loss functions for the problem at hand that focus on the perceptual quality of the generated images. For instance, using SSIM as the loss function can help the model focus on maintaining important perceptual features like textures, patterns, and structures in the image. SSIM measures the similarity between two images based on luminance, contrast, and structure, which helps the network focus.

When SSIM is used as the training loss function, we observed a SSIM value of 0.56 and an MSE value of 0.23. Although the MSE value is relatively the same as before, the SSIM increased and the resulting ouput also had higher quality fidelity. This means that SSIM might yield better perceptual results, as it prioritizes the structural integrity of the output. By using SSIM, the generated images better match the texture and color patterns of the original image, leading to visually more convincing results, as can be seen in Figure 5.2.

Figure 5.3 shows the difference between the application of MSE and SSIM as a loss function. It is evident that the resulting images generated using SSIM as the loss function have a much clearer appearance, both in terms of texture and color quality. The color blur aberration is significantly less noticeable in the SSIM-based model, as SSIM measures perceptual aspects such as luminance, contrast, and structure, while MSE only measures pixel-wise differences, leading to a lack of perceptual detail preservation.

Despite the benefits of SSIM, both models fail to capture urban features present in SAR images effectively. This shortcoming is primarily due to the limited size and number of feature maps in the current model architecture. Each feature map can detect a specific pattern in the image, such as edges, textures, or color variations, and increasing the number of feature maps can improve the model's ability to capture these diverse aspects of an image. Given enough maps one/multiple might focus on these city features enabling the model to learn its patterns to recreate it. However, increasing the number of feature maps also comes with a computational cost. More feature maps mean more parameters to train, which in turn requires more memory and computational power.

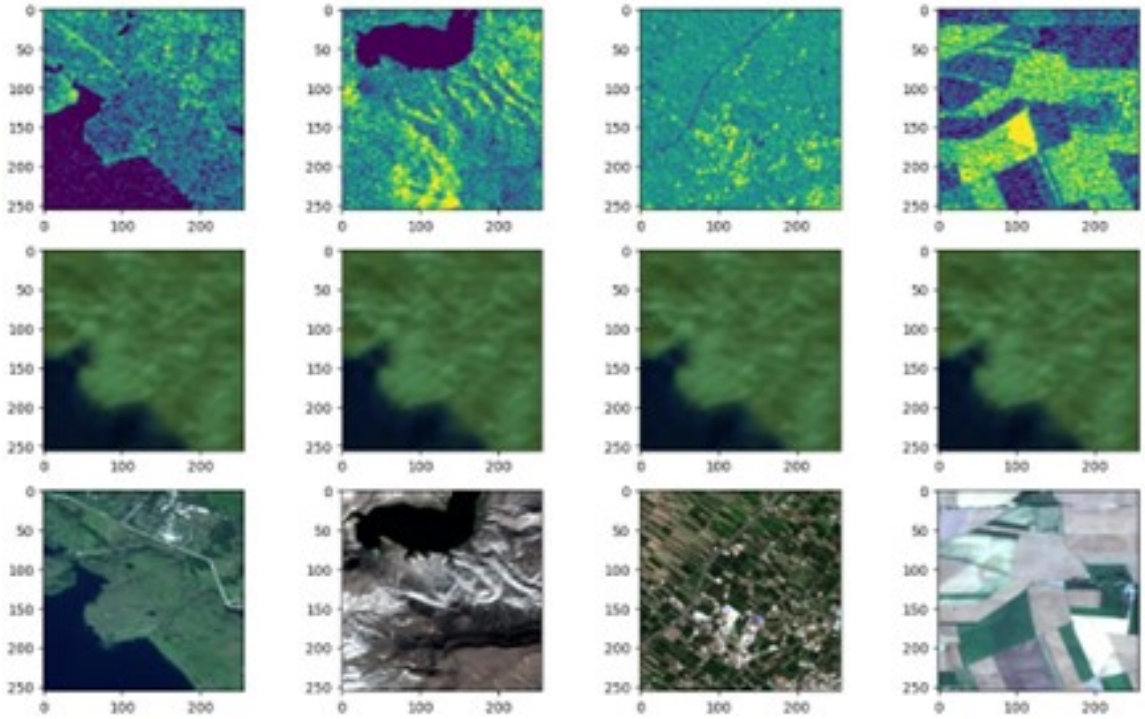

Figure 5.2 Reconstruction of an optical image using SSIM for training. SAR image (top), generated image(middle), optical image(bottom)

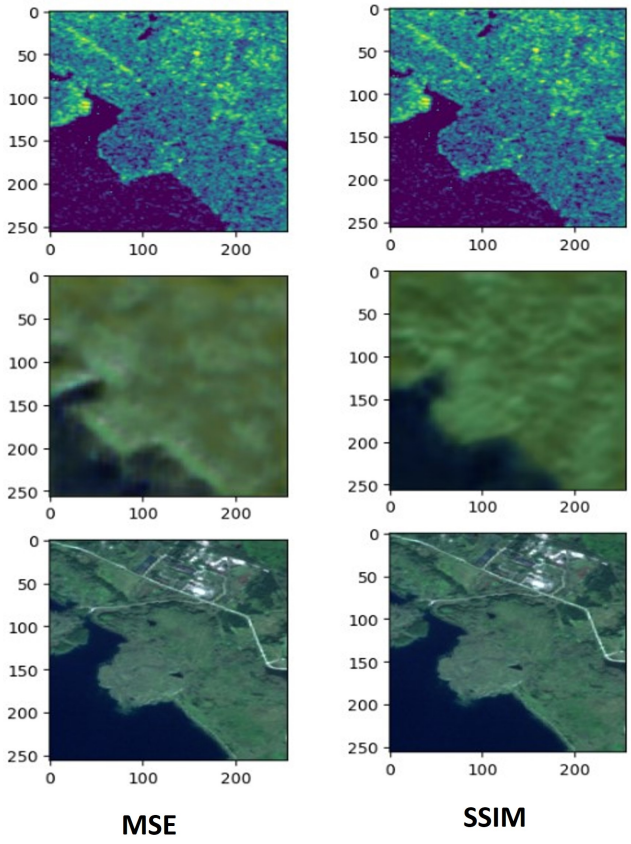

Figure 5.3 Model output comparison of MSE (left) and SSIM (right) as loss functions

By integrating skip connections into the network, we facilitate the transfer of richer information directly to the decoder block. When combined with the encoded features, this allows the network to generate representations that are truer to the original. Implementing these connections can be achieved by summing the feature maps from preceding layers with the feature maps from the encoder block that share the same spatial dimensions. With this method in place, we notice that the reconstructed images bear a closer resemblance to the original optical ones. However, when evaluating based on loss metrics, the outcomes appear relatively unchanged. This can be attributed to the loss functions which are not capable of capturing these added texture information when calculating the loss. Like the previous example if we use the MSE for training we get a MSE loss of 0.2 and a SSIM loss of 0,52. We can observe in the Figure 5.4 that more texture detail is kept in the image yet there are a few more color aberrations in the image.

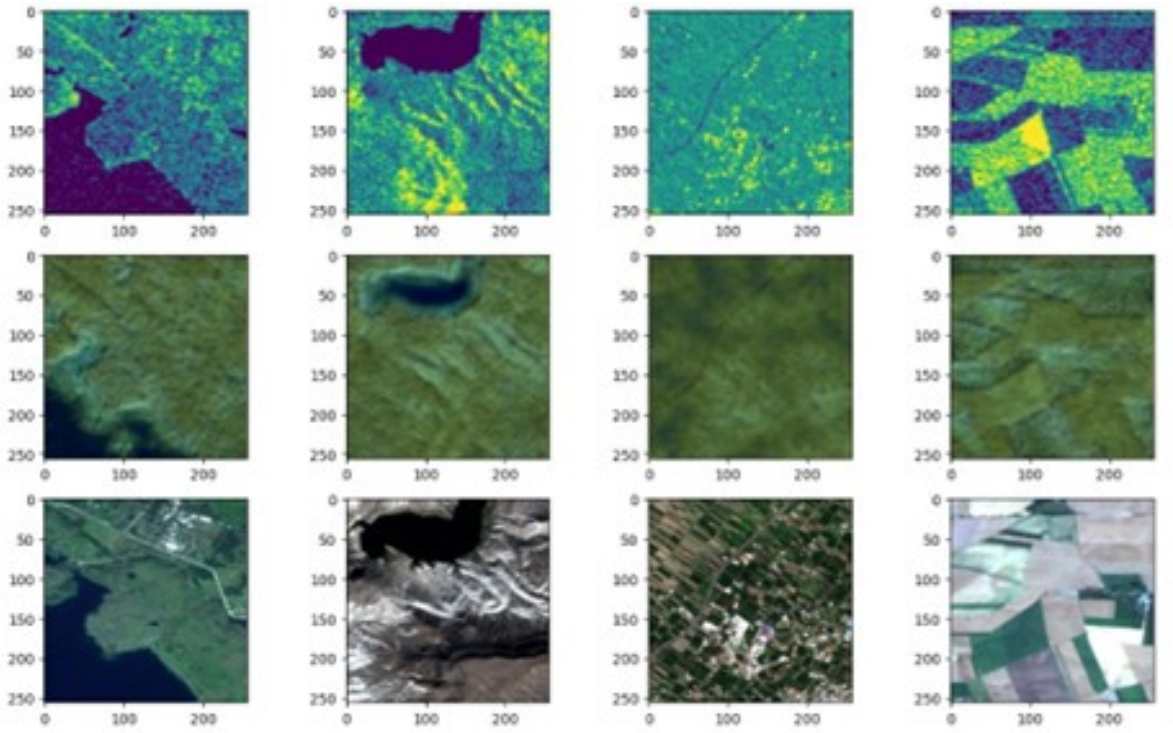

Figure 5.4 Reconstruction of an optical image using MSE and skip connectins for training. SAR image (top), generated image(middle), optical image(bottom)

By using the SSIM for training we get a SSIM loss of 0,63 and a MSE loss of 0.23. Like the previous example we can notice in Figure 5.5 that while keeping more texture it also increased the aberrations.

