POLITECNICO DI MILANO
Master's Degree in Computer Science and Engineering
Dipartimento di Elettronica, Informazione e Bioingegneria

Deep Learning for Sea-Ice Classification on Synthetic Aperture Radar (SAR) Images in Earth Observation

Classification Using Semi-Supervised Generative Adversarial Networks on Partially Labeled Data

Supervisor: Prof. Piero Fraternali

Master's Thesis by:
Francesco Staccone, Student ID 914740

Academic Year 2019-2020

To my beloved ones

Abstract

Earth Observation is the gathering of information about planet Earth's system via Remote Sensing technologies for monitoring land cover types and their changes. One of the most attractive use cases is the monitoring of polar regions, that recently observed some dramatic changes due to global warming. Indeed drifting ice caps and icebergs represent threats to ship activities and navigation in polar areas, and the risk of collision with land-derived ice highlights the need to design a robust and automatic Sea-Ice classifier for delivering up-to-date and accurate information.
To achieve this goal, satellite data such as Sentinel-1 Synthetic Aperture Radar images from the European Union's Copernicus program can be given in input to a Deep Learning classifier based on Convolutional Neural Networks capable of giving the content categorization of such images as output. For the task at hand, the availability of labeled data is generally scarce, therefore the problem of learning with partially labeled data must be faced. Therefore, this work aims at leveraging the broader pool of unlabeled satellite data available to open up new classification solutions.
This thesis proposes a Semi-Supervised Learning approach based on Generative Adversarial Networks, that takes in input both labeled and unlabeled data and outputs the classification results exploiting the knowledge retrieved from both the data sources. Its classification performance is evaluated and it is later compared with the Supervised Learning approach and the Transfer Learning approach based on pre-trained networks.
This work empirically proves that the Semi-Supervised Generative Adversarial Networks approach outperforms the Supervised Learning method, improving its Overall Accuracy by at least 5% in configurations with less than 100 training labeled samples available in the use cases under evaluation, achieving performance comparable to the Transfer Learning approach and even overcoming it under specific experimental configurations. Further analyses are then executed to highlight the effectiveness of the proposed solution.

Sommario

L'Osservazione della Terra consiste nel collezionare informazioni attraverso il Telerilevamento al fine di monitorare le variazioni della superficie terrestre. Uno dei casi più interessanti è il monitoraggio delle zone polari, poiché iceberg e calotte di ghiaccio alla deriva comportano una minaccia per le attività di navigazione nelle zone polari. Pertanto, il rischio di collisione con esse rende necessario un sistema automatico e robusto di classificazione Mare-Ghiaccio che produca informazioni accurate.
A tale scopo, dati satellitari come immagini Radar ad Apertura Sintetica raccolte da Sentinel-1 per il programma Europeo Copernicus possono essere fornite in input ad un classificatore ad Apprendimento Profondo basato su Reti Neurali Convoluzionali capace di produrre in output la classificazione di tali immagini. In tale contesto la disponibilità di dati etichettati è limitata ed è pertanto necessario apprendere con dati parzialmente etichettati. Questo lavoro punta a sfruttare il numeroso insieme di dati satellitari non etichettati per proporre nuove soluzioni di classificazione.
Questa tesi propone un approccio di Apprendimento Semi-Supervisionato basato su Reti Generative Antagoniste, che riceve in input sia dati etichettati che non etichettati e produce in output i risultati di classificazione facendo leva dulla conoscenza acquisita da entrambe le sorgenti. Le performance di classificazione raggiunte da tale approccio sono valutate e poi comparate con l'Apprendimento Supervisionato e l'Apprendimento per Trasferimento basato su reti pre-addestrate.
Questo lavoro dimostra empiricamente che l'approccio Semi-Supervisionato basato su Reti Generative Antagoniste supera in prestazioni l'Apprendimento Supervisionato, migliorandone l'Accuratezza Complessiva di almeno il 5% in configurazioni con meno di 100 dati etichettati disponibili durante la fase di apprendimento nei casi d'uso valutati, raggiungendo performance comparabili a quelle dell'Apprendimento per Trasferimento e superandole in configurazioni sperimentali specifiche. Infine, ulteriori analisi sono condotte al fine di evidenziare l'efficacia della soluzione proposta.

Contents

List of Figures

List of Tables

Acknowledgments

I would like to take this opportunity to express my deepest gratitude to all those who supported me during my whole academic career and who gave me the chance to successfully carry out this thesis work.
At first, I would like to thank my dear family with endless gratitude, that always supported me throughout my education and life by all means.
A special thanks goes to my lovely Francesca, for believing in me and making each day of this journey a better day, even from far away.
Furthermore, I wish to thank my awesome friends Andrea, Antonio, Luca, Manuel, Mattia, Paolo, Rino and Stefano, with whom I have shared the most of this journey and a series of unforgettable experiences.
I am sincerely grateful to my supervisors Prof. Piero Fraternali from Politecnico di Milano and Prof. Vladimir Vlassov and Tianze Wang from KTH Royal Institute of Technology for offering me the opportunity to work on this project and for their constant guidance and genuine support throughout the whole development of the thesis.
Finally, a big thanks goes to Politecnico di Milano, KTH Royal Institute of Stockholm and EIT Digital for offering me the chance of living this wonderful journey and obtaining this valuable degree.

Milan, July 1, 2020

Chapter 1

Introduction

In this chapter, the context, motivation, task and overall purpose of the thesis are presented. An introduction of the proposed solution and its delimitations are also provided, together with the research methods used during the whole work and its ethical and sustainability considerations.

1.1 Context

Classification is the process of labeling every sample contained in a pool of items under evaluation, with each label representing the category to which the specific object belongs [1]. For instance, the pool can be composed of images, website user behaviours or customer purchase patterns coming from different categories: images of cats and dogs, legitimate or malicious users, new or loyal customers must be discriminated and labeled accordingly.
Classification aims at predicting which input data belong to which class to improve the performance of the decision-making process in a specific use case. The prediction on unseen data in the data science world is made possible thanks to models trained on past data following several widely known Machine Learning (ML) and Deep Learning (DL) approaches.
Classification of images, in particular, is a widely used practice in the Computer Vision area and several other related domains, including Earth Observation (EO).
The improvement of Artificial Intelligence (AI) techniques in the last decades has allowed the scientific community to reach high-quality image classification results leveraging on powerful architectures such as Convolutional Neural Networks (CNNs). They are Artificial Neural Networks (ANNs) capable of extracting features from images taking advantage of local spatial coherence, reducing dramatically the number of parameters and opera-

tions needed compared to fully connected networks by using convolution on patches of adjacent pixels.
This makes them suitable for EO classification problems, in which satellite data such as Synthetic Aperture Radar (SAR) images collected by Sentinel-1 or optical images collected by Sentinel-2 satellites must be processed to categorize their content. In this context, developing innovative products for addressing EO-related problems has become a prime necessity for the European Commission program Copernicus [2] and other related projects such as ExtremeEarth [3], that leverage on Sentinel-1 SAR data [4] to provide automatic classifiers and more.
Research in both image and EO classification is indeed growing and promising, but there is still much room for improvement in terms of how CNNs can be used to achieve good results with scarcely available labeled training data, namely in a non-purely Supervised Learning (SL) environment.
In recent times different approaches involving CNNs such as Transfer Learning (TL) or Semi-Supervised Learning (SSL) have been tested out to overcome the scarcity of labeled data in Computer Vision classification tasks, obtaining encouraging results.
The EO classification scenario belongs to the same kind of tasks mentioned above, being characterised by a huge availability of unlabeled images collected by the satellites hour by hour, while just relatively few hand-labeled datasets are at disposal.
In this context, wondering which method is the best to face the shortage of labeled data in Earth Observation classification scenarios seems to be a challenging task, likewise interesting: it is even more so if you deal with polar areas images and you need to detect the presence of drifting ice caps or icebergs, representing a big threat for ship activities and navigation, to overcome the risk of collision.

1.2 Problem

The easiest and fastest way to face a classification problem in general, and an EO classification problem in particular, is to implement a SL classifier that during the training phase receives in input some labeled data and during the prediction phase outputs the categorization of the unseen data it deals with.
SL works well under one strong condition: enough labeled data are available at training time. This means that somebody must have built the dataset and hand-labeled it, or implemented an algorithm able to automatically perform the labeling. The first approach is extremely time-consuming in case of big

datasets, while the second one is very demanding and still requires some ground truth labels in input or the interaction by humans, ending up in a semi-automatic approach, namely Active Learning.
Having to deal with poorly available labeled data is not that rare, and implementing a purely SL approach within this scenario can lead to two distinct problems:

- Loss of large piece of information
- Increased risk of overfitting the data

The model obtained in this scenario suffers from the loss of a large piece of information due to its inability to exploit the widely available unlabeled data, that instead bring great knowledge with them. To exploit the unlabeled data the SSL approach has been introduced, whose possible applications are widely described in several books and literature surveys [5][6] showing evidence of its better performance than the purely SL method with small labeled data. Figure 1.1 gives an intuition of how the model performance varies with the size of the labeled samples.
Useful information could be also retrieved from different labeled datasets, if wider, and with common features to the data at disposal, such as another dataset of images in case of an image classification task. This method is commonly known as Transfer Learning [7], and its results are interesting from a dual perspective compared to the purely SL approach. They can be either positive or negative, depending on the level of relation between the source task and the target task. This behaviour is visible in Figure 1.2.

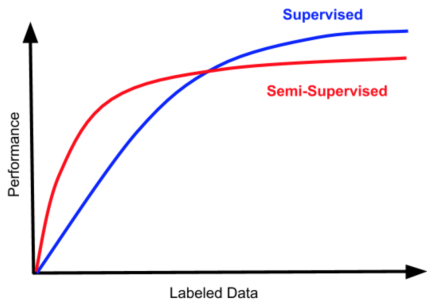

Figure 1.1: SSL performance at varying amount of labeled data, image available at [8]

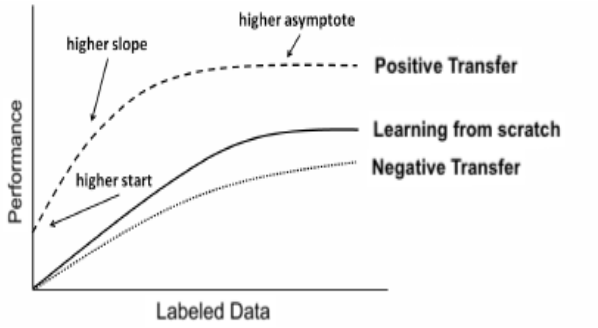

Figure 1.2: TL performance at varying amount of labeled data, image available at [9]

Moreover, the purely SL model is prone to overfit the small labeled dataset used as training set and may not be able to generalize with new unseen data

given in input during the prediction phase, resulting in a great prediction error despite the small training error achieved [10][11]. In this regard, Figure 1.3 clearly shows the inability of such a model to generalize with few training labeled data, causing a significant gap between prediction and training errors.

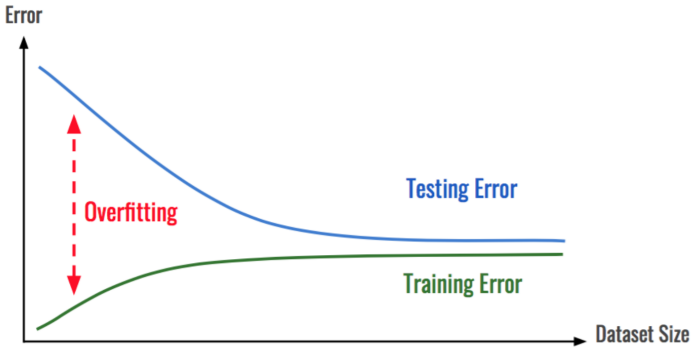

Figure 1.3: Error in relation to training set size, image available at [12]

Given the limitations of the SL approach that have been introduced above, it becomes necessary to conduct an analysis of the possible alternative approaches through the realization of a comparative framework, propose a novel technique and establish if it is possible to overcome the problem of learning with poorly available labeled data in an EO scenario.
The results of this extensive study are crucial for the reliable prediction of icebergs and ice caps in the polar regions, reachable only through the development of a robust Sea-Ice classifier whose knowledge is not just merely based on labeled data at disposal.

1.3 Purpose and Research Question

The availability of labeled satellite images is generally poor when compared with the huge pool of unlabeled images collected by satellites every hour daily. This applies also to satellite images representing polar regions, therefore the exploitation of unlabeled images is worthy of further investigation due to its potential benefits in terms of the amount of captured information and thus reliability of classification. The procedure consists of extracting knowledge, namely representations, before performing the Supervised step. This additional Semi-Supervised step brings to the main research question addressed by this work:

Given a set of polar satellite images and the task of classifying their content with scarcely available labeled data, this work aims to empirically study whether Semi-Supervised Learning based on Generative Adversarial Networks can outperform purely Supervised Learning delivering better performance. Semi-Supervised Learning is further compared with Transfer Learning based on pretrained networks.

The initial underlying hypothesis on which the research project is developed is that, when dealing with limited labeled images, the traditional SL classification approach can be outperformed by methods that leverage other widely available unlabeled images to build representations useful for the classification tasks.
Thus, a pipeline composed of two actions is proposed: first, extracting an initial set of representations from widely available unlabeled EO data (SSL) or from widely available labeled data coming from a related classification problem (TL); second, transferring the derived representations into a classifier, along with their few class labels, to train the system and produce the final result.
The description, discussion and comparison of the assessed SL, SSL and TL approaches are thoroughly introduced in the next chapters.

1.4 Goals and Research Objective

In order to answer the above-presented research question, a set of precise goals must be identified:

- **G1:** Undertake a deep study of the literature and the related work already present in the field, in order to have knowledge about the existing problems and solutions in the fields.
- **G2:** Identify a knowledge gap in the literature and propose an innovative approach that tackles the identified problem trying to fill the gap, in order to address the research question.
- **G3:** Investigate thoroughly the characteristics of the datasets under consideration and of the models under assessment, defining a baseline, in order to focus the problem and pave the way to the proposed solution.
- **G4:** Propose a solution to the problem, implement it and compare with the baseline and the most promising approaches from the liter-

ature. Then, discuss and evaluate the obtained results through the appropriate metrics, in order to introduce the main outcomes.

- **G5:** Draw some conclusions out of the comparison and outline possible improvements and future work.

These goals are presented to comply with a chronological order of steps to follow, defining the general research workflow that enables the achievement of the ultimate research objective: show that through the exploitation of unlabeled data it is possible to outperform the traditional SL approach.

1.5 Research Methodology

This work has been supported with scientific and empirical research methodologies [13] since the beginning of the research. Systematic steps have been fulfilled during the data preparation, models selection and execution of the experiments to assure validity, reliability and replication of the results. In this regard, both the data and the specific architectures to be used have been chosen as a result of a comprehensive literature review. The implementation choices that have been made and the evaluation of the results rely on the knowledge acquired during the whole project, and are thoroughly presented and discussed in the next chapters.
A quantitative method has been followed and enriched by supplementary qualitative evaluations, aiming to reach conclusions by experimenting on two different datasets. This allowed a full understanding and comparison of the proposed algorithms' performance on a wider range of cases, instead of concentrating on just one dataset. A deductive approach was used to verify the hypothesis, leveraging on the obtained results.
Since the identified research question aimed at studying whether an improvement in EO classification would be possible, the focus has been in the first place on implementing the already existing techniques (SL and TL) and in the second place to implement a novel SSL technique, that is introduced in the next section, benchmarking their results at the very end. The benchmarking was mainly based on Overall Accuracy (OA) and other performance metrics such as Precision, Recall and F_1 score, since they are the most commonly used metrics in the classification literature and most of the results deriving from the existing algorithms are provided through bounds defined on these metrics.
To run the experiments the same settings for each of the benchmarked approaches have been reproduced, such as the same train, validation and test

set sizes and the same maximum number of epochs and maximum batch sizes for the training phase, to fairly compare their results.

1.6 Proposed Solution and Contributions

This thesis addresses the presented problem by proposing a SSL technique that leverages on a revised version of the Generative Adversarial Networks (GANs). The proposed method, known as Semi-Supervised Generative Adversarial Network (SGAN) [14], aims to take into account not only the information extracted from the labeled data but also the knowledge contained in the unlabeled images at disposal and exploit them to extract features useful to handle the classification problem in the last stage.
The idea is to adopt this approach in the sea ice SAR satellite imagery world, introducing an extension of the original SGAN technique to solve the classification problem in the SAR domain which is characterized by wide availability of unlabeled images.
The DL architecture proposed, whose overview is shown in Figure 4.5, is a variant of the GAN architecture capable of performing a final classification. The learning process, as happens with traditional GANs, is based on the competition between the Generator and the Discriminator, which undergoes relevant variations from its original form. These variations involve creating logically separated supervised and unsupervised models for the Discriminator, attempting to reuse the output layers of the former as input to the latter.

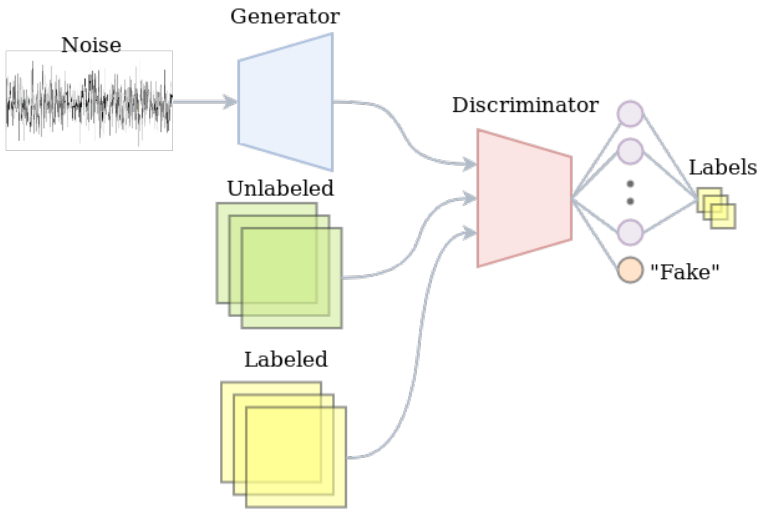

Figure 1.4: SGAN architecture, image available at [15]

The approach is based on the definition of the SSL model by Tim Salimans et al. [14] entitled "Improved Techniques for Training GANs." In the paper, they describe an implementation where at first the supervised model is created with K output classes and a softmax activation function, while in a second step the unsupervised model takes the output of the supervised one before the softmax and computes a normalized sum of the exponential outputs to produce its own output, namely the "real" or "fake" label. The "fake" label represents the $(K + 1)$-th output class of the Discriminator.
To answer the research question, this method is compared with the traditional SL approach implemented through a SAR-specific CNNs architecture, namely the DeepSAR-Net [16], and benchmarked with other approaches such as the VGG16-based [17] TL approach.
This comparison highlights the effectiveness of using the available unlabeled data in addition to the labeled ones, given the higher performance achieved by the SGAN approach compared to the traditional SL one. In particular, it improves the OA achieved by the SL approach by at least 5% in configurations with less than 100 labeled samples available at training time in the use cases analyzed.
More in general, this work also confirms the impact and the usefulness of including additional knowledge coming from the same domain (SSL) or a related domain (TL). Specifically, the SSL approach obtains performance comparable to the TL one, but with the great advantage of avoiding the TL pre-training step, hence saving great human effort for data hand-labeling, relevant training time and related CO_2 emission.
Moreover, a framework containing other variants of the above-mentioned approaches, such us the Fine-tuned (FT) variant of the TL method, the concatenation of the features extracted by the CNN with additional information (e.g. geographic coordinates) or the Support Vector Machine (SVM)-based classifier, has been developed for the sake of comparison.
Such a framework further enriches the contribution of this thesis, showing the capability of the SVM-based classifier of dealing with small labeled datasets, the usefulness of the concatenation of additional information to the CNN features and finally the higher performance of the TL-FT approach compared to the traditional TL approach thanks to its high-level features tailored on the target use cases.
Finally, images produced by the SGAN Generator open up to future possibilities of performing Data Augmentation and Label Refinery techniques to further improve the achieved performance.
To sum up, the main contributions provided by this work are the following:

- The extensive study and comparison of existing DL approaches in the EO domain have been performed, with the focus on the Sea-Ice classification problem with scarcely available labeled data;
- The extension of the SGAN architecture [14] based on the DeepSAR-Net [16] has been introduced. It outperforms the traditional DeepSAR-Net [16] SL approach by at least 5% in configurations with less than 100 labeled samples available at training time in both the use cases under evaluation, and competes with the TL approach based on VGG16 [17];
- The comparison of the training times and CO_2 emissions related to the SL, SSL and TL approaches has been proposed, pointing out the great advantage of the SSL approach of avoiding the TL pre-training step, that is computationally-heavy;
- The generative capabilities of the DeepSAR-Net-SGAN approach have been highlighted, opening up new Data Augmentation and Label Refinery possibilities;
- The performance enhancement related to DeepSAR-Net-SGAN when increasing the size of unlabeled data available has been shown;
- The SVM classifier capability of generalizing with datasets of small size has been proved, enhancing the SL approach performance especially when few labeled data are available;
- The performance enhancement in the traditional SL approach due to additional information given (e.g. geographic coordinates) has been highlighted, especially when few labeled data are available.

1.7 Delimitations

This thesis aims to explore the effectiveness of the SSL approach based on GANs focusing on SAR satellite images that contain mainly icebergs and ocean scenes collected under specific acquisition modes. Nevertheless, the SSL approach introduced by this thesis is expected to perform well also in other SAR-related tasks where satellite vignettes are characterized by footprints of similar size compared to the polar images ones. In this regard, the generality of the method could be tested out on the Copernicus [2] Food Security use case with the aim of classifying crops, performing crop yields estimation or continuously monitoring the water and nutrient availability in crops development.

Still, the more general purpose of defining a method for any kind of SAR or satellite imagery is only partially addressed in this work since just part of the method can contribute to it.
Moreover, being this a first attempt of applying a SGAN architecture on icebergs satellite imagery, areas for improvements still need to be explored. For instance, this research studies the training and the deployment of such a model in the ground stations which gather the images collected by the satellites, but does not evaluate its direct deployment on AI modules-equipped satellites. This kind of investigations is left for future work.

1.8 Ethics and Sustainability

The four principles of Ethics presented in [18] - Autonomy, Non-maleficence, Beneficence and Justice - are very influential not only in the field of medical ethics in which they were originally formulated, thus it is possible to evaluate the ethical implications of this thesis leveraging on this framework. The main ethical principles affected by this work are:

- Non-maleficence: this project aims to not harm anyone, instead it aims at helping military ships and civilian sailors to avoid collisions with icebergs and ice caps. Despite its good intentions, the system may output wrong predictions, therefore it is necessary to take care of possible false negatives (not recognized icebergs or ice caps). Moreover, if the system is hacked by any malicious user it could be used to harm people through the recommendation of wrong suggestions.

- Respect for autonomy: the system outputs a suggestion to the human, who is still able to make his own decision.

The United Nations have defined the 17 Sustainable Development Goals [19], whose aim is to provide a blueprint for peace and prosperity for people and the planet in the future. This thesis takes into account these goals and it is mainly related to some of them:

- 8^{th} and 12^{th} Sustainable Development Goals "Decent Work and Economic Growth" and "Consumption": the project may cause additional computational power and electricity consumption compared to the traditional SL approach since it leverages also unlabeled data, but it allows to avoid hand-labeling of images which may be very expensive.

- 13^{th} Sustainable Development Goal "Climate Action": the aim of the project is to avoid accidents and collisions that might damage the nature and its ecosystem.

Besides, the scope of this thesis is in line with the Copernicus program [2] overall aim, which is to help humans to better understand the planet Earth and sustainably manage its environment.
In this regard, the Copernicus program is related to the Thematic Exploitation Platforms (TEPs) activity of the European Space Agency (ESA). The TEPs are virtual work environments that provide access to EO data related to specific areas such as geohazards, coastal, hydrology, forestry, polar, food security and urban themes together with computing resources required to work with them. In particular, this project is linked to the Polar Regions TEP.

1.9 Thesis Structure

The structure of this research is hereby presented:

- **Chapter 1** presents the *Introduction* to this research work, useful to let the reader perceive the motivation and the purpose of the study.
- **Chapter 2** presents the *Background* knowledge related to this project, necessary to have a comprehensive understanding of the problem under assessment and the proposed solution.
- **Chapter 3** presents the *Related work* conducted by other authors that has proved to be a source of inspiration to deeper investigate the scenario under consideration and provide a fancier approach. Together with Chapter 2, it contributes to the achievement of goals **G1** and **G2**.
- **Chapter 4** presents the *Deep Learning model architectures* that have been fundamental for the development of the research project, in such a way that readers can understand and later reproduce the proposed solutions.
- **Chapter 5** presents the *Experimental evaluation* conducted and the related results obtained during the research work, describes the exploited data and provides an evaluation of the main outcomes. Together with Chapter 4, it contributes to the achievement of goals **G3** and **G4**.
- **Chapter 6** presents the *Conclusions and future work* of the research work providing a summary of the whole study, draws the main conclusions out of it and introduces to future work. It fulfills goal **G5**.

Chapter 2

Background

This chapter introduces the fundamental knowledge related to the work under investigation, useful to have a comprehensive understanding of the problem at hand, of its related works and of the solution proposed in the next chapters.