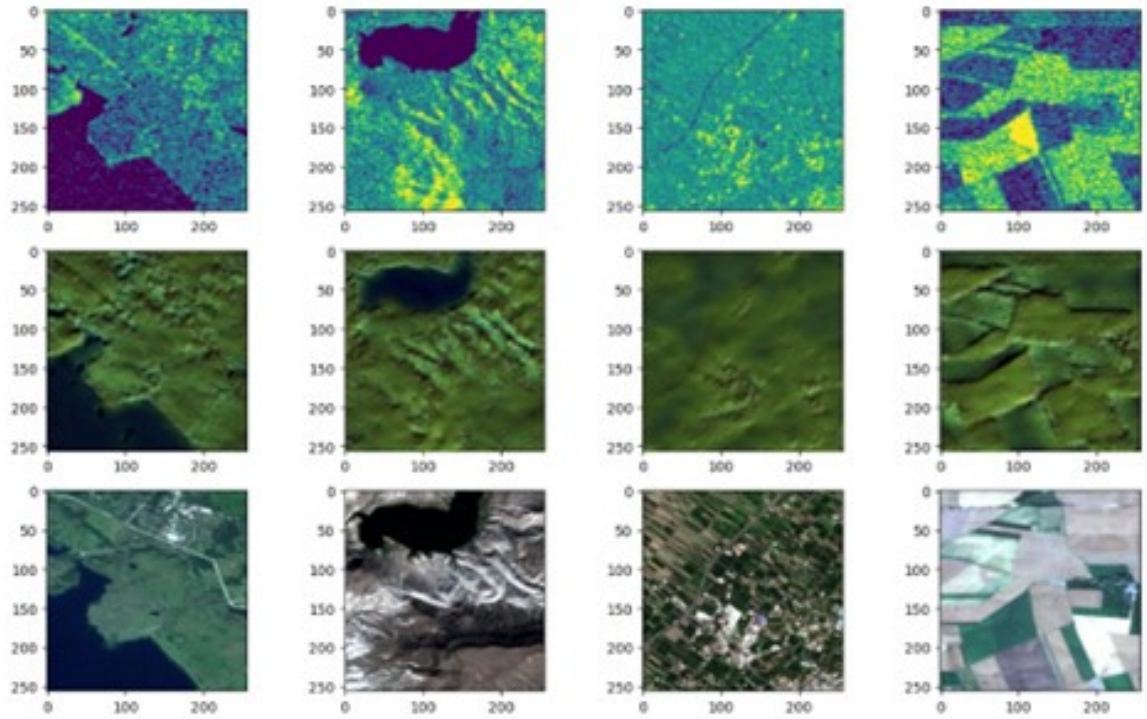

Figure 5.5 Reconstruction of an optical image using SSIM and skip connectins for training. SAR image (top), generated image(middle), optical image(bottom)

Figure 5.6 showcases the multiple combinations of models one can have with just 2 loss functions and the application of skip connections. Due to the SSIM structural component the resulting image edges are much more noticeable than when using MSE. Skip connection greatly reduce the image blur however due to the model inability to capture much of the edges and details the chromatic aberrations created are also more noticeable. This is because the decoding block is now combining the SAR image features when these are passed through the encoding block, meaning that the SAR features also become more pronounced in the optical image, leading to some aberrations.

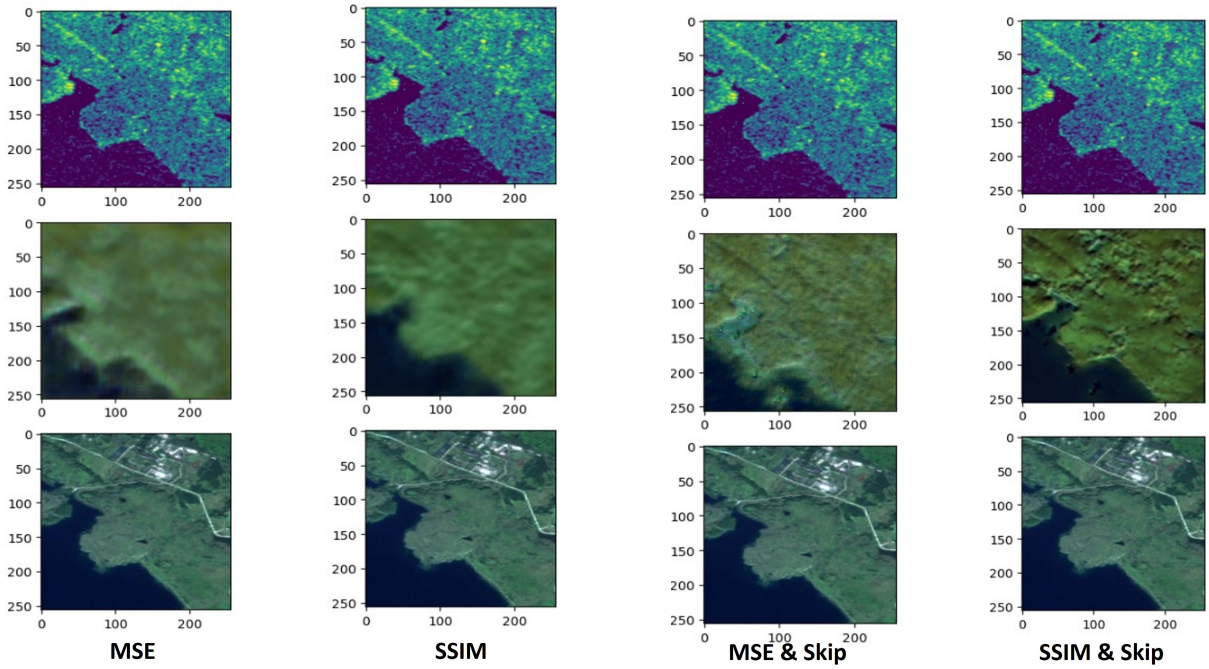

Figure 5.6 Comparison between the different model outputs

By splitting the channels into three different ones (red, green and blue) and creating three specialized networks we can generate each channel independently as to reduce the dimensionality of each generation. Afterwards these independent channels are combined into a 3 channel image that represents the optical RGB image. This architecture is depicted in Figure 5.7.

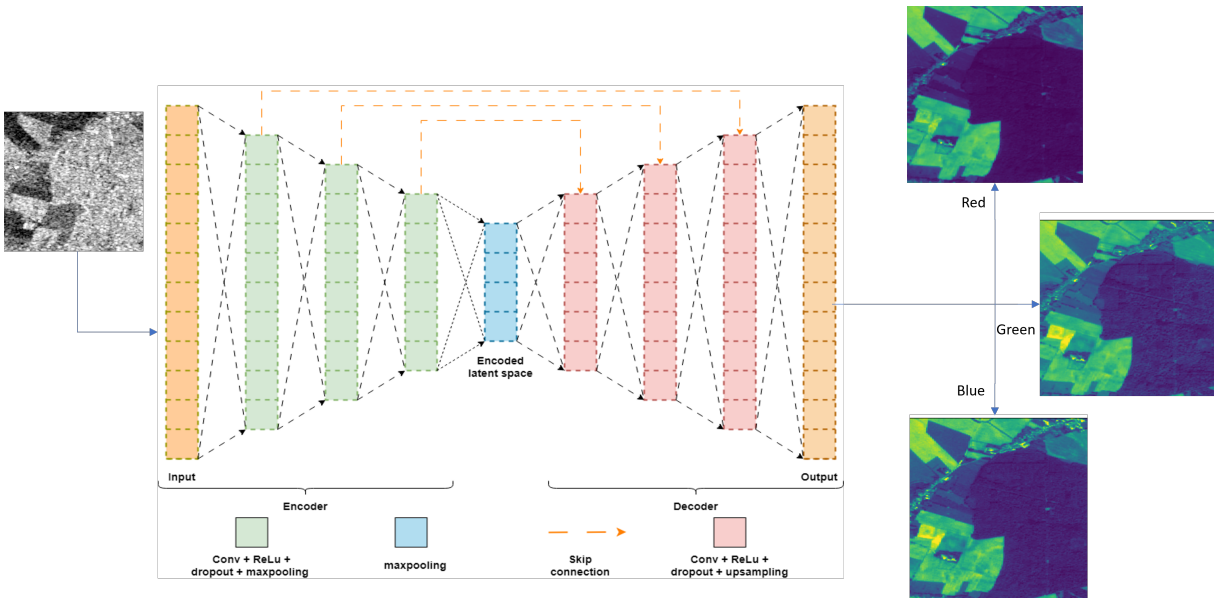

Figure 5.7 Specialized agent model architecture.

For the red channel the obtained loss was 0.0252, for blue it was 0.0287 and for green it was 0.0176. When observing the recreated color channels independently and compared to the true optical color channels (as observed in Figure 5.9). Even though the loss was lower than when using a single generator we can observe more inconsistencies in the color generation.

While the loss results are better the inconsistencies appear since the green channel became overfitted as it is the predominant color in most images as observable in Figure 5.8.

Considering that the green color channel overfitted and became more prevelant than the other channels, one approach could be to try and equalize the color histograms, as these color chanels are now independently from another, as to try to reduce the impact that each channel has. The resulting equalized color channel can be analyzed in the bottom row of the Figure 5.8.

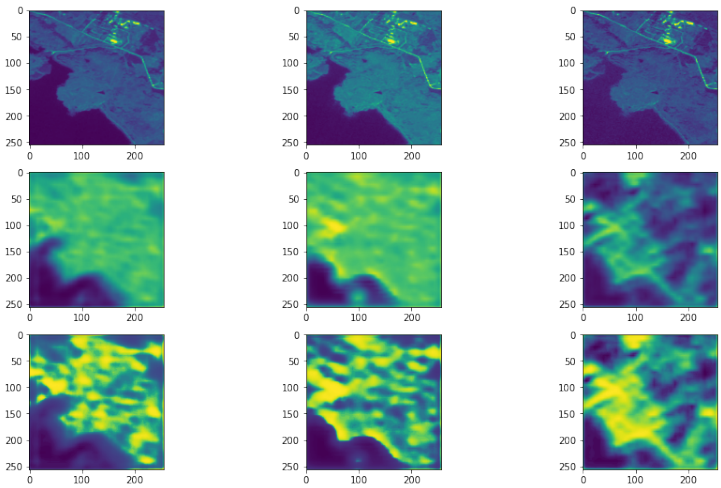

Figure 5.8 True RGB optical color channels (above), generated color channels (middle), and equalized color channels (below)

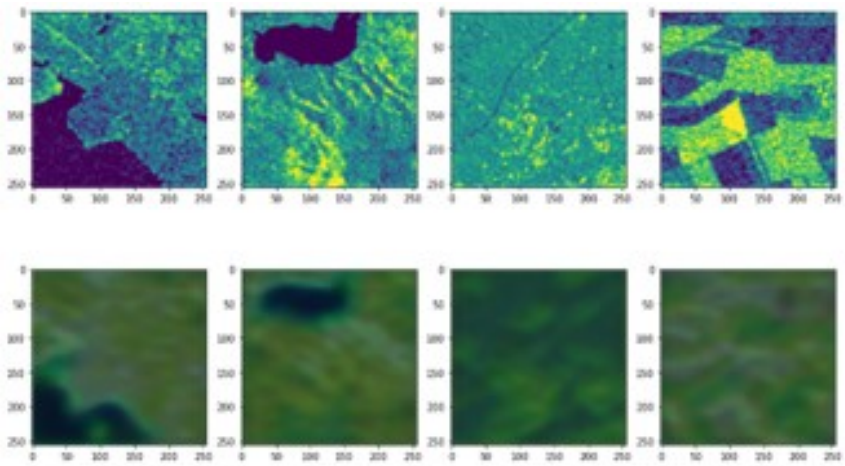

Figure 5.9 Optical image recreation by splitting the output channels into different networks and training using a single class.