2.1 Satellite Data

Satellite data mainly concern information about Earth and other planets gathered by satellites floating in the universe. One of the most common applications for satellite data is EO: satellites floating in the Earth's orbit constantly collect and deliver authentic information about the surface, weather changes and other relevant phenomena happening on the planet.
The gathered information is mainly generated via Remote Sensing (RS) technologies capable of observing the Earth with devices physically remote from it and transmitting images or other data back to the ground stations. Observation satellites such as ESA Sentinel-1 and Sentinel-2 are launched to relatively low altitude orbits (respectively 693 and 786 km from the surface of the Earth) to be able to address their imaging tasks.
Data may be collected through active or passive RS technology. Among the passive systems, those able to gather data only in sunlit and cloudless circumstances through optical and thermal sensors, there is the Sentinel-2 satellite; among the active systems that send energy to Earth and measure the energy received back through radar and laser technologies, there are two types of systems used for microwave imaging: Real Aperture Radar (RAR) and SAR, to which Sentinel-1 belongs. An active system that provides its own illumination and its microwaves are able to pass through clouds and other weather with little attenuation.

2.1.1 Synthetic Aperture Radar

The technique of microwave fine-resolution two-dimensional radar imaging called SAR, among the other RS technologies, is the only instrument able to penetrate clouds and that does not require sunlight to record reliable data. It provides timely data stream opportunities for several applications at land and sea and is especially relevant for monitoring those areas of the Earth which most of the time are covered by clouds. For organizations, businesses and governments, this means possibilities for acting on time in case of specific phenomena and tackling the detected issues, no matter the weather conditions. Main civilian applications of radar imagery include land use analysis, oceanography, agriculture and more [20], while main military applications include surveillance, navigation and more.
SAR generated images are monochrome since this technique measures only the scalar quantity of reflectivity, while its resolution may range from several tens of meters down to a few inches. In particular, this resolution should be available in both the range and cross-range dimensions, quantities that are introduced below.
SAR is a radar system, the acronym for RAdio Detection And Ranging. Radars work at radio frequency and are composed of a transmitter, a switch, an antenna, a receiver and a data recorder, as presented by [21] and visible in Figure 2.1.

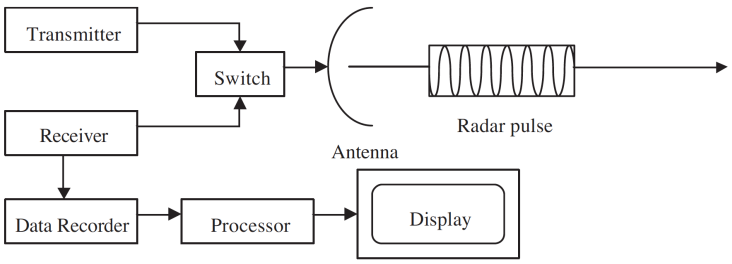

Figure 2.1: Radar building blocks, image available at [21]

The transmitter is the block responsible to generate the electromagnetic waves at radio wavelengths and pass them to the switch, which directs the pulse to the antenna, waits for the echo signal and then returns it to the receiver. The antenna transmits the electromagnetic waves towards the area to be imaged and collects the related echoes. Finally, the receiver converts the returned signal to a digital number and passes it to the data recorder which stores it for later processing. The radar flies with constant speed along the track direction, while its antenna produces a beam illuminating

the ground.
As previously stated, there are two kinds of dimensions that that is worth considering: range and cross-range. As introduced in [22] and shown in Figure 2.2, the antenna scans in azimuth angle θ, thus at a range R the beamwidth is $R\theta$ meters to a good approximation. The cross-range dimension, instead, is the direction orthogonal to range. This is relevant since the two scatterers in the picture are considered resolvable only if they are separated by the width of the antenna beam, as the second case represented.

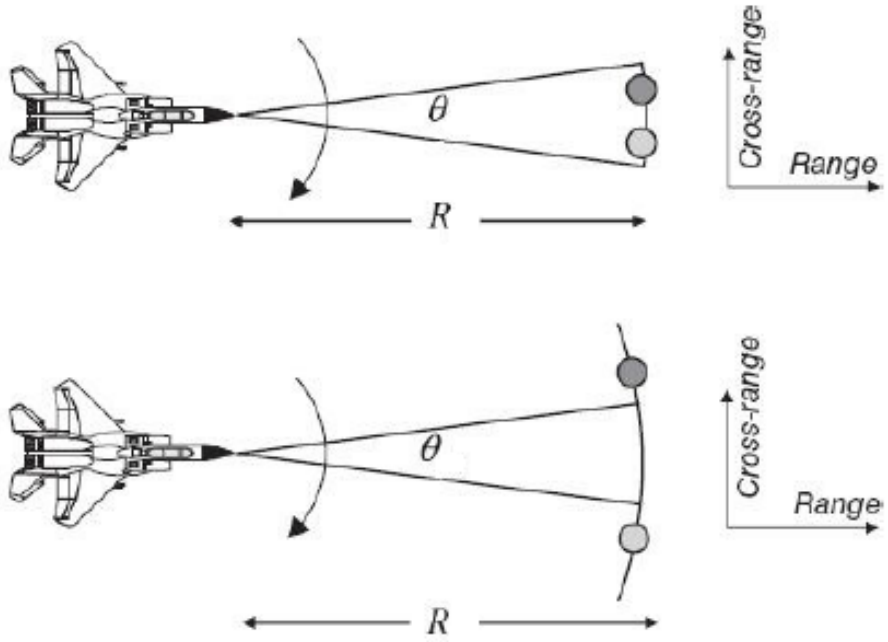

Figure 2.2: Radar range and cross-range dimensions, image available at [22]

The two resolutions related to the cross-range and range dimensions are respectively:

- Azimuth or cross-range resolution, that is mainly determined by the beamwidth θ and is given by:

$$\begin{cases} \rho_a = R\theta \\ \theta = k\frac{\lambda}{l} \end{cases}, \text{ so:}$$

$$\rho_a = k\frac{R\lambda}{l}$$

where λ is the wavelength of the transmitted signal, l is the radar aperture length or antenna width, R is the vertical to the terrain and k is the scale factor that depends on the antenna design and whose value is often around 1.
Azimuth resolution indicates the ability of a radar to separate two targets in the direction parallel to its motion, and in particular represents the minimum distance on the ground at which the two scatterers can be imaged separately.
Since the azimuth resolution is strongly dependent on the aperture length l and degrades in proportion to the range R, a larger antenna operating at a restricted range needs to be employed to obtain satisfying results. At last, given its dependence on wavelength λ, the resolution would be further improved with higher frequencies.

- Range resolution, that is mainly determined by the pulse length and is given by:

$$\rho_r = \frac{c\tau_p}{2sin\theta}$$

where c is the speed of light, τ_p is the pulse length and θ is the incidence angle between the line which connects the antenna to the object and the vertical to the terrain.

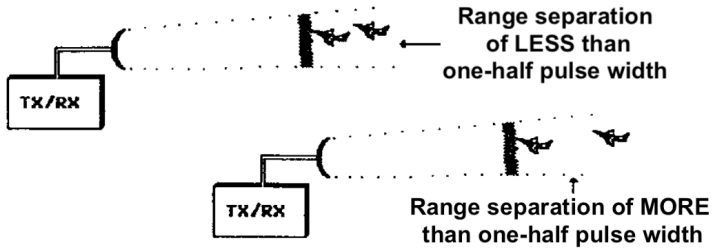

Figure 2.3: Range resolution and pulse length, image available at [23]

As Figure 2.3 shows, if the distance between two targets is less than half the pulse width, they will be not detected as distinct by the radar. Thus, the pulse length should be as short as possible to improve range resolution. Simultaneously, the radar pulses need to transmit enough energy to enable the detection of the reflected signals. Therefore, if the pulse is shortened, its amplitude must be increased to keep the same amount of energy in the pulse.

As highlighted by the two previous notions of resolution, the larger the antenna the radar is equipped with, the finer the detail it can resolve, at least for what concerns RARs. But such changes are impractical, as visible from Figure 2.4 in which is shown the relative size of a fighter aircraft and an array of antennas large enough to achieve a ρ_a of $3m$ at X band and $10km$ range.

Figure 2.4: Unfeasible array of antennas, image available at [22]

SARs, instead, solve the mechanical problems involved in building the equipment needed to transmit a very short and high-energy pulse leveraging on a signal processing approach, as introduced below.

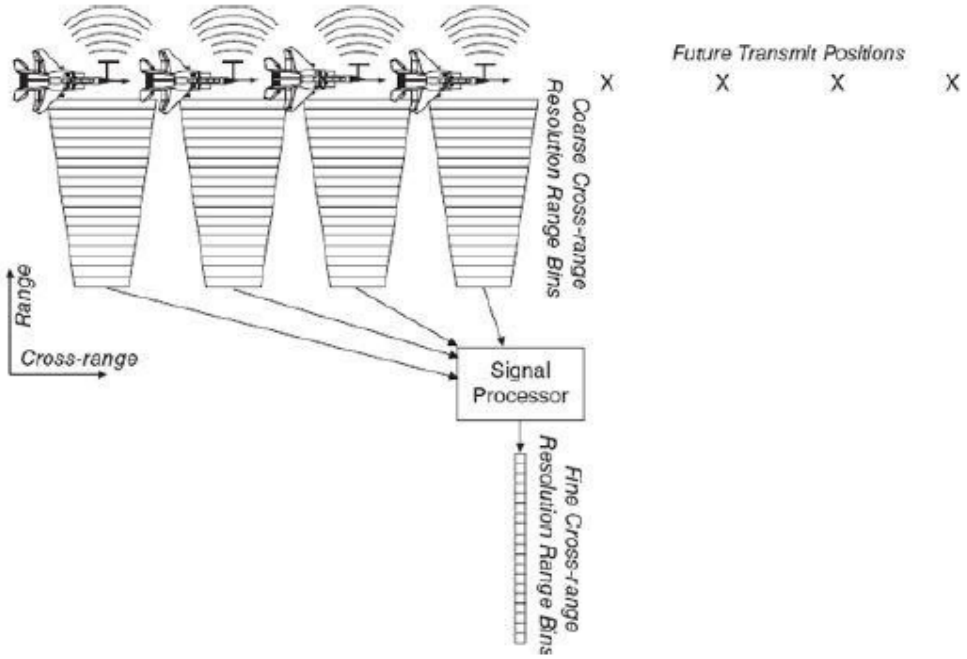

Figure 2.5: SAR signal processing, image available at [22]

To obtain acceptable azimuth resolution at spacecraft altitudes SARs aim at synthesising a long antenna by using the forward linear motion of the platform. As suggested from the picture above, the data are collected serially, one element at a time, rather than in parallel all at once. After process-

ing, the data have fine resolution in both range and cross-range. It does so by implementing methods such as the Two-dimension algorithm [24], which processes the range and azimuth data simultaneously, the Range Doppler Processing, which does range compression processing before the azimuth compression processing, or the Chirp Scaling approach, which provides pulse compression by frequency modulation.
An extensive analysis of the above-mentioned algorithms has been done by Cumming and Wong [25], while other useful details on SAR signal processing are provided by Curlander and McDonough [26] and Lanari and Franceschetti [27].
Undertaking these analyses, the approximated azimuth resolution that it is possible to obtain with SAR is:

$$\rho_a = \frac{l}{2}$$

where l is the length of the antenna in the along-track direction: in SARs a smaller physical antenna generates broader beamwidth θ, allowing a larger maximum synthetic aperture size.
This is a remarkable result and shows the independence of the SAR azimuth resolution to the wavelength and the slant range, thus the altitude. Since range resolution is independent to the altitude as well, a SAR can operate at any height with no variations in resolution. As a consequence, this technology is used with aircraft based imaging radars and spaceborne operations.