However, while this equalization is done considering that the color channels are independent, in practiacal terms these are not, and so when applying the histogram equalization and consecutive combination of the colors channels the produced image becomes a inconsistent mixture of colors (observable in Figure 5.10), that while retain outer shape, as this is represented by a big gradient shift in both red and green channels due to the sea as observable in the bottom row of Figure 5.9, the pixel colors of the red and green band get equalized torwards the average pxiel value, which in turn mixes the pixel hue values when combining.As a

result, this approach creates the need to create an extra network that is trained to combine the channels in a more natural generation.

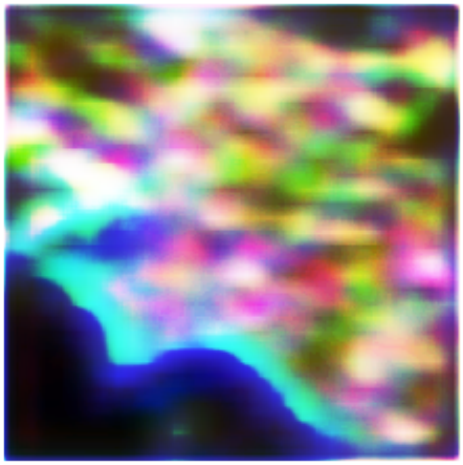

Figure 5.10 Combination of equalized color channels generated by each specialized network

An alternative strategy to address the current issue is to deploy a distinct autoencoder for each class. This method, in contrast to employing a universal image-to-image model, tailors each autoencoder to a particular terrain type. As a result, while it enhances performance for the designated class, it might degrade outcomes for other categories. This effect can be likened to overfitting. However, in the context of image reconstruction, this specialization might be advantageous. Yet, for real-world applications of this technique, a prerequisite would be a classification network that precedes the autoencoder, categorizing terrains into types such as grassland, barren land, and so on. A feasible way to implement this is by utilizing pretrained models like ResNet. Here, only the concluding layers require fine-tuning to train it to the specific challenge at hand. Once the terrain classification is determined, the image is subsequently channeled to the corresponding encoder. A notable benefit of this methodology is the fact that the training process was already done. However fine-tuning the network can prove to be computational expensive. Figure 5.11 show that for this model the predominet color is still green, meaning that this channel is still overfitting.

By expanding the network size to 64, 32, 16, 8 for the encoder and using a mirrored configuration for the decoder, coupled with a higher number of epochs, it becomes evident that the network can capture enhanced texture details. However, this enhancement also amplifies the issue of color aberration, as channels tend to become more specialized and distinct.

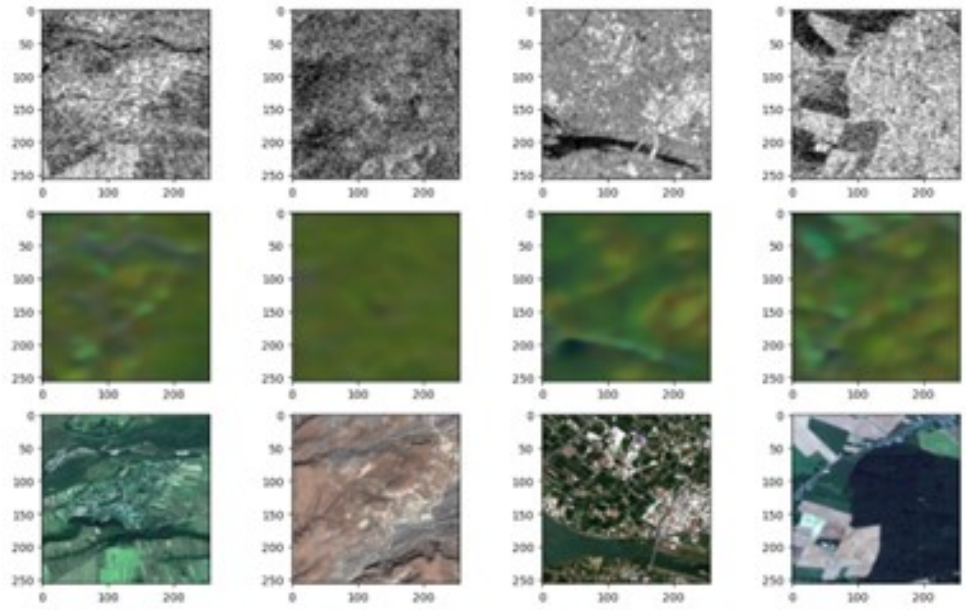

Figure 5.11 Optical image recreation by splitting the output channels into different networks and training using a single class.

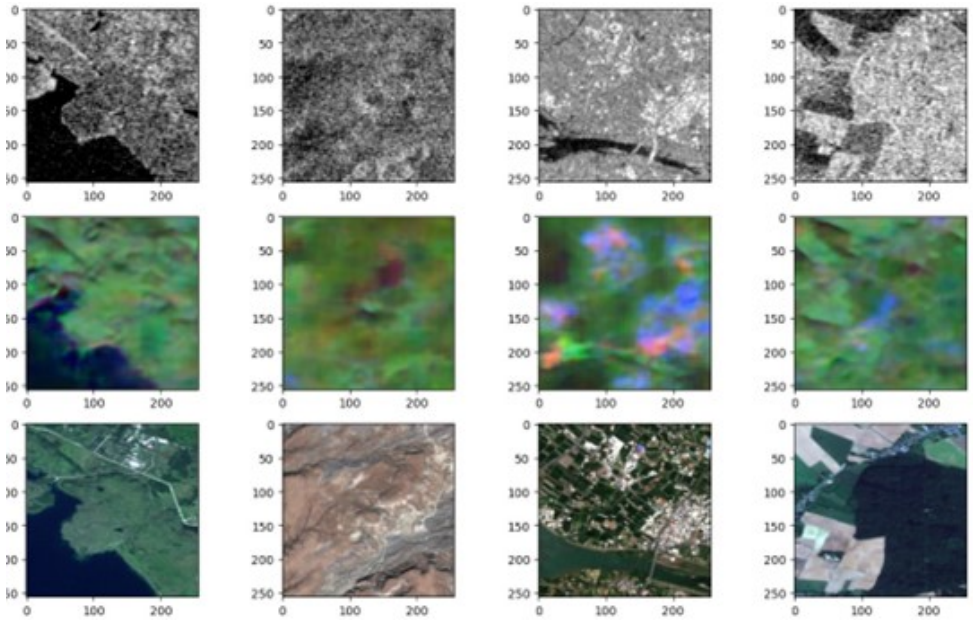

Figure 5.12 Optical image recreation by splitting the output channels into different networks and training using a single class and increasing the networks size

By adopting the initial approach where the networks generate an RGB image in a single step without separating the channels, and by enlarging the network size to 128, 64, 32, 16 for the encoder (and the inverse for the decoder), coupled with an increased number of epochs, we notice a significant reduction in the MSE loss, which drops to 0.014. As illustrated in the subsequent figure, expanding the network enables us to capture and retain a greater amount of information, encompassing both texture nuances and aberrations.

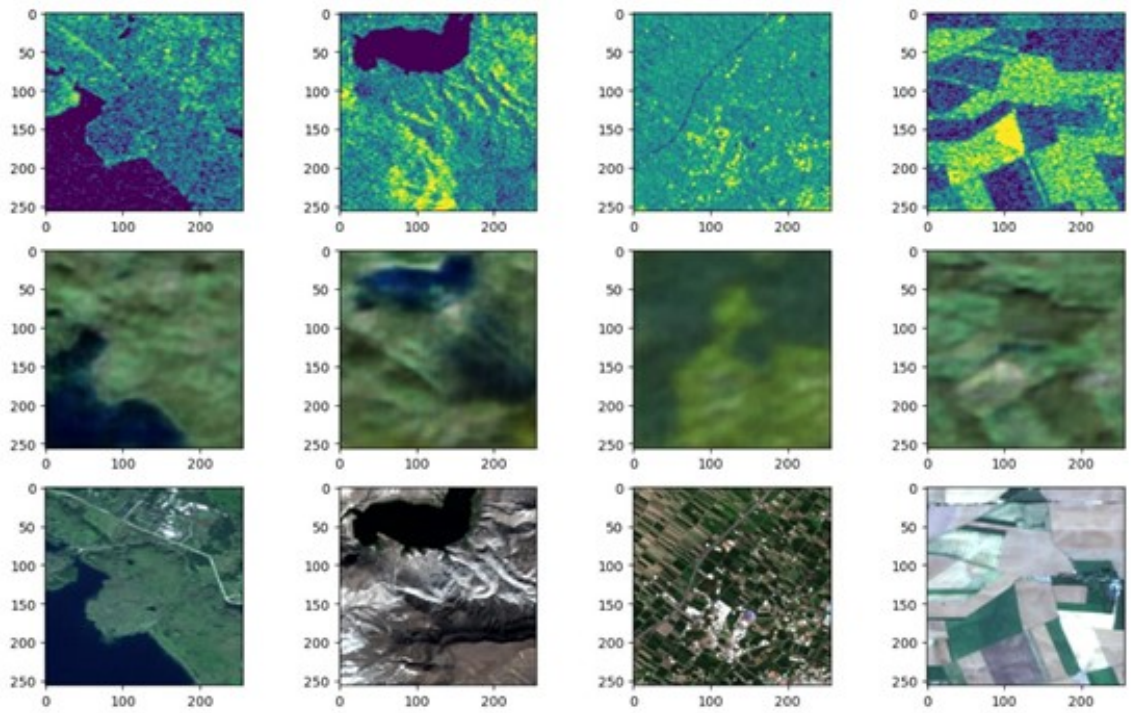

Figure 5.13 Optical image reconstruction using a larger network with more epochs

If we observe the training loss (see Figure 5.14), it is possible to observe that the model is heading towards convergence, meaning possible signings of overfitting. As a result the model might be either reaching a local minnimum or a global minnimum, which for the later might be impossible to solve without adding deeper layers.

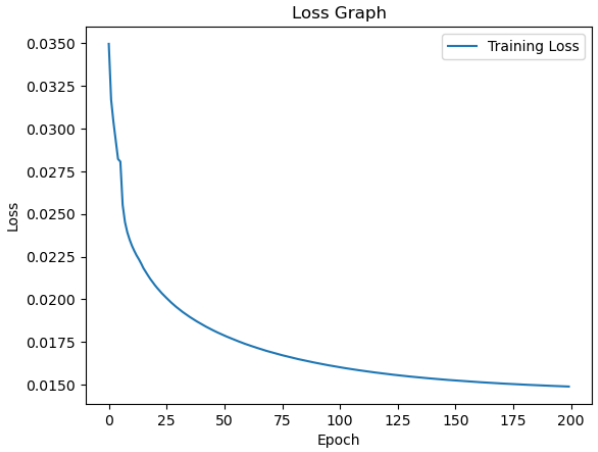

Figure 5.14 Training loss

When we incorporate skip connections, the loss decreases to 0.010. By using various techniques, we can produce an image that captures both color and texture details of larger features. However, it's noteworthy that the network struggles to retain finer details, such as those in cities, or rapidly changing color patterns, like in agricultural fields.

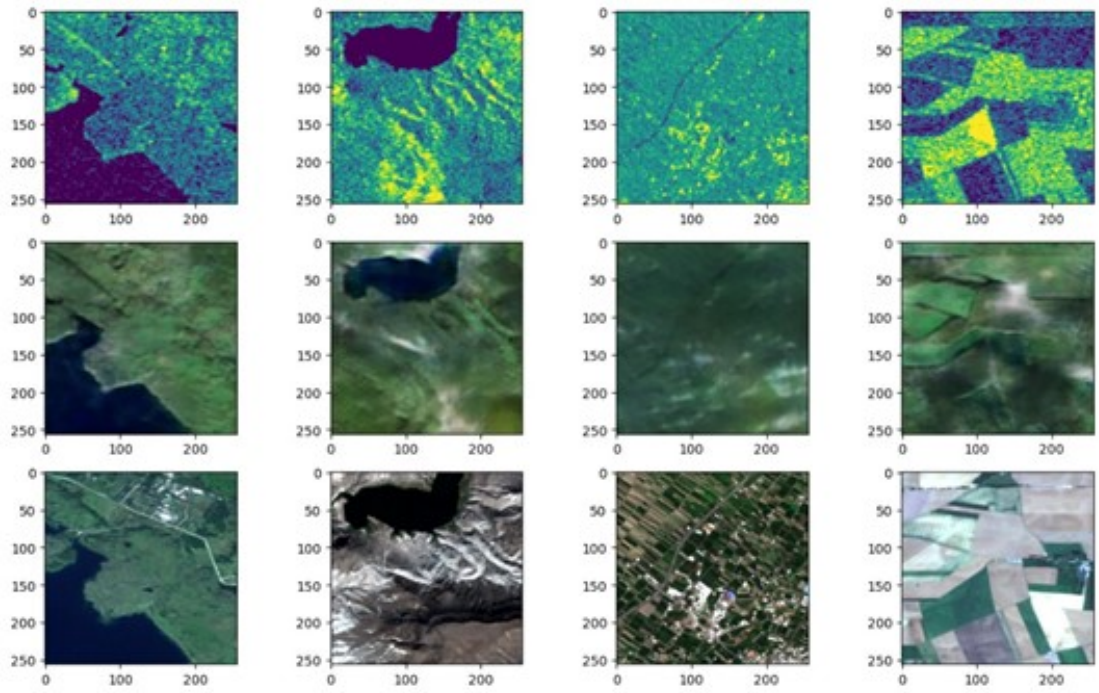

Figure 5.15 Optical image reconstruction using a larger network with more epochs and skip0 connections

By looking at the loss graph in Figure 5.16 it is possible to observe that the model while improving it still does not show signs of overfitting, which means that given extra processing power and an increased model the results could have greater quality.

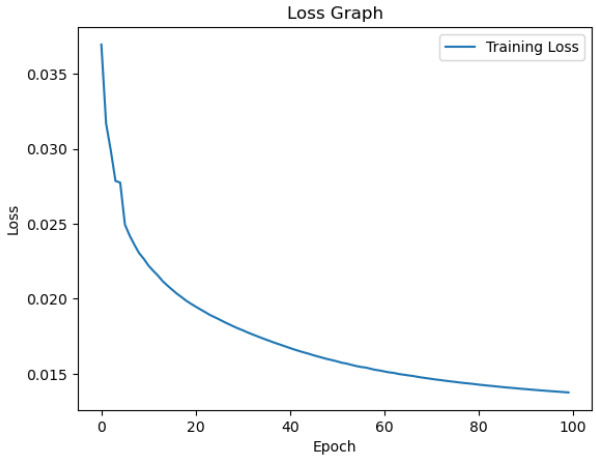

Figure 5.16 Training loss function

Chapter 6

Limitations

The main limitation when dealing with neural network models is the available computing power. While increasing the network size will lead to better and more accurate generalizations, it also requires more VRAM to load the model and the data which can be bottlenecked by the current VRAM. Increasing the model will also affect the speed at which the network iterates through the data, which will slow down the training process. While the architecture would greatly benefit from adding other models such as a GAN, it was not feasible to add it due to this computing power bottleneck. While it is not possible to train larger networks it is possible to infer that these would most likely increase accuracy due to the fact that the current network is not overfitting and so it still has not reached its maximum performance. Possible solutions to this computing problem are to move the network to a cloud-based platform like Google Colab but this creates other problems as the data used to train the model does not have any sort of API that the model in a cloud based environment could access. This would create the need to also migrate the database to the cloud and due to the size of the dataset it would not be practical as the storage offered by these services is also limited or paid.

The quality and quantity of data is also a major limitation to neural network performance. A poisoned dataset or a small dataset will lead the network to incorrectly learn features. While the obtained dataset was of good quality and provided a good quantity of data points, the network would still greatly improve if the dataset was more populated, however due to the topic of the thesis it is difficult to find good quality datasets of both SAR and Optical paired images. Manual extraction of data points helps to complement the dataset but due to the effort needed to process and manually align the SAR and Optical images this manual extraction of data at this point of the thesis cannot fully replace the need for an already made dataset. As future work an automatic script could be made that extracts both SAR and optical images, pre-processes and aligns them and saves it to a dataset. However, automating this is not trivial due to the inherent fact that both SAR and Optical images can be captured at different incident angles and at different times.

Another limitation to the model is the inherent fact that unless the same system was used to capture both the optical and SAR images, these might not display the same features. If the area of interest is a harbor where many vessels are entering and leaving the port each minute,

a difference of a few minutes will display the vessels in different positions between the images. When the network learns the patterns to convert from SAR to optical, these moved vessels will act as poisoned data points that might trick the network into incorrectly coloring the water as red or a vessel as blue. This introduces the need to detect and remove vessels as a preprocessing step to ensure a correct feature generalization.

Vehicles like cars and bikes also introduce poisoned data as they might increase the reflected backscatter of a certain area or, if traveling at a high speed, might suffer from Doppler shift and appear in incorrect positions in the image [8]. These also misguide the network, which will lower the model's quality. Yet, due to their smaller size when compared to the satellite ground resolution, they won't poison the data as much as vessels. This effect depends however on the utilized resolution. When using SAR and optical systems with better resolutions, this effect will be more pronunced as the smaller veichels will be less blurred into the background and consequently this Doppler shift of the object will be more visible.

Any other feature that changes between the SAR and optical systems, such as buildings being demolished, or any geographic variation, like a flash flood or a landslide, will also cause poisoned data and should be removed from the dataset. Other problems, such as different incident angles between the systems, can also degrade the network.

However, these problems only affect the training phase of the data. After training, there is no need for the optical image, and if the network was correctly trained to also color these moving features, it will correctly place them. A new network can be trained specifically for coloring these, acting as a specialized network only called when a vessel is detected. Since this would require a second network and, due to computational limitations, it is preferable to discard any segment of an image that includes a vessel.

There are many different algorithms to detect objects in images, ranging from traditional thresholding methods to more advanced neural networks trained to detect and classify ships. Global thresholding algorithms are the simplest and work by defining a threshold value, and if any pixel is above such value, it is considered a pixel of interest. This has the disadvantage that it doesn't take into consideration the material from which the object was constructed, as smaller wooden objects might not reflect as much compared to large steel structures, creating false negatives.

As seen in Figure 6.1, global thresholding can lead to these limitations when applied to ship detection.

The next types of algorithms are based on an adaptive threshold, with the cell averaging Constant False Alarm Rate (CFAR) detector being the most commonly used. This type of algorithm works by finding pixels that, when compared with the background, are unusually bright.

As illustrated in Figure 6.2, CFAR algorithms help to mitigate some of the issues seen with global thresholding.

CFAR operates by examining a cell of interest which potentially contains a target. This is done by comparing the statistical values of the surrounding neighbor pixels to the pixel being

Figure 6.1 Global thresholding

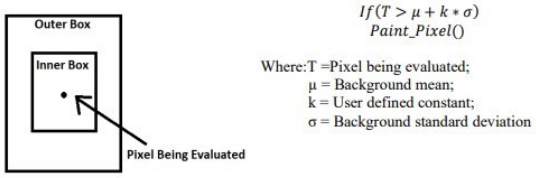

Figure 6.2 CFAR algorithm

evaluated. If the pixel being evaluated has a considerably higher backscatter value compared to the background, it is then considered a pixel of interest. These types of algorithms create false alarms, so they must be eliminated, leading to the creation of discriminatory algorithms that try to discard false alarms by analyzing cluster properties like size, angle, speed, pixel positioning statistics such as pixel standard deviation compared to the mean pixel position, among others.

As shown in Figure 6.3, a comparison between global thresholding and CFAR algorithms demonstrates their different approaches and effectiveness.