2.1.2 Sentinel-1

Sentinel-1A and Sentinel-1B are two satellites launched by ESA respectively on 03 April 2014 and 25 April 2016, with a lifespan of 7 years and with the aim of Land and Ocean monitoring, including sea ice observations and iceberg monitoring. The objective of the mission is to provide medium and high-resolution SAR imaging in all weather conditions operating in a range of radar frequencies around 5.404 GHz called C-Band.
The SAR operating configuration is defined by the swath width and spatial resolution, which is usually expressed in meters and represents the smallest size a single pixel in the sensor covers on the ground.
As visible in Figure 2.6 Sentinel-1 can operate at 4 beam modes:

- Strip Map Mode: 5 m (range) x 5 m (azimuth) spatial resolution, 80 km Swath
- Interferometric Wide Swath: 5 m x 20 m spatial resolution, 250 km Swath

- Extra-Wide Swath Mode: 25 m x 100 m spatial resolution, 400 km Swath
- Wave (WV) Mode : 5 m x 20 m spatial resolution, 20 km x 20 km vignettes collected every 100 km along the orbit

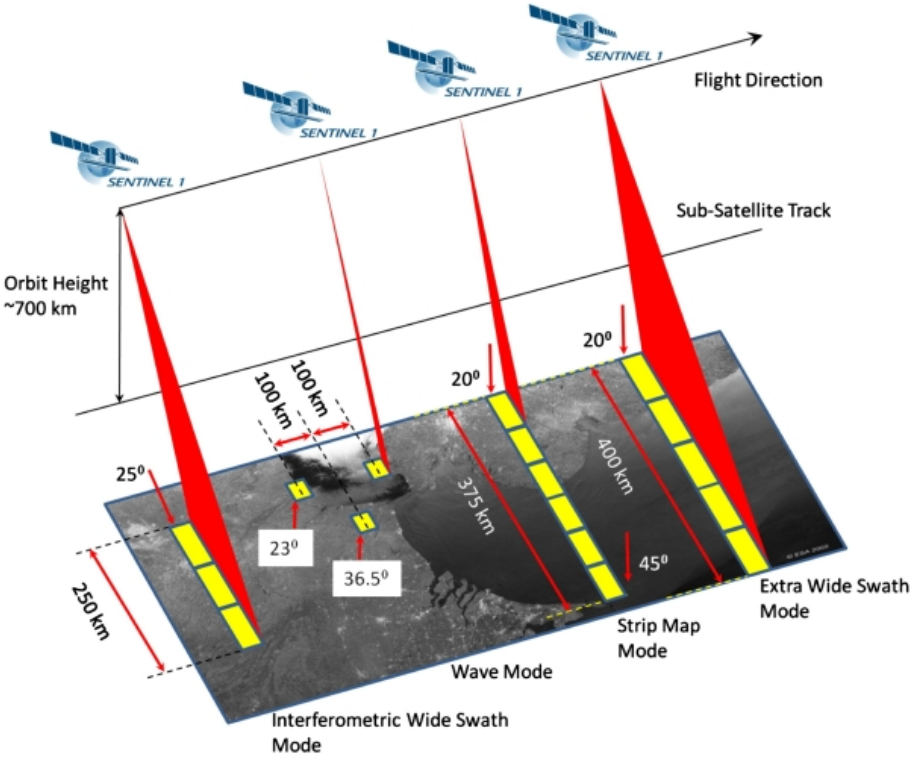

Figure 2.6: Sentinel-1 acquisition modes, image available at [28]

Sentinel-1 can use the full Synthetic Aperture and the complete signal data history to produce the highest possible spatial resolution Single-Look Complex (SLC) SAR image product, or may generate multiple looks by averaging over range or azimuth bandwidths. The SLC processing algorithm takes into account both the data in azimuth and range resolutions and combines them to form an image following a procedure explained in [29].
For what concerns the polarisation, the Sentinel-1 instruments support operations in single polarisation (HH or VV) or dual polarisation (HH + HV or VV + VH). As shown in Figure 2.7, HH means both transmitting and receiving horizontally while HV means transmitting horizontally and receiving vertically. SAR acquisitions performed at different polarisations result in slightly different representations of the same target, due to the variation of the backscatter signal intensity that returns to the sensor depending on

the chosen polarisation. Some polarisations might be more sensitive to flat surfaces while some others may be more suitable to describe land variations.

Figure 2.7: Polarisation modes, image available at [30]

WV acquisitions, in particular, consist of several vignettes acquired in either VV or HH polarisation every 100 km along the orbit and alternately on two different incidence angles (approximately 23° and 36° respectively).
Strip Map acquisitions, instead, consist of several vignettes mostly acquired in HH + HV polarisation with incidence angle varying from 18.3° to 46.8°.

2.2 Classification

The classification problem is a predictive task whose goal is to best approximate a mapping function from the input variables to the output variables, that are discrete. The final aim is to correctly identify which category the new data will fall into.
In particular, it is a technique in which a model, called classifier, is trained to learn from input samples and then used to generalize on new unseen instances [31] based on patterns identified in the training data. Classification tasks can be split into multiclass and binary problems.
In binary classification problems, input data belong to a positive or a negative class. The output of a binary classification algorithm is a binary prediction vector, where each new given sample is labeled as positive or negative.
In multiclass classification problems, instead, input data belong to one of the available classes. The output of a classification algorithm, in this case, is often a 1-hot encoded prediction vector, where each new given sample is labeled with an encoding corresponding to the class it belongs to.

2.2.1 Classification methods

Logistic Regression

The Logistic Regression algorithm, also known as Logit Regression, is an extension of the Linear Regression for classification purposes. In particular,

it estimates the probability that an instance belongs to a specific class (e.g. What is the probability that the considered image represents an iceberg?). Instead of fitting a straight line or hyperplane as the Linear Regression does, the Logistic Regression model makes us of the logistic function to flatten the output of a linear equation between 0 and 1, interpreted as a probability. Based on this probability and a threshold the model predicts whether the instance belongs to the positive class (e.g. iceberg) or it does not. This makes it a binary classifier, thus the name Binary Logistic Regression.
The logistic function, also called sigmoid function, is defined as follows:

$$\sigma(z) = \frac{1}{1+e^{-z}}$$

And it looks like this:

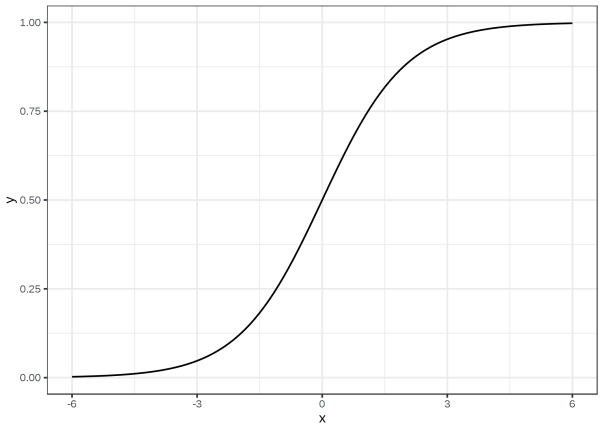

Figure 2.8: Sigmoid function

Differently from the Linear Regression model, which directly outputs the weighted sum of the input features (plus a bias term), the Logistic Regression computes this same quantity but outputs the logistic of this result, as visible here:

$$z = w_0 x_0 + w_1 x_1 + w_2 x_2 + ... + w_n x_n = w^T X$$

$$\sigma(z) = \sigma(w^T X) = \frac{1}{1+e^{-w^T X}}$$

Once the Logistic Regression estimates the probability $\sigma(z)$ that a sample belongs to the positive class, it can output its prediction $\hat{y}$:
$\sigma(z) < 0.5$ when $z < 0$, and $\sigma(t) \geq 0.5$ when $z \geq 0$, so the Logistic Regression predicts $\hat{y} = 1$ if $z = w^T X$ is positive and $\hat{y} = 0$ if it is negative.
In the case of Linear Regression, it is possible to find the optimal weights by solving the normal equations. Logistic regression is somewhat more difficult

since there is no closed-form solution for its optimal weights. The objective of the training is to set the parameter vector so that the model estimates high probabilities for positive instances ($y = 1$) and low probabilities for negative instances ($y = 0$). This idea is captured by the following cost function:

$$cost(\hat{y}, y) = \begin{cases} -log(\hat{y}) & \text{if } y = 1 \\ -log(1 - \hat{y}) & \text{if } y = 0 \end{cases} \tag{2.1}$$

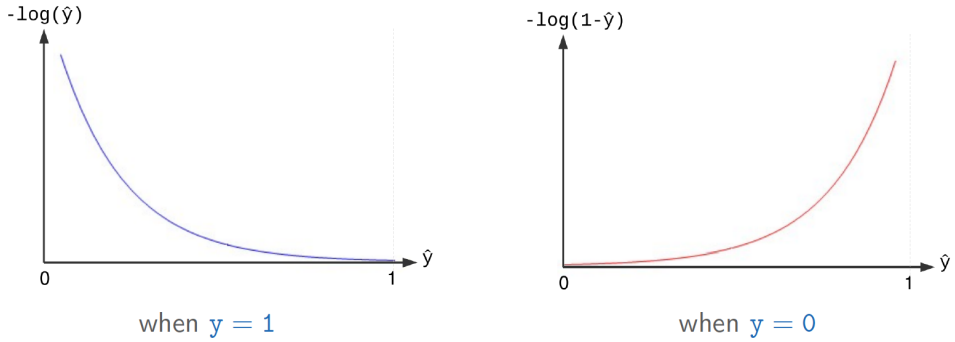

Figure 2.9: Binary Logistic Regression cost function

The cost is desired to be: close to 0, if the predicted value $\hat{y}$ will be close to true value y; large, if the predicted value $\hat{y}$ will be far from the true value y. Therefore, if $y = 1$, $-\log(\hat{y})$ will give a result close to 0 if $\hat{y}$ is close to 1. Conversely, the cost to pay grows to infinity as $\hat{y}$ approaches to 0.
Instead, if $y = 0$, $-\log(1 - \hat{y})$ will give a result close to 0 if $\hat{y}$ is close to 0. Conversely, the cost to pay grows to infinity as $\hat{y}$ approaches to 1.
This function also grants the convexity to the function the Gradient Descent algorithm has to process, which thus is guaranteed to find the global minimum.
The cost function over the whole training set is the average cost over all the training instances. It can be written in a single expression known as the Log Loss, shown below:

$$J(w) = -\frac{1}{m} \sum_{i=1}^{m} y^{(i)} \log(\hat{y}^{(i)}) + (1 - y^{(i)}) \log(1 - \hat{y}^{(i)})$$

Following a probabilistic interpretation it is possible to come up with the Negative Log-Likelihood equation, which is the same as the Logistic Regression cost function written above. Therefore, minimizing the negative Log-Likelihood, also known as Cross-Entropy given its ability to measure the difference between two probability distributions, means minimizing the Logistic Regression cost function $J(w)$.

According to the Gradient Descent algorithm (and its Batch, Mini-Batch and Stochastic variants), the minimization takes place starting at a random initialization of the weights and then repeating the following steps, until a stopping criterion is satisfied:

- Determine a descent direction $\frac{\partial J(W)}{\partial w}$;
- Choose a step size η;
- Update all the parameters simultaneously: $w^{(next)} = w - \eta \frac{\partial J(W)}{\partial w}$.

For what concerns the Multinomial Logistic Regression, things slightly differ from the previous case. Instead of two classes, it deals with k classes and therefore one set of parameters w is no longer sufficient and it needs k sets of parameters W, since it has to estimate the result of each individual label. Instead of the sigmoid function, in the multiclass case the softmax function is used:

$$\hat{y_j} = p(y = j|x; w_j) = \sigma(w_j^T X) = \frac{e^{w_j^T x}}{\sum_{i=1}^{k} e^{w_i^T x}}$$

The softmax function is able to calculate the probabilities of each target class over all possible target classes, and if computed for two classes it is equivalent to the sigmoid function. The prediction outputs the class with the highest estimated probability.
As in the previous case, the cost function is represented by the Cross-Entropy.

Support Vector Machine

SVM is a classification algorithm that has been introduced in 1995 by Cortes et al. [32]. It trains a classifier by first creating a feature-space, mapping original instances into it and then defining linear partitions (linear separating hyperplanes) of the feature space into categories. SVM finds the partitions that separate the data into categories by maximizing the orthogonal distance between the nearest points of each category and the separating hyperplanes. This distance, called margin, is shown in Figure 2.10. The closest point to the boundary for each category is called support vector, and it is used to calculate the separating hyperplane.
There exist also techniques through which SVM can perform multiclass classification, such as the One-against-one in which SVM classifiers for all possible pairs of classes are created [34]: when applied to test data, each classifier gives one vote to the winning class and in the end the sample is labeled with the class having most votes.

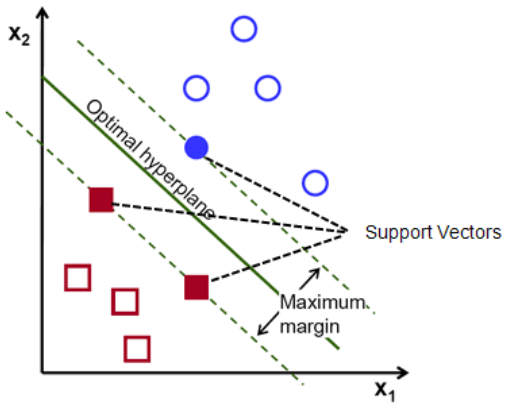

Figure 2.10: SVM support vectors intuition, image available at [33]

The success of SVM is linked to the ability to work well in non-linear cases as well, by creating non-linear separation between classes. It can do so by leveraging on different types of kernel transformations [35], such as the Radial Basis Function kernel (RBF). The RBF kernel on two instances x and $x^{'}$, represented as vectors in the feature space, is defined as:

$$K(x, x^{'}) = e^{\frac{||x - x^{'}||^2}{2\gamma}}$$

This function is of the form of a bell-shaped curve, whose height ranges between 0 and 1 and therefore can be interpreted as a similarity measure among the points; its width, instead, is determined by the parameter γ: the smaller its value the wider is the bell, as visible in Figure 2.11, meaning that the decision boundary is less effected by individual data points like in Figure 2.12.