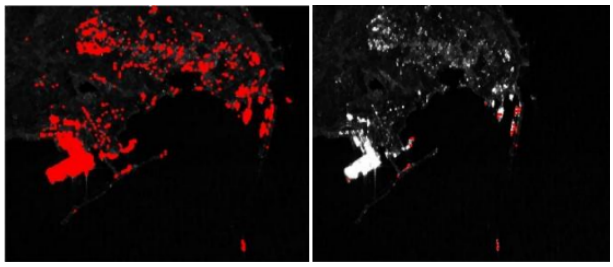

Figure 6.3 Comparison between global thresholding and CFAR algorithms

After the objects have been identified, these can be removed from the dataset to prevent introducing noise into the network. A future approach could involve implementing a neural network specialized in coloring these types of moving objects. For this, it would require an algorithm that detects these objects, and for those patches, it calls a network that was only trained to color said object. However, this only works for cases where the object is present in the SAR image. If the object is only present in the optical image, it will always act as poisoned data, as the network will not have any reference point in the SAR image and thus must be

removed.

Chapter 7

Conclusions and future work

The objective of this thesis was to create a computationally efficient solution to reliably convert from the SAR image domain to a color domain recreating the corresponding optical image of the area of interest, with the objective of obtaining state-of-the-art results using only a fraction of the computational resources that GAN systems would use. Prior to training, some preprocessing and normalization are done to ensure that the training process is stable and to reduce the odds of the explosion/vanishing gradient problem.

Other techniques like residual skip connections and single-channel recreation were tested to determine the validity of these solutions. Skip connections significantly improved the results while maintaining the computational cost of the solution. Splitting the output by channel increased the accuracy of each channel, but when combining these, the results show some aberrations. For this, another network can be created to fuse the channels more consistently. The applied loss also weighs on the results, as it will define the way the network learns and optimizes its solution. MSE metrics provide more faithful color recreation, while SSIM is able to keep more structural information.

A possible future improvement to the architecture is by applying Feature Pyramid Networks (FPN) to the encoding and decoding layers. FPNs are a class of CNN architectures designed for handling object detection and recognition tasks at multiple scales. These networks capture details in multiple spatial resolutions by either consecutively downsampling the image or segmenting it into smaller pieces. By capturing features in multiple spatial resolutions, FPNs enhance the network's ability to extract both finer-grained and more general low-level details, improving performance across various computer vision tasks.

FPNs typically consist of two main components: a backbone network and a feature pyramid network. The backbone network serves as the foundation of the FPN and is responsible for extracting low-level and high-level features from the input image. These layers can be trained from scratch or fine-tuned from pre-trained models to suit the specific requirements of the problem at hand. Common pretrained backbone networks used in FPNs include variants of the ResNet, VGG, or EfficientNet architectures, known for their strong performance in image recognition tasks.

The backbone network typically comprises a series of convolutional layers followed by

pooling layers, which progressively reduce the spatial dimensions of the feature maps while increasing their depth. Alternatively, if the image is segmented into smaller chunks, convolutions are applied to each segment individually, and the results are combined for the feature pyramid network.

The FPN component builds upon the feature maps generated by the backbone network to create a multi-scale feature pyramid. This pyramid structure is achieved through a combination of lateral connections and upsampling, or stitching, depending on whether the backbone downsamples the image or segments it. Lateral connections involve establishing connections between feature maps at different levels of the backbone network. Specifically, feature maps from higher-resolution layers of the backbone network are combined with feature maps from lower-resolution layers through element-wise addition or convolutional operations. These lateral connections enable the propagation of high-resolution semantic information to lower-resolution feature maps, enhancing the network's ability to capture fine details while maintaining contextual understanding across different scales.

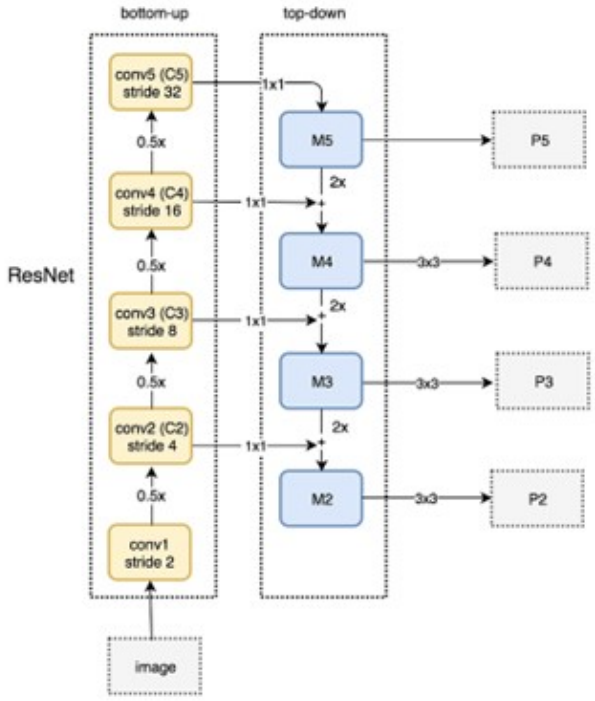

Figure 7.1 Feature Pyramid Network architecture

Upsampling or stitching is then performed to further refine the feature pyramid. Upsampling involves increasing the spatial resolution of feature maps through interpolation techniques such as bilinear or bicubic interpolation, effectively restoring lost details during downsampling. Stitching, on the other hand, involves combining feature maps from segmented regions of the image to create a coherent feature pyramid. This process ensures that the network can effectively capture information at different scales and spatial resolutions, enabling robust feature extraction and improving the network's performance on tasks such as object detection and semantic segmentation. For the output, it was also possible to extract it in multiple spatial resolutions which can further help the network to generalize the features and faithfully reconstruct

the optical images. Even though these networks yield better results, the computational cost of adding this technique must be considered as extracting features in multiple resolutions might overwork the available hardware and might require new one.

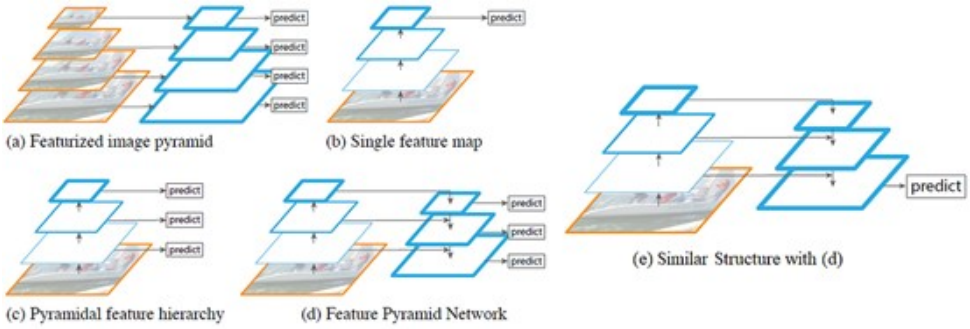

Figure 7.2 Possible Feature Pyramid Network approaches

Other approaches based on GAN have proven to be effective. These networks are self-trained networks that effortlessly improve the outputted results by pairing two networks to compete with each other as the generative network tries to fool the discriminative network with "fake" optical images generated by the autoencoder and the discriminator is tasked with detecting if the image presented to it is indeed a real or generated optical image. These two networks compete in a game-theoretic scenario, which helps them improve over time until the generator produces nearly indistinguishable samples from the original data. The discriminator input would be the SAR image, and the output would be the optical generated image. The discriminator learns to distinguish between real data (drawn from the training dataset) and fake data (created by the generator). Its input is real data or generated data, and the output is the probability that the input data is real, from which the generator trains from to create increasingly more convincing images.

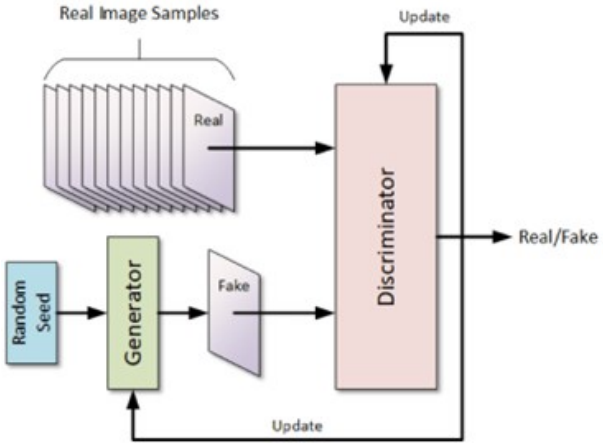

Figure 7.3 Generative Adversary Netowrk architecture

A solution to the problem of coloring vessels and moving vehicles in SAR images can be overcome through the creation of specialized models tasked with learning more concise representations for a specific set of objects. For this, a network would be trained mainly with images of the objects of interest, i.e., vessels, cars, planes, etc. This specialized network

could not be directly applied to the SAR image as it would be overfitted due to only training on a specific set of images. For this, it would need an algorithm like CFAR or others that detect these objects of interest and apply the specialized network just on these objects. For the problem of small structures, like missing buildings in the reconstructed optical image, it is possible to follow a similar path. If increasing the network size still proves to not be sufficient to capture these details, a network could be trained on capturing these finer patterns and adding them to the generated optical image. For this, the network would need to be trained on the "error" of the main generator, where the input for this network would be the difference between the generated and the real optical image. With this, the network would focus on fixing the main errors of the other network, possibly aiding in representing small structures.

Polarimetric SAR data could also be used to enhance the data information given to the network. SAR systems are able to acquire data at different imaging modes, including a single-polarized mode when the transmitted and received signals are of the same polarization, transmits for example in horizontal polarization and receiving the backscatter also in an horizontal polarization, creating two possible combinations of horizontal-horizontal (HH) or vertical-vertical (VV)[46, 47].

SAR remote sensing systems are also capable of capturing data in a dual-polarized mode. For this the systems transmits a signal in a single polarization (H or V) and receives two backscatter signals (HH and HV or VV and VH), corresponding to the original sent signal polarization and the received signal polarization. This is done by having two receiver antennas in orthogonal directions. Considering classical antennas like dipoles, where its orientation defines the polarization it captures in relations to the transmitter. One antenna could be horizontally to capture horizontal polarized backscatter and another vertically, making it possible to capture both polarizations. Another practice is to place them diagonally, with an angle of -45º and +45º.

This way each of the antennas would capture the combination of both the horizontal and vertical components of the backscattered wave, that when combined with the other diagonally placed antenna allows the system to separate both the horizontal and vertical component due to the orthogonality of the antennas. This also makes the system less dependent on the target orientation as these capture the combined vertical and horizontal component, while keeping the received signals orthogonal. It is also possible to obtain the backscater signal in a quad-polarized mode, or fully-polarimetric mode, when the systems transmits both the single-polarized horizontal and vertical signal and captures both possibilities for each of the sent polarized signal, that captures all four possible combinations.

Recently with the application of antenna arrays the backscatter signal capturing has become more dynamic, where by controlling each element of the array independently, it is possible to control the emitted signal polarization with a single antenna, altering from vertical to horizontal without the need to reconfigure it. It is also possible to capture both vertical and horizontal polarizations with a single antenna.

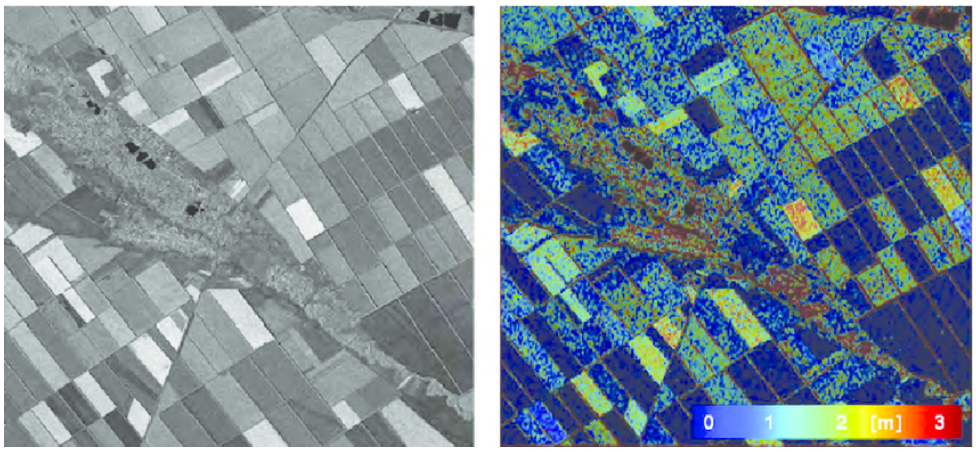

Figure 7.4 SAR image (left) and polarimetric SAR image (right)

Polarization refers to the direction of the electric field vector of an electromagnetic wave. This defines how the radiated radiated electromagnetic wave changes both in direction and relative magnitude in terms of the electric field vector over time, and is characterized by some parameters. Considering a wave traveling in a k-axis this can be decomposed into three orthogonal components x-axis, y-axis and z-axis. Considering that z-axis represents the direction of the wave (meaning that is parallel to the signal), the wave can be characterized by its amplitude, the initial signal phase, the phase difference over time, the signal tilt in relations to the x-axis and the elliptical properties of the wave (amplitude, angle and elliptical direction), in relations to its direction [46, 47].

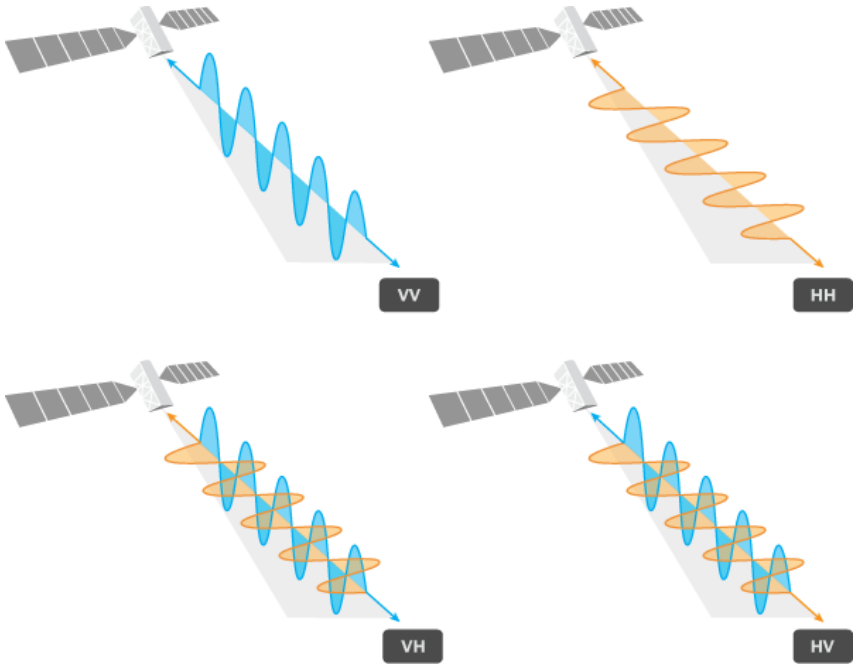

Figure 7.5 Visual representation of transmitted polarization and reflected signal polarization

In terms or radar systems, radar polarization describes the process of acquiring, processing and analyzing the polarization state of an electromagnetic wave. When radar system are considered, as it transmits and receives electromagnetic waves, the process of interest is the

scattering process of when the transmitted electromagnetic wave reaches the target and interacts with it, scattering in multiple directions, which part of it eventually reaches the remote sensing system. This scattering process depends both on the emitted signal and the target properties, such as its geometry or construction material, as this affects the direction and intensity of the reflected wave, possibly causing multiple reflections. The reflection process can be described by a scattering matrix, which contains the measured complex scattering coefficients from the different polarization combinations [46, 47]. Considering this the orientation of the target will affect the scatter process, where a scatterer aligned with the polarization of the eletromagnetic wave might reflect better than a perpendicularly aligned target. Due to this different emited polarizations can capture different aspects for the same features.

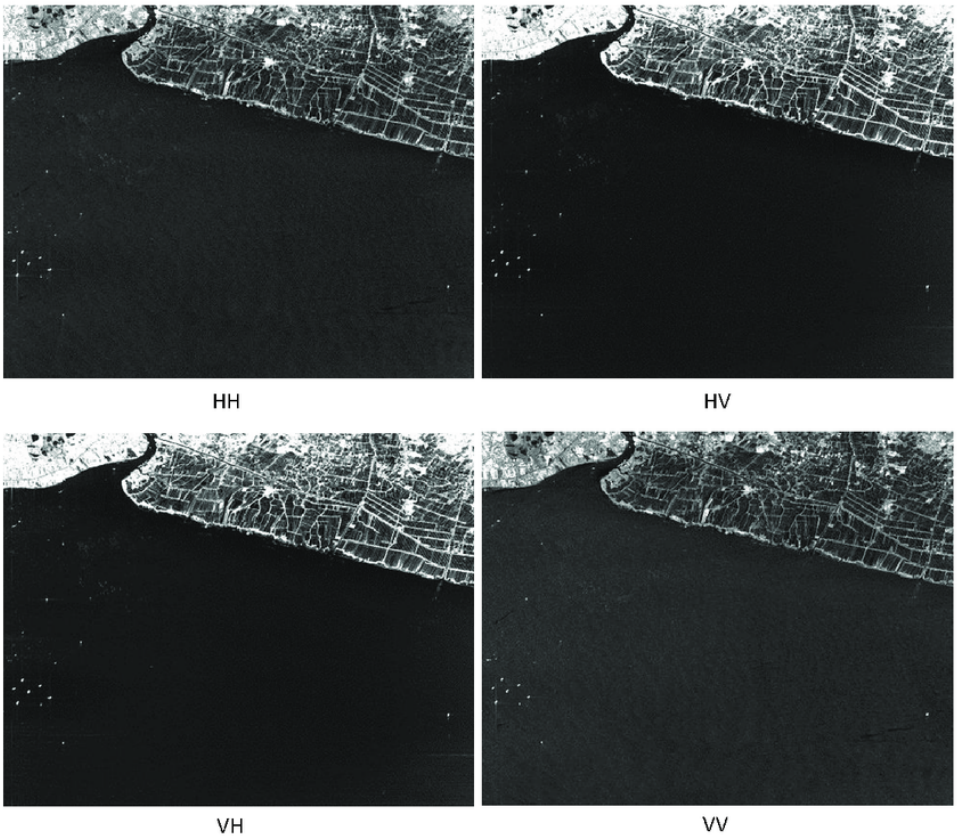

Figure 7.6 Example of the effects of the target orientation and geometry between the different polarizations

Since the acquisition of polarimetric SAR data involves systematically transmitting pulses in one polarization and measuring the backscattered signals in both the same and orthogonal polarizations. For instance, when a horizontally polarized wave is transmitted, it may interact strongly with linear features aligned horizontally, such as roads or flat surfaces, and the HH channel would capture this response. Conversely, the HV and VH channels, known as cross-polarized channels, are sensitive to depolarizing targets like vegetation. In such cases, the structure of leaves and branches causes multiple scattering events that change the polarization state of the waves, resulting in a significant cross-polarized return. Vertically polarized transmissions interact differently, often highlighting vertical structures like trees or man-made

features such as buildings [46, 47].

Each of the polarized signals can be analyzed individually and could be individually feed to the network, training to color the SAR image no matter the received polarization. It is also possible to feed the network all the polarized images independently, meaning the network would learn what patters matter the most in each polarization to color the SAR image. Another approach when analyzing polarimetric SAR involves assigning each of the polarizations to a different color band (for example setting the red color channel to the HH polarization, the green channel to the HV polarization and blue to the VV polarization).

This results in a pseudo colored image, where each colors represents the combined backscatter energy of the different polarization modes. With this the network would be fed a single image that represents the combination of the different polarization and its dependencies, meaning that an highly reflective scatterer in all directions would translate to whiter colors, whereas an obtect that tends to reflect more in the HH direction would appear more red. With this the network could learn its dependencies, which could possible give more information to the network to correctly color the image.