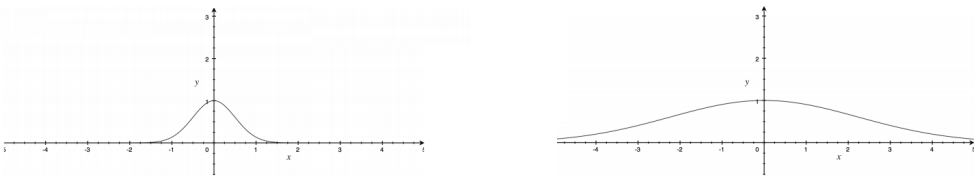

Figure 2.11: RBF γ parameter choice affects the bell shape, image available at [36]

2.2.2 Classification evaluation

Evaluating the model is a crucial part of any classification problem. The proposed solution can give satisfying results when evaluated using a certain

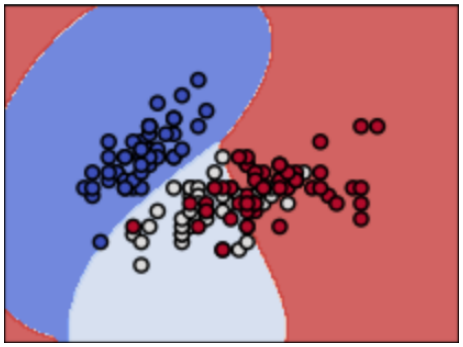

Figure 2.12: RBF non-linear decision boundaries shapes are affected by the value of γ parameter, image available at [37]

metric, but may give poor results when evaluated using other metrics.
Several methods to evaluate the performance of a classifier do exist. It is useful to introduce the concept of Confusion Matrix to understand them. The Confusion Matrix is used to describe the performance of a classifier on a set of test data for which actual values are known.
In a binary classification task, there are two possible predicted classes: positive and negative, and this also applies to multiclass classification if each time one class is considered as positive and the rest of the classes as negative. Then there are the class actual values (positive or negative) defined by their ground truth labels. According to this logic, it is possible to define four classification possibilities:

- **True Positives** (TP): set of samples belonging to the positive class, that the model correctly predicted as positive.
- **False Positives** (FP): set of samples belonging to the negative class, that the model incorrectly predicted as positive.
- **False Negatives** (FN): set of samples belonging to the positive class, that the model incorrectly predicted as negative.
- **True Negatives** (TN): set of samples belonging to the negative class, that the model correctly predicted as negative.

The Confusion Matrix, as shown in Figure 2.13, summarizes these concepts collecting information about the support of each group. It is used as a starting point to calculate the most common evaluation metrics, presented below. Based on the Confusion Matrix, three main evaluation metrics can be defined:

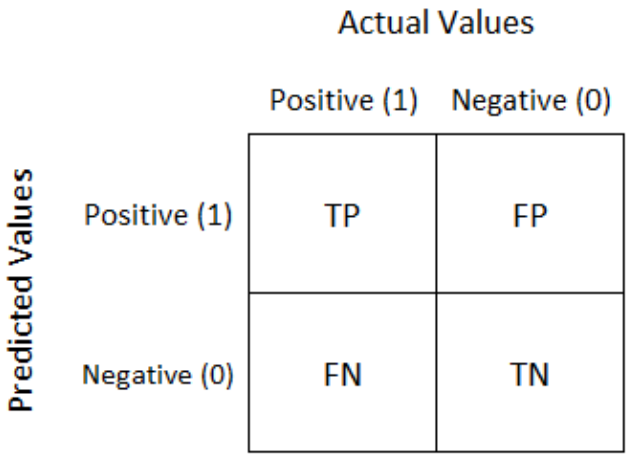

Figure 2.13: Confusion Matrix, image available at [38]

- The **Overall Accuracy** is calculated as the number of all correct predictions divided by the total number of instances in the test set. Accuracy is generally a good indicator, but it could become misleading when dealing with an unbalanced dataset.

$$OA = \frac{TP+TN}{TP+TN+FP+FN}$$

- The **Precision** is the number of correct positive results divided by the number of both correct and incorrect positive results predicted by the classifier. Precision is a good measure to consider when the cost of false positives is high.

$$P = \frac{TP}{TP+FP}$$

- The **Recall** is the number of correct positive results divided by the number of all relevant samples (e.g. all samples that should have been correctly predicted as positive). Recall is a good metric to consider when there is a high cost associated with false negatives, such as the case of iceberg detection.

$$R = \frac{TP}{TP+FN}$$

Many other metrics are used to evaluate machine learning models, such as Area Under the ROC Curve (AUC-ROC) or $\mathbf{F_1}$ **score**, that takes into account both Precision and Recall according to the formula:

$$F_1 = \frac{2*P*R}{P+R}$$

OA, Precision, Recall and F_1 score are the metrics used to evaluate the performance of the proposed methods later in this work.

2.3 Deep Learning

In this section are presented the main theoretical and background notions underlying the DL field, a subset of ML, and its main applications that involve ANNs architectures such as CNNs or GANs.

2.3.1 Machine Learning definition

The main challenge that AI tackles is solving tasks that are hard for people to describe formally. In this scenario, ML is the part of AI that lets computers solve this kind of task by learning from experience: an ML algorithm is an algorithm that is able to learn from data.
What does learning mean? According to Mitchell [39]: "A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E".
The experience E is composed by examples, collections of features measured from events or objects that the ML algorithm can process in the first place. Task T is the activity that the ML system can fulfill once the learning process performed on the examples is complete. Here lies the main difference between learning and task: learning is the means of achieving the ability to perform the task. At last, the performance P is the quantitative measure used to evaluate the ability of a ML algorithm to perform the concerned task, and it is usually specific to the task.
For instance, in a classification task the ML algorithm is asked to specify which categories some test examples given in input belong to, and it performs this task building a mathematical model learnt from training examples and evaluating its performance through specific quality metrics like the OA. In general, the resulting model is reliable when the training examples follow a distribution similar to that of future test examples, so that the ML algorithm becomes able to make good predictions on the test data without explicitly being programmed on how to perform this task.
ML algorithms can be categorized into different types based on the type of experience, namely input and output data, they handle during the learning process and type of task they are intended to solve. Here follows a textual summary of the main ML categories:

- Supervised Learning (SL): the ML algorithm experience is composed of a dataset containing examples associated with a label, or target. Its name is due to the presence of a notional supervisor during the learning process that assigns the labels to the data. Instead, in the case of SSL, that is a slightly different variant of SL, part of the labels is present while another is missing.

- Unsupervised Learning (UL): the ML algorithm experience is composed a dataset containing examples made of features, through which the system learns useful properties of the dataset intrinsic structure without the help of any target.

- Reinforcement Learning (RL): in this case there is not an actual prior experience on which the ML algorithm bases its learning process. Here the learning process is based on a trial-and-error method and is composed of an agent that interacts with its environment, performs actions and gets rewards.

A visual summary, with some additional details, is attached in Table 2.1.

Categories	**SL**	**UL**	**RL**
Definition	Machine learns from labeled data	Machine learns from unlabeled data	Agent learns from rewards related to its actions in the environment
Problems	Regression and classification	Association and clustering	Reward-based
Data	Labeled data	Unlabeled data	No predefined data
Training	External supervision	No supervision	No supervision
Approaches	Maps labeled inputs to known outputs	Understands patterns and discovers the output	Follows the trial-and-error method

Table 2.1: Overall comparison of RL, UL and RL

The focus of this thesis is mainly dedicated to the SL and SSL approaches. As visible from the table above, these categories usually address regression tasks, in which the output variable is continuous, or classification tasks, in which the output variable categorical or discrete as already discussed in the previous section.
This project, in particular, tackles the problem of classifying satellite images representing classes such as icebergs or ice caps.

2.3.2 Neural Networks

DL is a subfield of ML algorithms based on learning data representations. Its main objective is to mimic the neural networks of the human brain, that is why ANNs are inspired by human biological neurons. Humans indeed learn thanks to the millions of interconnected neurons in their brain that transmit electrical impulses and activate each other via axons [1]. In particular, these impulses are called action potentials and make the synapses release chemical signals known as neurotransmitters. When a neuron receives an adequate amount of neurotransmitters within a few milliseconds, it fires its own action potentials towards other neurons, which especially in the cerebral cortex are organized in consecutive layers.
Similarly to our brain, SL leverages on Neural Networks (NNs) made of consecutive layers to learn how to transform an input X to an output Y. As the name suggests, NNs are composed of neurons, that represent the basic logic units of any NN. McCulloch and Pitts in 1943 [40] introduced a simple model of the artificial neuron with one or more binary inputs and one binary output. A few years later, in 1957, Frank Rosenblatt invented the Perceptron [41], a slightly different neuron architecture composed by a single layer of Linear Threshold Units (LTUs), whose inputs and output are numbers and in which each input connection is coupled with a weight. The Perceptron training algorithm aims at reinforcing the connections that help reduce the error and it is inspired by Hebb's rule [42], according to which the connection between two neurons tends to grow when they fire simultaneously.
In 1969 Minsky and Papert [43] highlighted some weaknesses of Perceptrons, such as their incapability of solving the XOR classification problem, and it turned out that such limitations could be eliminated by stacking multiple Perceptrons. The result is a Multilayer Perceptron (MLP), composed of one input layer, one or more hidden layers of LTUs, and one final layer of LTUs called the output layer, which represents the foundation of modern ANNs.
The modern neuron takes a vector of inputs $X = [x_1, .., x_n]$ and computes

an output y. The computation happens as follows:

1. It multiplies each input vector of inputs $X = [x_1, .., x_n]$ by a vector of weights $W = [w_1, .., w_n]$

2. It computes the summation of the weighted inputs and adds a bias term $b : (x_1 \Delta w_1) + .. + (x_n \Delta w_n) + b$

3. It applies an activation function f to the summation, such that: $y = f(x_1 \Delta w_1 + .. + x_n \Delta w_n + b)$

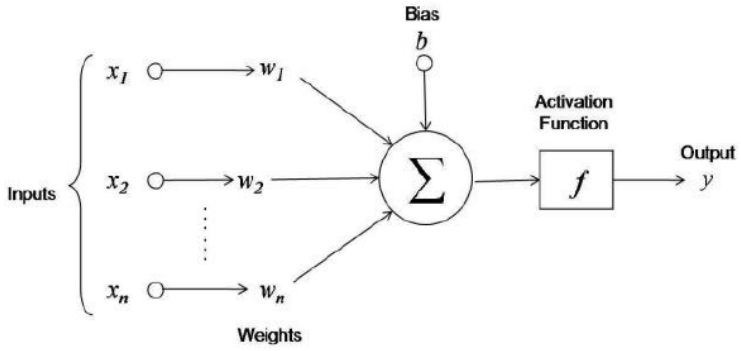

Figure 2.14: Neuron architecture, image available at [44]

Figure 2.14 shows the basic neuron architecture. Common activation functions f are the *sigmoid*, the *hyperbolic tangent (tanh)* or the *Rectified Linear unit (ReLu)* [45] functions. The use of non-linear activations allows NNs to generate non-linear mappings between inputs and outputs, meaning that they are able to compute and learn any theoretical function and thus are considered as universal function approximators [46]. Figure 2.15 shows the structure of a single-layer NN, that, as well as the MLP, is organized in layers of neurons. If a NN is composed of more hidden layers it is known as a multi-layer NN or Deep Neural Network (DNN).
A NN learns through the minimization of the output error, often defined by a cost function. For instance in SL tasks, when training data are labeled, a common choice for the cost function J is the Mean Square Error (MSE):

$$J(w) = MSE = \frac{1}{m} \sum_{i=1}^{m} (y^{(i)} - \hat{y}^{(i)})^2$$

where, for each test sample i, y represents the actual value while $\hat{y}$ represents the predicted value. As already introduced in section **2.2.1**, for classification tasks is usually preferred the Cross-Entropy cost function.

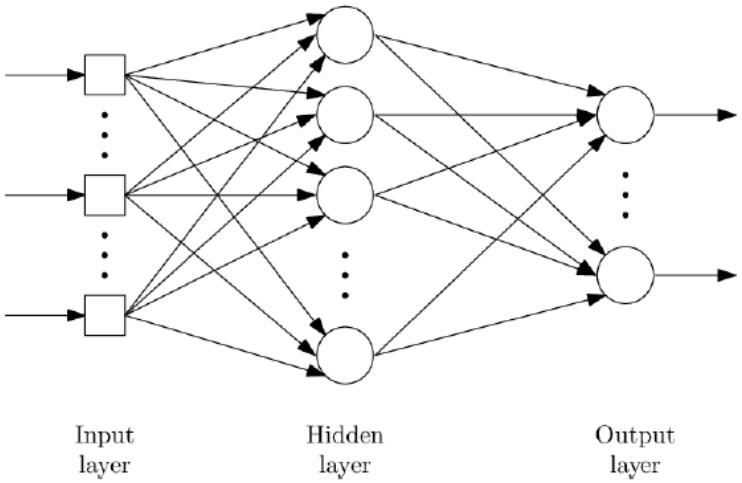

Figure 2.15: Neural Network structure

To minimize the cost function and learn the right output to produce, the Gradient Descent method is used to update the NN's weights. It is an optimization algorithm that consists of iteratively moving in the direction of steepest descent as defined by the negative of the gradient, which is computed as:

$$\frac{\partial J(W)}{\partial w}$$

To implement this algorithm a technique known as Backpropagation is used. When inputs are given to the NN the outputs are computed according to the current weights $W = [w_1, .., w_n]$, and the cost function is calculated as well as its gradient. Therefore Backpropagation consists of propagating the gradient of the cost function backward through the network to update each weight involved in the computation of the error. Once the backward pass is complete and thus the gradient is successfully computed, the weights are updated as follows, according to a learning rate η:

$$w^{(next)} = w - \eta \frac{\partial J(W)}{\partial w}$$

This means calculating the gradient with respect to the weights using the chain rule, an example of which is shown in Figure 2.16 and the formulas below:
The Backpropagation steps to compute $w_1^{(new)}$ are:

$w_1^{(new)} = w_1 - \eta \frac{\partial E_{total}}{\partial w_1}$, where:

$\frac{\partial E_{total}}{\partial w_1} = \frac{\partial E_{total}}{\partial out_{h1}} \frac{\partial out_{h1}}{\partial net_{h1}} \frac{\partial net_{h1}}{\partial w_1}$, where:

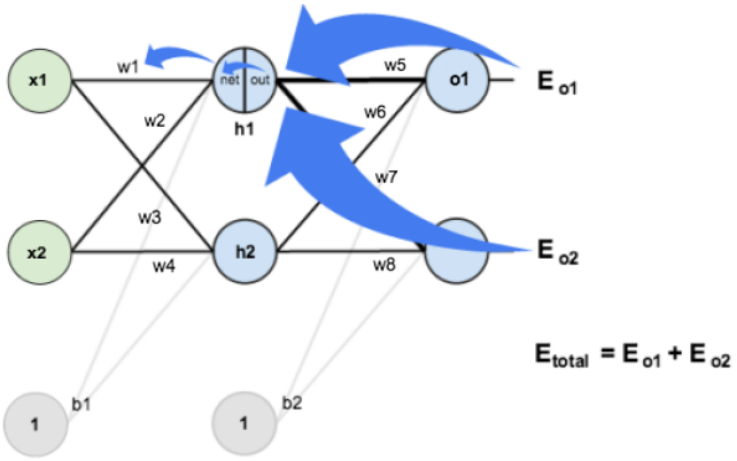

Figure 2.16: Backpropagation, image available at [47]

- $\frac{\partial E_{total}}{\partial out_{h1}} = \frac{\partial E_{o1}}{\partial out_{h1}} + \frac{\partial E_{o2}}{\partial out_{h1}} = \frac{\partial E_{o1}}{\partial out_{o1}} \frac{\partial out_{o1}}{\partial net_{o1}} \frac{\partial net_{o1}}{\partial out_{h1}} + \frac{\partial E_{o2}}{\partial out_{o2}} \frac{\partial out_{o2}}{\partial net_{o2}} \frac{\partial net_{o2}}{\partial out_{h1}}$, where:
  - $\frac{\partial E_{o1}}{\partial out_{o1}} = -2\frac{1}{2}(target_{o1} - out_{o1})$
  - $\frac{\partial out_{o1}}{\partial net_{o1}} = 1$ if $activation function = ReLu$ and $net_{o1} > 0$
  - $\frac{\partial net_{o1}}{\partial out_{h1}} = w_5$

  and:

  - $\frac{\partial E_{o2}}{\partial out_{o2}} = -2\frac{1}{2}(target_{o2} - out_{o2})$
  - $\frac{\partial out_{o2}}{\partial net_{o2}} = 1$ if $activation function = ReLu$ and $net_{o2} > 0$
  - $\frac{\partial net_{o2}}{\partial out_{h1}} = w_7$
- $\frac{\partial out_{h1}}{\partial net_{h1}} = 1$ if $activation function = ReLu$ and $net_{h1} > 0$
- $\frac{\partial net_{h1}}{\partial w_1} = x_1$

In practice, three strategies are mainly used to implement the Gradient Descent algorithm:

1. Batch Gradient Descent, that calculates the error for each sample in the training set and updates the model only after all training samples have been evaluated. Model updates are performed at the end of each training epoch, a cycle through the entire training set. As a consequence, it may be slow on very large training sets;

2. Stochastic Gradient Descent, that picks a random instance in the training set at every step to compute the gradient, thus it is faster than Batch Gradient Descent but less regular;

3. Mini-Batch Gradient Descent, a variation of the Batch Gradient Descent that splits the training set into small random batches used to calculate that error and update model coefficients accordingly. It represents a sort of trade-off solution and it is the most common practice in the DL field.

To face the challenges (e.g. slow training process) that choosing one of these three strategies may entail, some optimization algorithms are used in the DL community. Among them it is worth mentioning: the Momentum optimizer [48], that helps to escape from bad local minima including previous gradients information into a specific vector added to the local gradient; the RMSProp optimizer [49], which adapts the learning rate dividing it by an exponentially decaying average of the past squared gradients; the Adam optimizer [50], that not only stores an exponentially decaying average of the past squared gradients like RMSProp, but also keeps an exponentially decaying average of the past gradients similar to Momentum.
Another big challenge that one can face while training NNs is overfitting the model on the training data, especially when the model has a large number of parameters. To address this issue, some approaches that can be followed are: Early Stopping, which takes care of monitoring the error computed on the validation set and stops the training process as soon the validation error reaches a minimum; Dropout, which consists of randomly dropping out each neuron temporarily during training, that is like averaging the effects of different NNs. These NNs may overfit in different ways, but the net effect of Dropout is to reduce overfitting.
To proceed with the next sections, in the DL field the most common implementations of NNs are: Feedforward Neural Networks [51], Recurrent Neural Networks[52], Long-Short Term Memory Neural Networks [53], CNNs [54], GANs [55] and Autoencoders [56].
Since the methods proposed in this thesis rely on CNNs and GANs, extensive analysis and presentation of these architectures are proposed in the following sections.

2.3.3 Convolutional Neural Networks

CNNs [54] are specialized NNs for processing data with grid-like topologies, such as images, that are 2D-grids of pixels. CNNs get their name from an operation called *convolution*, that is used instead of the traditional matrix multiplication in at least one of the NN layers. This operation takes inspiration from the behaviour of the human visual cortex neurons: they have a local receptive field and thus react only to stimuli located in a limited region

of the visual field.
As a result, in a CNN architecture neurons in the first layer are not connected to every pixel in the input image but only to those in their receptive fields. In turn, each neuron in the following layer is connected only to neurons located within a small region in the previous layer and so on, as shown in Figure 2.17. This architecture allows the first hidden layers to concentrate

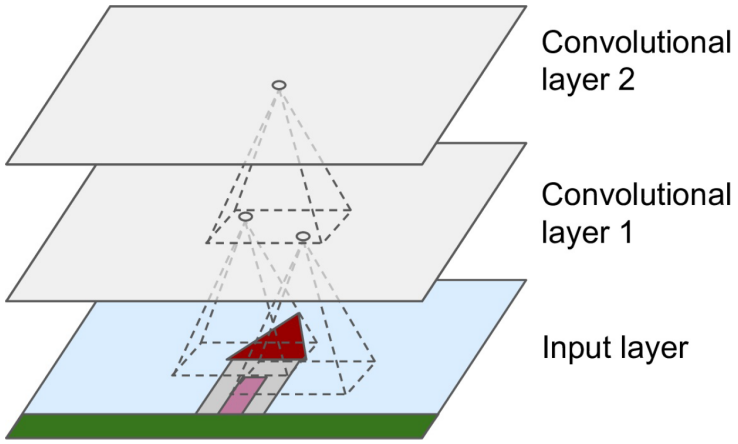

Figure 2.17: CNN intuition, image available at [1]

on small low-level features and assemble them into large high-level features in the next hidden layers, summing up the overall aim of CNNs: extracting higher-level features from high dimensional data.
More in detail, each layer of a CNN applies one or more filters on the input data and generates as output a simplified representation of its input. Each neuron is associated with a particular region (known as receptive field) of the input and applies to that region a transformation described by the filter (also known as kernel). All the neurons in a certain layer usually apply the same filter (but on different regions), give the result as input to the activation function and generate the output, as shown in Figure 2.18. Moreover, it is also possible to apply more filters in a given layer to extract features in different ways. For instance, in the case of RGB images classification, there are always three channels (Red, Green and Blue) and each channel applies at least one filter.
In contrast to traditional NNs, mainly based on fully connected layers and weights used exactly once when computing the output of a layer, CNNs leverage on techniques introduced above such as sparse interaction and parameter sharing to reduce the ML system memory requirements and improve its statistical efficiency. In the end, the number of parameters that define

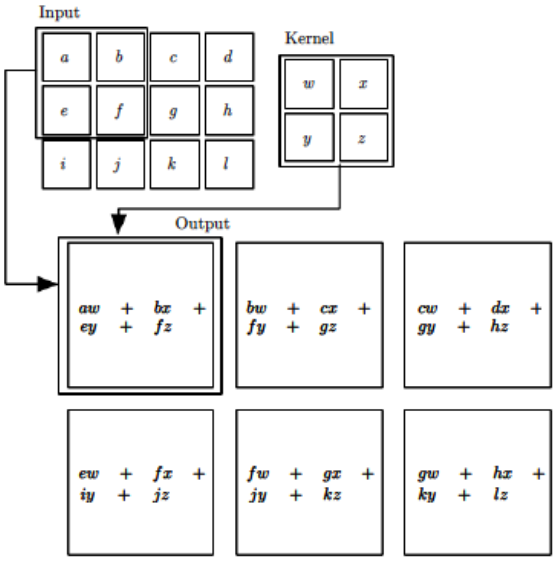

Figure 2.18: CNN Kernel, image available at [10]

a CNN only depends on the size of the filters, the number of filters (also known as feature maps) and the number of channels.
In addition to the *Convolutional* layers, a CNN is also composed of *Pooling* layers: they aim at subsampling the input data to further reduce the computational load, the number of parameters and thus the memory usage. What a Pooling layer does is replacing the output of a certain layer with a summary statistic of the nearby outputs (e.g. Max-pooling [57] function reports the maximum output within a rectangular neighborhood, while the Average-pooling reports its average). This procedure helps to make the representation invariant to small translations of the input, a useful property if there is no need to know the exact location of a feature.
In order to fully introduce the CNN architecture, shown in Figure 2.19, it is necessary to finally include the *Fully connected* or *Dense* layers and the *Flattening* operation. The role of the is to take the result from the last Convolutional module in input and output a K dimensional vector, where K is the number of classes that the model has to choose from. In this process, the Flattening operation is necessary to convert the output of the Convolutional layers part into a 1D feature vector to be used by the dense layer for the final classification.
For what concerns the update of the kernel weights and the overall CNN learning process, it is important to mention the role of the activation functions such as the ReLu ($f(x) = max(0, x)$) placed between the Convolutional

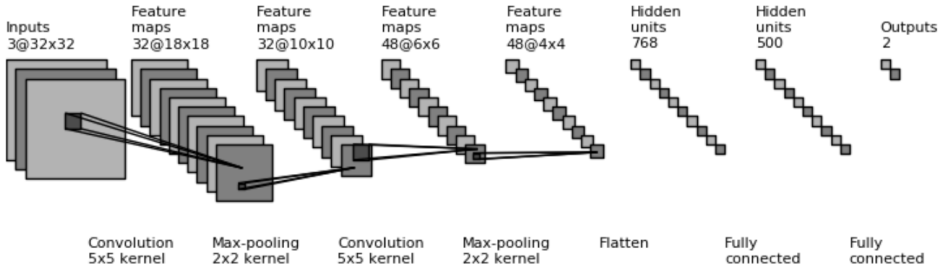

Figure 2.19: Full CNN architecture, image available at [58]

and the Pooling layer: given its non-saturating property, the vanishing gradient problem during the Backpropagation is avoided. The learning procedure of the CNNs is indeed based on the Gradient Descent algorithm and its chain rule as well as the traditional NNs: the gradient can be now interpreted as the measurement of how the change in a single pixel in the weight kernel affects the defined loss function, and when backpropagating it often risks to get smaller and smaller as the update progresses down to the lower layers. Another technique that is worth mentioning when dealing with the vanishing gradient problem is the *Batch Normalization*, that tries to avoid that the distribution of each layer's inputs changes during training due to the change of the parameters of the previous layer. It does so by zero-centering and normalizing the inputs of the current batch, then scaling and shifting the result by specific parameters.
In this context, it is worth mentioning how the Pooling layers affect the Backpropagation: in the case of Average-pooling, the error is multiplied by $\frac{1}{NXN}$, with N size of the pooling block, and it is assigned to the whole pooling block. In the case of Max-pooling, instead, the error is just assigned to the unit it comes from and thus the gradient from the next layer is passed back to the only neuron which achieved the max while all other neurons get zero gradient.

2.3.4 Generative Adversarial Networks

When it comes to Generative Adversarial Networks (GANs) [55], it is necessary in the first place to introduce the meaning of the terms *Generative* and *Adversarial*. *Generative* models are a class of statistical models different from the Discriminative ones: the former can generate new data while the latter can discriminate between different types of data. In particular, given a set of data instances X and labels Y:

- Generative models aim at capturing the joint probability $P(X, Y)$, or just $P(X)$ if no labels are available.