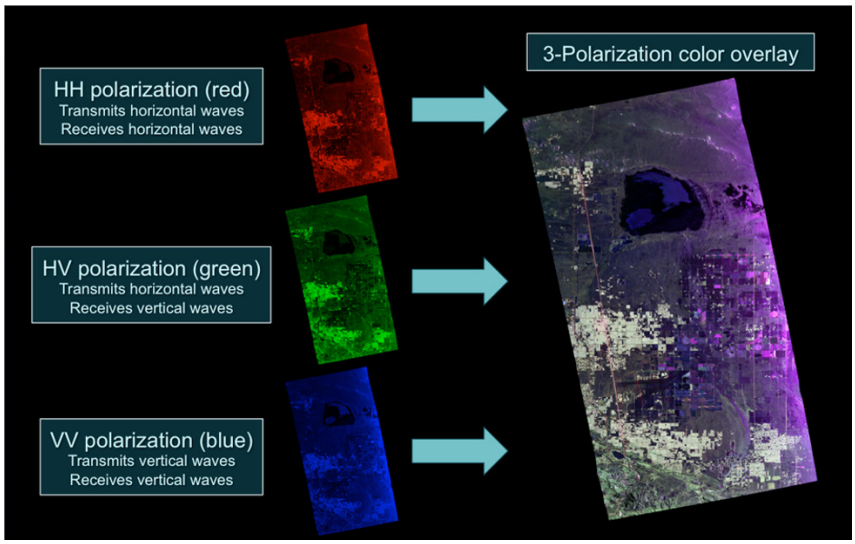

Figure 7.7 Combination of polarizations to create a pseudo-colored image to highlight certain features

Phase information could also be used to heighten the network performance as this domain captures the phase differences of the backscatter image. Phase information could be used to correctly identify and correctly color moving targets, as these would translate to a Doppler shift and therefore slight changes in phase, when compared to the sent signal. Phase information also allows us to obtain the relative distance of the target and its geometry as the phase differences between the received signal relative to the transmitted signal, captures the phase shift accumulated by the signal as it propagates to the target and back is proportional to the path length, which is affected by the target distance and the number of multi-paths that the signal had to perform due to geometry to reach the remote sensing system.

In SAR systems, as the radar platform moves it collects data from slightly different positions and time. By coherently processing these signals, SAR creates a two-dimensional image

where the phase variations correspond to the target's changes over time. The phase difference, usually called interferometry images, between two SAR images taken from slightly different positions can be used to derive elevation information. This phase difference arises due to differences in the path length caused by the terrain's elevation. Different target geometries also affect how the radar signal is scattered, where due to geometry the signals can reflect off multiple surfaces which introduce additional phase shifts that can constructively or destructively interfere with the received signal. This could give additional context to the network as inhomogeneous terrain such as the sea could have slight variations in phase, from which it could create better representations of the optical remote sensing system.

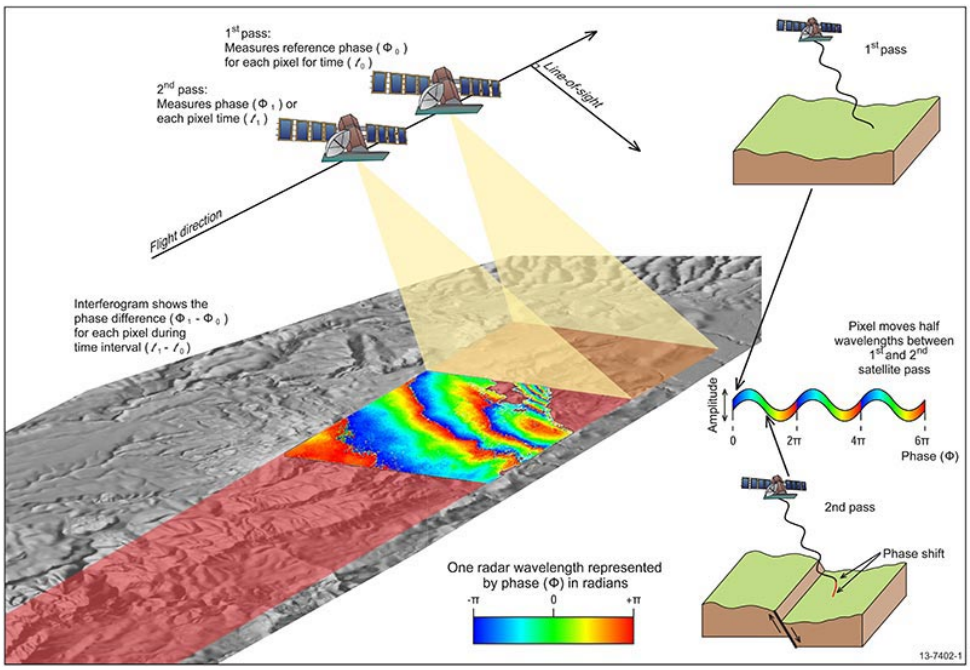

Figure 7.8 Topography mapping using interferometry images due to differences in terrain elevation, causing a phase shift

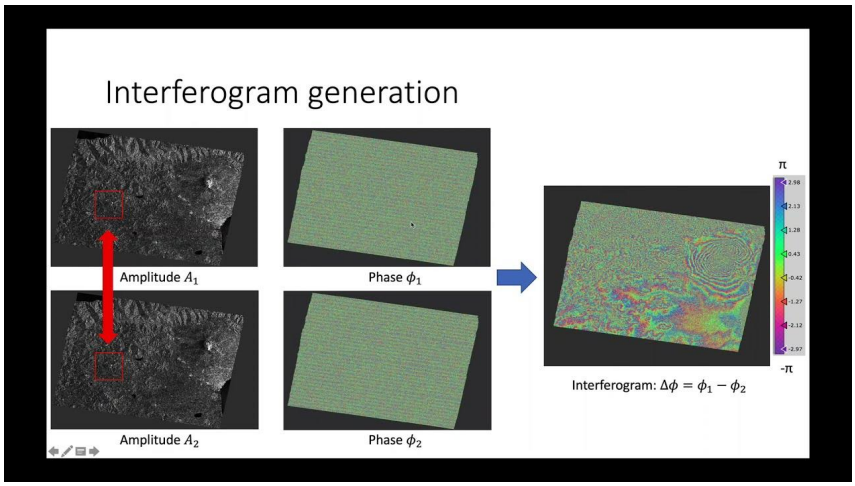

Figure 7.9 Interferometry image generation

With more recent advancements in SAR systems it is now possible to obtain high-resolution

SAR images, reaching spacial resolutions of a few centimeters. Such systems are more expansive and usually more restricted and offer images, where the level of detail can give a clearer picture of the scene and allow an easier comprehension an analysis by non-trained human operators. However these systems will always lack color information due to the nature of the system, no matter the resolution, that for certain missions can be critical. The developed model could be trained on such images, where due to the level of detail the resulting image would also be better. However this would also require to either train on images with more data points if the aim is to keep the same relative level of zoom than before, as for lower resolution systems, each pixel would translate to a bigger area for the same scene. Or it would require training on smaller geographical areas if the aim is to train the model using the same resolution images (256x256). Another consideration is the spacial resolution and quality of the optical system, as if the system does not have sufficient resolution, this could lead to worse results as the disparity in inputted image and outputted image resolution would require upscale of the optical image, and could lead to either blurry images or the model ignoring certain pixels.

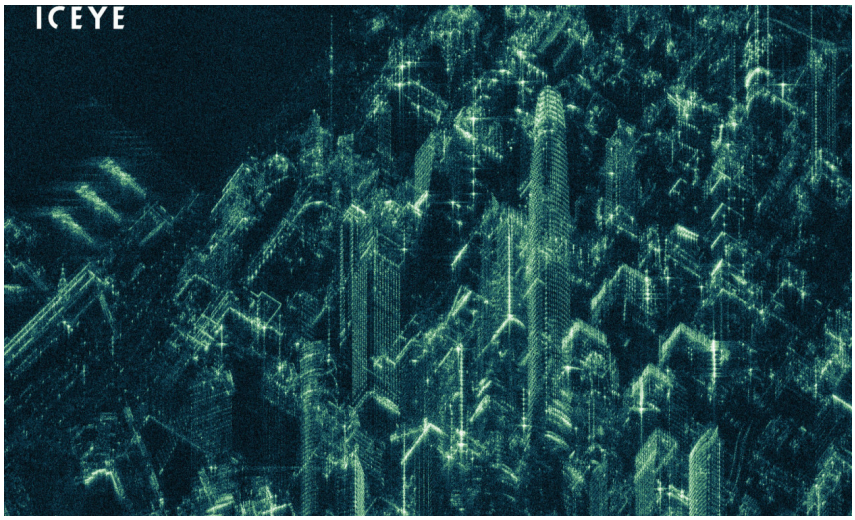

Figure 7.10 High-resolution SAR system of urban sites, using a resolution of 25cm obtained using ICEYE satelitte constelation

Newer developments in neural networks also offer different approaches to the problem, where new networks such as Kolmogorov-Arnold Networks (KAN) shift the training paradigm [48]. While traditional NN have fixed activation functions represented by a node, simulating neurons and learnable weights on the connecting edges between nodes, the KAN model does not implement these trainable weights. Instead the model has learnable activation functions on its edges and its nodes consist only in sums of the inputs [48]. These activation functions can be decomposed of multiple single 1D functions parametrized as a spline, which is a mathematical smooth curve that passes through control points or knots, which can be decomposed into a composition of polynomial pieces on subintervals [48]. The combination of each piece creates the smooth 1D spline, that then are combined to create the activation function. These splines are learned during training, and when summed create the variable activation function. This allows the model to better define its decision regions, which could lead to better pattern

recognition and generalization. Using this approach KANs do not need to implement learnable weight matrices as these are replaced by the spline, which are less computationally expensive when compared to NN[48].

Model	Multi-Layer Perceptron (MLP)	Kolmogorov-Arnold Network (KAN)
Theorem	Universal Approximation Theorem	Kolmogorov-Arnold Representation Theorem
Formula (Shallow)	$f(\mathbf{x}) \approx \sum_{i=1}^{N(\epsilon)} a_i \sigma(\mathbf{w}_i \cdot \mathbf{x} + b_i)$	$f(\mathbf{x}) = \sum_{q=1}^{2n+1} \Phi_q \left(\sum_{p=1}^{n} \phi_{q,p}(x_p) \right)$
Model (Shallow)	(a) *fixed* activation functions on *nodes*; *learnable* weights on *edges*	(b) *learnable* activation functions on *edges*; sum operation on *nodes*
Formula (Deep)	$\mathrm{MLP}(\mathbf{x}) = (\mathbf{W}_3 \circ \sigma_2 \circ \mathbf{W}_2 \circ \sigma_1 \circ \mathbf{W}_1)(\mathbf{x})$	$\mathrm{KAN}(\mathbf{x}) = (\mathbf{\Phi}_3 \circ \mathbf{\Phi}_2 \circ \mathbf{\Phi}_1)(\mathbf{x})$
Model (Deep)	(c) MLP(x): $\mathbf{W}_3$, σ_2 (*nonlinear, fixed*), $\mathbf{W}_2$, σ_1, $\mathbf{W}_1$ (*linear, learnable*), x	(d) KAN(x): $\mathbf{\Phi}_3$, $\mathbf{\Phi}_2$ (*nonlinear, learnable*), $\mathbf{\Phi}_1$, x

Figure 7.11 MLP and KAN comparison

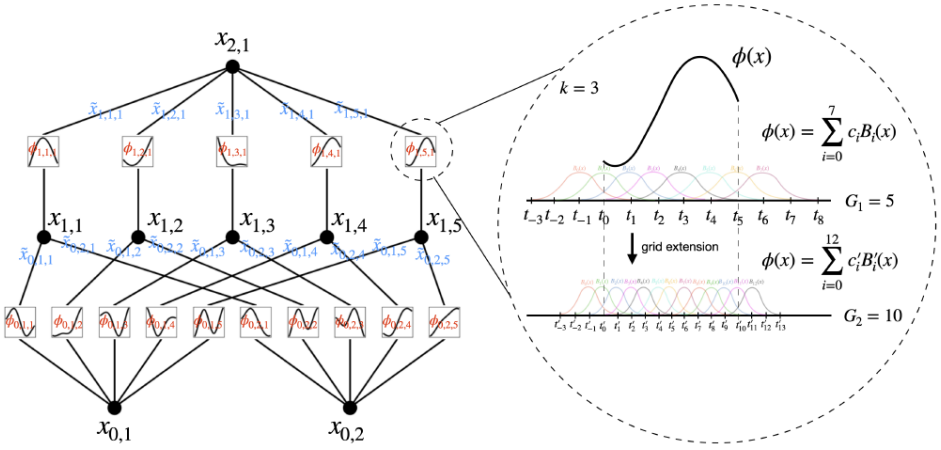

Figure 7.12 KAN learnable activation function decomposed into spline functions

Bibliography

[1] Xiaodong Zhou, Chunhua Zhang, and Song Li. A perceptive uniform pseudo-color coding method of sar images. In *2006 CIE International Conference on Radar*, pages 1–4, 2006.

[2] Shefali Aggarwal. *PRINCIPLES OF REMOTE SENSING*, pages 23–38. World Meteorological Organization (WMO), Dehra Dun, India, 2003.

[3] Ulrik Mårtensson. Introduction to remote sensing and geographical information systems, 2011.

[4] Thomas Lillesand, Ralph Kiefer, and Jonathan Chipman. *Remote Sensing and Image Interpretation (Fifth Edition)*, volume 146. 01 2004.

[5] Anders Knudby. Emr interactions with the earth's atmosphere and surface, 2023.

[6] C. A. Wiley. Synthetic aperture radars. *IEEE Transactions on Aerospace and Electronic Systems*, AES-21(3):440–443, May 1985.

[7] Steven W. Smith. Chapter 11 - fourier transform pairs. In Steven W. Smith, editor, *Digital Signal Processing*, pages 209–224. Newnes, Boston, 2003.

[8] Hélène M. Oriot. Moving target detection on sar images. 2014.

[9] Geoawesomeness. Moving objects and their displacement in sar images, 2024.

[10] S.A.S. Werness, W.G. Carrara, L.S. Joyce, and D.B. Franczak. Moving target imaging algorithm for sar data. *IEEE Transactions on Aerospace and Electronic Systems*, 26(1):57–67, 1990.

[11] Xiaohong Chen, Qian Sun, and Jun Hu. Generation of complete sar geometric distortion maps based on dem and neighbor gradient algorithm. *Applied Sciences*, 8(11), 2018.

[12] Natural Resources Canada. Radar image distortions - fundamentals of remote sensing, 2024.

[13] Kunihiko Fukushima. Neocognitron: A hierarchical neural network capable of visual pattern recognition. *Neural Networks*, 1(2):119–130, 1988.

[14] James Anderson. *McCulloch–Pitts Neurons*. 01 2006.

[15] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning representations by back-propagating errors. *Nature*, 323:533–536, 1986.

[16] Y. Bengio and Yann Lecun. Convolutional networks for images, speech, and time-series. 11 1997.

[17] Georgia Koukiou. Perceptually optimal color representation of fully polarimetric sar imagery. *Journal of Imaging*, 8(3), 2022.

[18] Qian Song, Feng Xu, and Ya-Qiu Jin. Radar image colorization: Converting single-polarization to fully polarimetric using deep neural networks. *IEEE Access*, 6:1647–1661, 2018.

[19] Shruti Gupta, Dharmendra Singh, and Sandeep Kumar. Fusion of color histograms using pca for sar data classification. In *2015 National Conference on Recent Advances in Electronics Computer Engineering (RAECE)*, pages 244–247, 2015.

[20] Oliver Lang and Parivash Lumsdon. Integration of colorized single-pol sar data into optical image mosaics. In *2011 IEEE International Geoscience and Remote Sensing Symposium*, pages 2785–2788, 2011.

[21] Guang Ji, Zhaohui Wang, Lifan Zhou, Yu Xia, Shan Zhong, and Shengrong Gong. Sar image colorization using multidomain cycle-consistency generative adversarial network. *IEEE Geoscience and Remote Sensing Letters*, 18(2):296–300, 2021.

[22] Zhe Guo, Haojie Guo, Xuewen Liu, Weijie Zhou, Yi Wang, and Yangyu Fan. Sar2color: Learning imaging characteristics of sar images for sar-to-optical transformation. *Remote Sensing*, 14(15), 2022.

[23] M. Schmitt, L. H. Hughes, M. Körner, and X. X. Zhu. Colorizing sentinel-1 sar images using a variational autoencoder conditioned on sentinel-2 imagery. *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, XLII-2:1045–1051, 2018.

[24] M. Schmitt, L. H. Hughes, M. Körner, and X. X. Zhu. Colorizing sentinel-1 sar images using a variational autoencoder conditioned on sentinel-2 imagery. *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, XLII-2:1045–1051, 2018.

[25] Lianfa Li, Ying Fang, Jun Wu, and Jinfeng Wang. Autoencoder based residual deep networks for robust regression prediction and spatiotemporal estimation, 2018.

[26] Lianfa Li. Deep residual autoencoder with multiscaling for semantic segmentation of land-use images. *Remote Sensing*, 11(18), 2019.

[27] Xinyu Bai and Feng Xu. Sar to optical image translation with color supervised diffusion model, 2024.

[28] Xinyu Bai and Feng Xu. Accelerating diffusion for sar-to-optical image translation via adversarial consistency distillation, 2024.

[29] Yucheng Pan, Liheng Zhong, Jingdong Chen, Heping Li, Xianlong Zhang, and Bin Pan. Sar image despeckling based on denoising diffusion probabilistic model and swin transformer. *Remote Sensing*, 16(17), 2024.

[30] Chayakrit Krittanawong, Kipp Johnson, Robert Rosenson, Zhen Wang, Mehmet Aydar, Usman Baber, James Min, W Tang, Jonathan Halperin, and Sanjiv Narayan. Deep learning for cardiovascularmedicine: A practical primer. *European heart journal*, 40, 02 2019.

[31] Andrinandrasana David Rasamoelina, Fouzia Adjailia, and Peter Sinčák. A review of activation function for artificial neural network. In *2020 IEEE 18th World Symposium on Applied Machine Intelligence and Informatics (SAMI)*, pages 281–286, 2020.

[32] Zahraa Kadhim, Hasanen Abdullah, and Khalil Ghathwan. Artificial neural network hyperparameters optimization: A survey. *International Journal of Online and Biomedical Engineering (iJOE)*, 18:59–87, 12 2022.

[33] Mohaimenul Azam Khan Raiaan, Sadman Sakib, Nur Mohammad Fahad, Abdullah Al Mamun, Md. Anisur Rahman, Swakkhar Shatabda, and Md. Saddam Hossain Mukta. A systematic review of hyperparameter optimization techniques in convolutional neural networks. *Decision Analytics Journal*, 11:100470, 2024.

[34] Shreyas Fadnavis. Image interpolation techniques in digital image processing: an overview. *International Journal of Engineering Research and Applications*, 4(10):70–73, 2014.

[35] X Engineer. Bilinear interpolation. `https://x-engineer.org/bilinear-interpolation`, 2023.

[36] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation, 2015.

[37] Guoping Xu, Xiaxia Wang, Xinglong Wu, Xuesong Leng, and Yongchao Xu. Development of skip connection in deep neural networks for computer vision and medical image analysis: A survey, 2024.

[38] Copernicus. Copernicus sentinel-1 mission overview. `https://sentiwiki.copernicus.eu/web/s1-mission`, 2023.

[39] Dirk Geudtner, Ramón Torres, Paul Snoeij, Malcolm Davidson, and Björn Rommen. Sentinel-1 system capabilities and applications. In *2014 IEEE Geoscience and Remote Sensing Symposium*, pages 1457–1460, 2014.

[40] Copernicus. Copernicus sentinel-1 products overview. `https://sentiwiki.copernicus.eu/web/s1-products`, 2023.

[41] Copernicus. Copernicus sentinel-2 mission overview. https://sentiwiki.copernicus.eu/web/s2-mission, 2023. Accessed: October 6, 2024.

[42] Copernicus. Copernicus sentinel-2 data products. https://sentinels.copernicus.eu/web/sentinel/missions/sentinel-2/data-products, 2023.

[43] M. Schmitt, L. H. Hughes, and X. X. Zhu. The sen1-2 dataset for deep learning in sar-optical data fusion. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, IV-1:141–146, 2018.

[44] J. Lee, L. Jurkevich, Piet Dewaele, Patrick Wambacq, and A. Oosterlinck. Speckle filtering of synthetic aperture radar images: A review. *Remote Sensing Reviews*, 8, 02 1994.

[45] Nupur Saxena and Neha Rathore. A review on speckle noise filtering techniques for sar images. *International Journal of Advanced Research in Computer Science and Electronics Engineering (IJARCSEE)*, 2(2), 2013.

[46] Irena Hajnsek and Yves-Louis Desnos. *Polarimetric Synthetic Aperture Radar Principles and Application: Principles and Application*. 01 2021.

[47] Junrong Qu, Xiaolan Qiu, Wei Wang, Zezhong Wang, Bin Lei, and Chibiao Ding. A comparative study on classification features between high-resolution and polarimetric sar images through unsupervised classification methods. *Remote Sensing*, 14(6), 2022.

[48] Ziming Liu, Yixuan Wang, Sachin Vaidya, Fabian Ruehle, James Halverson, Marin Soljačić, Thomas Y. Hou, and Max Tegmark. Kan: Kolmogorov-arnold networks, 2024.