- Discriminative models, instead, aim at capturing the conditional probability $P(Y|X)$.

While Discriminative models just tell how likely a label is to be applied to the instance, Generative models address a more difficult task since they include the distribution of the data and tell how likely a given example is. Generative models have to model more: for instance, when they deal with images, they have to capture correlations and complex distributions related to the likelihood of objects of appearing next to each other, while Discriminative models just need to look for a few details and can ignore many of the existing correlations to perform classification. More formally, Generative models try to model how data are placed throughout the space.
For what concerns the term *Adversarial*, instead, its meaning relies on the fact that GANs are based on the idea of making NNs compete against each other during training to push them to excel.

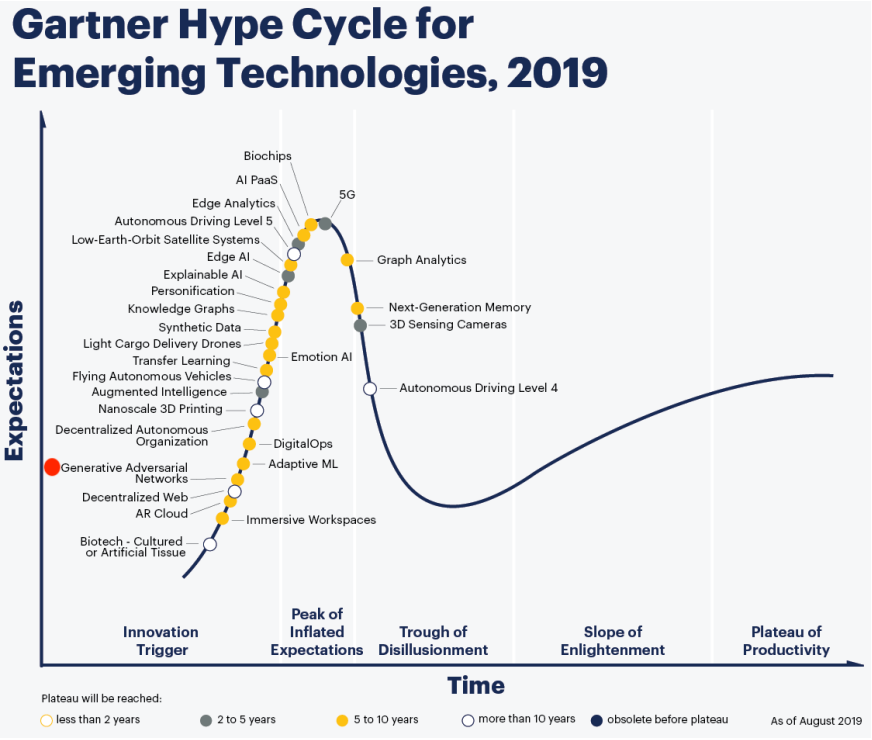

Figure 2.20: Gartner Hype Cycle 2019, image available at [59]

GANs, which according to the Gartner Hype Cycle for Emerging Technolo-

gies (Figure 2.20) represent of the most promising techniques on the rise in 2019, are composed of two NNs: a NN called Generator, that aims at generating data that look like they are drawn from to the same distribution as the training data, and a NN called Discriminator that aims at discriminating real data from fake data.
For what concerns the Generator, it takes a random noise as input (usually represented through a Gaussian distribution) and outputs an image. Random inputs can be seen as the latent representations (e.g. codings) of the image to be generated, which have a much lower dimensionality than the actual data. Thus the Generator can be used to generate brand-new images just feeding it some Gaussian noise.
For what concerns the Discriminator, instead, it takes as input either a real image from the training set or a fake image from the Generator, and behaves as a classifier in guessing whether the input image is fake or real, as visible in Figure 2.21.

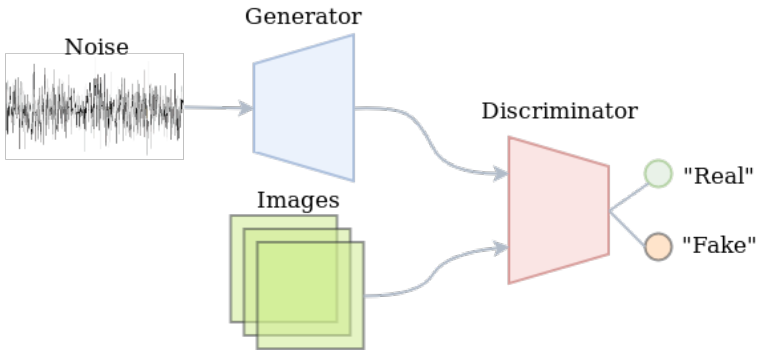

Figure 2.21: GAN overall structure, image available at [15]

The Adversarial training consists of the Discriminator trying to correctly distinguish fake images from real images, while the Generator tries to fool the Discriminator producing images that look as real as possible. The GAN is composed of two NNs with different objectives and thus is not trained as a traditional NN. The training is divided into two phases that are alternated during the process:

1. The first to be trained is the Discriminator, which receives in input an equal number of real images labeled as 1 and of fake images labeled as 0, and is trained on this labeled batch for one step, using the binary Cross-Entropy cost function. In this phase, the Backpropagation updates only the weights related to the Discriminator.

2. In the second phase, the Generator is trained: it produces a batch

of fake images and the Discriminator is asked to guess whether these images are real or fake. Differently from before, real images are not included in the batch and fake images are labeled as 1 to teach the Discriminator to wrongly classify them as real. In this procedure, the Backpropagation updates only the weights related to the Generator, while the weights of the Discriminator are frozen.

Since GANs aim at replicating a real probability distribution, loss functions that reflect the distance between the distribution of the real data and the synthetic ones must be used. One of the possible approaches is to use the Minimax loss function [55], that includes both the Discriminator's probability estimate $D(G(z))$ that a fake instance $G(z)$ is real and the Discriminator's probability estimate $D(x)$ that a real instance x is real, dropped during the training of the Generator since it can only affect the term related to the distribution of the fake data.
Mathematically, the Discriminator aims at maximizing the expected value over the real data E_x of the log probability for real images and the expected value over all generated instances E_z of the log of the inverted probabilities of fake images, while the Generator tries to minimize it:

$$E_x[log(D(x))] + E_z[log(1 - D(G(z)))]$$

The formula derives from the Cross-Entropy between the real and generated distributions, thus the problem is more commonly implemented as a traditional binary classification problem with labels 0 and 1 for synthetic and real images respectively. The GAN model is hence fit seeking to minimize the average binary Cross-Entropy, also called Log Loss as introduced in section **2.2.1**.

2.4 Representation Learning

Representation Learning [60] typically consists of transforming input data or extracting features from them, making it easier for the model to perform a task like prediction or classification. In a DL context, representation refers to the characteristics of the transformed input along the NN architecture and it can be considered a good representation if it makes the learning task easier. For instance, training a classifier in a SL manner leads to a different representation at every hidden layer, with each new representation taking on characteristics that make the classification easier.
The concept of representations can be really useful when dealing with DL architectures, since through them it is possible to share statistical strength

and acquired information across different tasks for which just a few examples are available (TL), or using knowledge from unsupervised tasks to perform supervised ones (SSL). These two approaches are further discussed in the following subsections.

2.4.1 Transfer Learning

Extensive studies about the transferability of features across DNNs have been conducted in recent years [61]. The basic idea that underlies this technique is the following: if a model to recognize some objects has already been trained and a new NN able to recognize other objects that may have some features in common with the previous ones must be trained, it is possible to kickstart the training of the new NN initializing its weights by reusing the lower layers of the first NN. In this way, the new NN will have to learn the higher-level features specific to its task but not all the low-level structures that occur in most pictures such as edges etc. An overall representation of the TL approach is shown in Figure 2.22, where a classification task is taken as an example. The number of final layers to be trained in the new NN is variable: in case we do not want to train just the final classifier but we want to go deeper and retrain some previous layers, the TL approach is referred to as Fine-tuned.

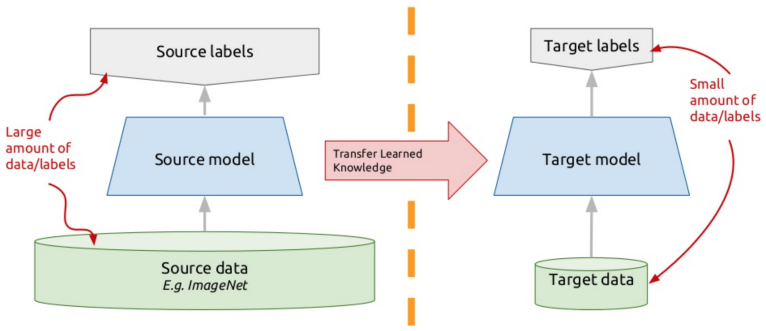

Figure 2.22: Transfer Learning overall idea, image available at [62]

2.4.2 Semi-Supervised Learning

When facing DL problems, and specifically those related to images (e.g. satellite images), it is common to have very large amounts of unlabeled data and relatively few labeled data through which train the model. As a consequence, while in such a scenario traditional SL techniques might lead

to severe overfitting, SSL techniques offer the chance to learn also from the unlabeled data [6]. Indeed, good representations learnt from the unlabeled data can be used to solve the SL task. These two steps are known as pre-training phase, which receives in input the unlabeled data, and SL phase, which receives in input the labeled data and leverages the weights received from the previous step, as shown in Figure 2.23.

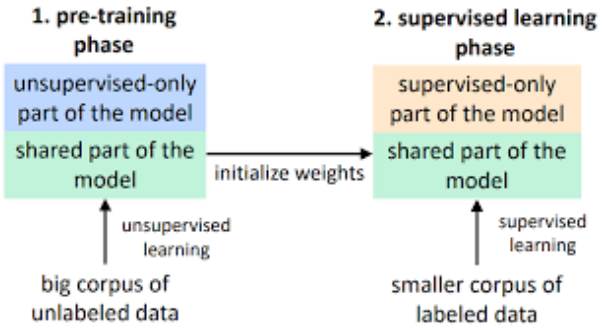

Figure 2.23: Semi-Supervised Learning overall idea, image available at [63]

Therefore, if plenty of unlabeled training data are available, they can be used in the pre-training phase to train an unsupervised model such as an Autoencoder or a GANs model, then reuse the layers of the encoder or the GAN's Discriminator in the SL phase adding the output layer for the specific task on top, and Fine-tuning the final NN.
Figure 2.24 depicts the idea that underlies the SSL-AE approach, based on a pre-training phase which consists of the Autoencoder trying to replicate in output the unlabeled images it receives in input. During this operation, the encoder learns how to perform an effective features extraction from the input samples and this knowledge is then transferred to the SL phase. The weights of the final SL model are indeed initialized with the encoder weights obtained during the pre-training phase, and an output layer such as a softmax is put on top of the architecture in case of classification. When performing the final training, the weights coming from the pre-training phase are frozen, and only the last layers involved in the target task are trained in a supervised manner.

For what concerns the SGAN approach, instead, the process is a bit different from that of the Autoencoder, although there is a strong symmetry between the role of its encoder and the SGAN Discriminator and between its decoder and the SGAN Generator respectively. In particular, the former are the architectures that are responsible to perform the actual SL task after the pre-training phase, while the latter are fundamental for the proper execution of the pre-training phase due to their capability of producing images in

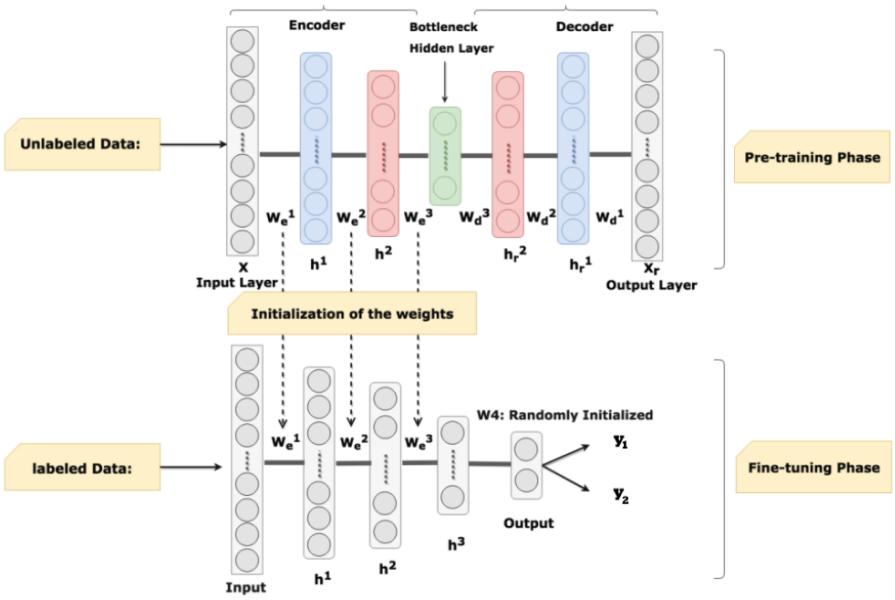

Figure 2.24: Semi-Supervised Learning with Autoencoders [64]

output.

The SGAN approach is thoroughly introduced and discussed in the later chapters together with its implementation details, since it is part of the solution proposed by this thesis.

Chapter 3

Related work

In this chapter, the existing research works in the literature that represent the foundations of this thesis are introduced and discussed.

Since the proposed method aims to apply DL techniques for EO classification (SAR images specifically) by leveraging a pipeline composed by a Representation Learning step ahead of a Supervised classification step, the related works can be grouped into DL techniques for EO classification and methods to perform the Representation Learning ahead of the SL classification step.
An overview of the state-of-the-art in the topic of DL classification for EO is presented, examining the main approaches used to handle this problem. A particular focus on polar images is also provided.
On the other hand, methods to perform the Representation Learning ahead of the SL classification are explained as well, focusing on the most relevant works for the architectures later proposed in Chapter 4.

3.1 Deep Learning for Earth Observation classification

In recent years, the interest towards the topic of DL applied to Earth-related tasks has significantly increased.
As reported by Hu et al. [65], this attitude is mainly due to the fact that former methods for solving RS-based scene classification or target recognition tasks can just generate mid-level image features with restricted representative ability thus not achieving great performance.
These methods such as Spatial Pyramid Matching Kernel [66], Spatial Pyramid Cooccurrence Kernel [67], min-tree kd-Tree [68] and Sparse Coding [33]

are mainly based on the Bag of Visual Words (BoVW) approach [69] which disregards spatial information and strongly relies on the extraction of handcrafted local features. This prevents the building of more representative higher-level features, abstractions of the lower-level features that can exhibit more discrimination playing a dominant role in scene classification tasks.
Among the possible EO scene classification tasks, sea ice detection is currently gaining more and more concern given the sudden changes that occurred in polar areas caused by global warming. Developing an automatic sea ice product for monitoring the status, movement and melting of ice caps and icebergs has become a prime necessity for the European Commission program Copernicus [2] and other related projects such as ExtremeEarth [3] that leverage on Sentinel-1 SAR data [4] to provide innovative solutions to the issue at hand.
SAR satellite images are indeed very suitable for sea ice classification and DL applications, as discussed below.

3.1.1 SAR-based Sea Ice Classification

As the review by Zakhvatkina et al. [70] supports, many works in the last decades have been carried out to derive sea ice information from high-resolution SAR images. This is due to the ability of SAR to provide day-and-night measurements regardless of any weather condition or natural illumination, maintaining a high spatial resolution.
The national ice services of Canada, United States of America, Norway, Russia and other countries have been employing for years high-resolution SAR data as the main source for monitoring sea ice cover during ships navigation and several marine operations [71][72][73][74], exploiting the continuous and intensive development of SAR observing systems simultaneously with the launch of new satellites. From the first SAR satellite SEASAT (USA) launch in 1978 up to the C-Band Sentinel-1A/B (European Commission and ESA) launches in 2014 and 2016 lot of improvements in terms of image capture have been done such as multiple SAR modes, different incidence angles, resolutions, polarisations and swath widths.
More data have become accessible to the scientific community and new approaches involving feature-based NNs and DL have been experimented, generating new research questions and results.
In this regard, Wang et al. [75] provide us with the TenGeoP-SARwv, a dataset composed of more than 37000 SAR acquisitions of Sentinel-1A in WV mode and VV polarization [4] representing ten different oceanic phenomena. This work is specifically aimed to benefit the development

of massive ocean SAR data analysis which includes DL signal processing algorithms.
In particular, the images are collected in 20 km x 20 km scenes at two alternate incidence angles of 23.8° (WV1) and 36.8° (WV2) and each pixel within each vignette has 5 m spatial resolution. The frames are also submitted to three processing steps to enhance broadscale features of oceanic phenomena and make them ready for ML purposes calibration of backscatter coefficient δ_0 [4], downsampling and normalization. The dataset is finally provided in both Portable Network Graphics (PNG) and Georeferenced Tagged Image File Format (GeoTIFF), respectively in 8 and 16 bits scales.
Among the ten categories available, this work selects the three most suitable scenes as target labels for the sea ice classification purpose: Icebergs (IB), Pure Open Waves (POW) and Sea Ice (SI). Experiments on these three cases are conducted and presented in the following chapters.

3.1.2 Convolutional Neural Networks for Earth Observation

DL in recent years has achieved state-of-art results on a variety of tasks, including visual recognition and image classification. Among the different types of NNs, CNNs have been the most extensively studied due to their promising results in various fields [76]. EO and RS, similarly to other domains close to Computer Vision, turned out to be good applications of such techniques.
As Nogueira et al. showed in their work [77], CNNs perform better than state-of-the-art solutions based on mid-level descriptors such as the already mentioned BoVW [69]. They performed experiments with six popular CNNs (including the VGG16 [17]) using three EO datasets, comparing three possible strategies for exploiting the power of the existing CNNs: full training, Fine-tuning, and using them as feature extractors. Also the study from Castelluccio et al. [78] adds another piece of evidence in this sense. This comparison is of primary importance when dealing with problems in which few labeled data are available and training a new network is a challenging task, therefore the results of these works provide important insights for the task that this thesis aims to address.
Other applications of CNNs have been proven to be successful, like that of Li et al. [16] on SAR imagery. They propose a CNN architecture called DeepSAR-Net which consists of five main building blocks composed by Convolutional, Max-pooling, Batch Normalization, ReLu and Fully-connected layers. Experiments on the Moving and Stationary Target Acquisition and Recognition (MSTAR) dataset [79] illustrate encouraging results in the EO

domain, giving further evidence of the suitability of such architectures for the task at hand. This architecture is later used in the next chapters as a component of the SGAN approach and for benchmarking purposes.
The next section investigates more thoroughly the problem of learning with scarce labeled data in the EO field and presents possible approaches and techniques drawn from related works to overcome this issue.

3.2 Representation Learning methods

EO data are nowadays quickly accessible thanks to open access data hubs that contain daily data collected from several satellites. While labeled data in this scenario are generally expensive to get, unlabeled data are plentiful and cheaper, or rather, come for free with the regular acquisitions made by the satellites.
Such a feature paves the way for the overcoming of the traditional SL in this domain, revealing the great opportunities lying in alternative approaches that involve a Representation Learning step ahead of the classification. Two promising techniques in this direction are TL and SSL, presented in the following sections introducing some works that have made use of both.

3.2.1 Transfer Learning for Earth Observation

Since not enough data are available in the target domain to generalize well in the classification task, TL aims to learn from another domain closely related to the target one, known as the source domain.
As the work from Marmanis et al. [80] suggests, when the target domain consists of EO data it is a good idea to split the model into two individual processing stages, a pre-trained model and a trainable model.
The first phase involves the use of a popular pre-trained CNN, namely the Overfeat [81], employed for generating a set of representations using a fixed set of weights obtained from the ImageNet dataset [82]. The second component of the architecture is a trainable NN that accepts as input the previously derived features, along with their respective class labels, and is trained through standard Backpropagation and Stochastic Gradient Descent.
Employing the proposed pipeline the authors successfully tackle the UC Merced Land Use aerial dataset classification problem [83], outperforming some of the mid-level descriptors already mentioned in this work [66][67][68][33][69]. Results are shown in Figure 3.1 to give an idea of how effective this approach may be.
Another interesting work on this path is the one provided by the same au-

METHOD COMPARISON OVER THE UCML BENCHMARK

Method & Algorithm	Test-set Accuracy
BOVW [2]	71.8%
SPMK [1]	74%
SPCK++ [2]	76%
Sparce Coding [4]	81.7%
Salient Unsupervised Learning [6]	82.8% ± 1.18%
MinTree + KD-Tree [3]	83.1% ± 1.2%
CNN with Overfeat feature	**92.4 %**

Figure 3.1: OA results, Marmanis et al. [80]

thors of the TenGeoP-SARwv dataset [75]. Wang et al. propose in [84] an adaptation of the Inception-v3 CNN [85] to train a model dedicated to the classification of oceanic phenomena collected in SAR mode. They examined two training strategies: TL and Fine-tuning, respectively training only the final classifier layer in the first case and Fine-tuning all the layers in the CNN architecture in the second case. Fine-tuning turned out to be the best option in terms of OA, as shown in Figure 3.2.

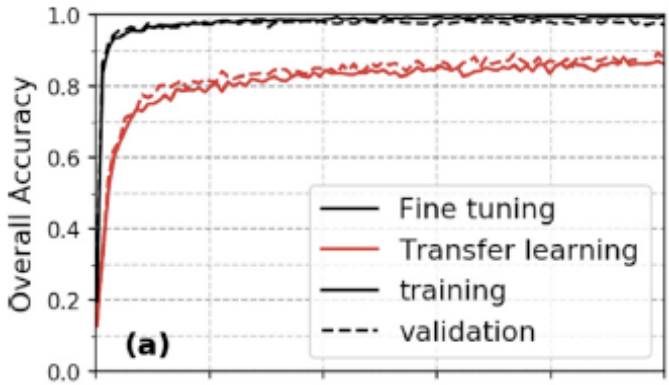

Figure 3.2: OA results, Wang et al. [84]

The experimental setup of their study is inspiring for the task at hand and is later reproduced in this thesis for benchmarking purposes.

3.2.2 Semi-Supervised Learning for Earth Observation

Since not enough labeled data are available in the EO domain to generalize well in the classification task, SSL aims to exploit the related widely available unlabeled data to improve the performance of the supervised learner. Differently from normal images, the SAR images are formed by the interaction of the transmitted microwaves with the targets and their pixels refer to the backscattering properties of the ground. Given the recognised difference between optical and SAR imagery, this section considers a large number of unlabeled SAR scene images as the source dataset instead of ImageNet, on which the TL approach is based.
There exist several methods to perform SSL: the subsections below introduce the Autoencoders-based and the GANs-based versions starting from previous work relevant to address the task of this thesis.

Autoencoders

This approach consists of training an Autoncoder to reproduce both the labeled and unlabeled inputs it receives and then using the encoder part to perform the SL step just with labeled data. The underlying hypothesis of this approach is that the latent space defined by the encoder should capture useful information about the data such that they become easily separable in the SL problem.
Huang et al. in their work [86] propose the use of a Stacked Autoencoder to address the MSTAR aerial dataset target recognition task leveraging on features extracted from the wider TerraSAR-X dataset. The results obtained by their method, reported in Figure 3.3 as "CNN-TL-bypass", are promising for the SAR domain and give useful hints for this work towards the SSL approach. It is worth noting that the gap between the fully SL approaches

Size of Training Set	all	part200	part100	part50
A-ConvNet	98.51%	98.02%	96.41%	95.34%
AlexNet-ImageNet-TL	96.62%	96.16%	95.75%	93.48%
CNN-baseline	98.88%	98.60%	98.14%	96.37%
CNN-TL	99.05%	98.88%	98.30%	97.11%
CNN-TL-bypass	**99.09%**	**99.09%**	**98.50%**	**97.15%**

Figure 3.3: OA results, Huang et al. [86]

(A-ConvNet, CNN-baseline) and the SSL one grows as the size of the labeled

data in the training set decreases.
Another interesting approach based on Variational Autoencoders is instead suggested by Kingma et al. [87], who demonstrate its effectiveness in the Computer Vision field on the MNIST [88] handwritten digits dataset, as shown by the loss values of their "M" architectures in Figure 3.4.

N	NN	CNN	TSVM	CAE	MTC	AtlasRBF	M1+TSVM	M2	M1+M2
100	25.81	22.98	16.81	13.47	12.03	8.10 (± 0.95)	11.82 (± 0.25)	11.97 (± 1.71)	**3.33** (± 0.14)
600	11.44	7.68	6.16	6.3	5.13	–	5.72 (± 0.049)	4.94 (± 0.13)	**2.59** (± 0.05)
1000	10.7	6.45	5.38	4.77	3.64	3.68 (± 0.12)	4.24 (± 0.07)	3.60 (± 0.56)	**2.40** (± 0.02)
3000	6.04	3.35	3.45	3.22	2.57	–	3.49 (± 0.04)	3.92 (± 0.63)	**2.18** (± 0.04)

Figure 3.4: Loss values, Kingma et al. [87]

Differently from the previous approach that mapped the input to a fixed latent vector, input here is mapped to a distribution and therefore the model learns the parameters of a probability distribution representing the data. Since it learns to model the data, by sampling from the distribution it is possible to generate new input data samples: for this reason it is known as a generative model, such as the GANs introduced in the next subsection.

Generative Adversarial Networks

GANs make effective use of large unlabeled data to train a Generator model via a Discriminator model with each of the two trying to fool the other one. In some circumstances the Discriminator can be used as a starting point for developing a classifier: this is the case of the SGAN, introduced for the first time by Salimans et al. [14]. This model represents an extension of the GAN architecture and implies the simultaneous training of a supervised Discriminator, an unsupervised Discriminator and a Generator model.
The proposed architecture ends up in a classification model that generalizes well on unseen samples thanks to the knowledge extracted from the unlabeled data: the Discriminator is in fact updated to predict $K + 1$ classes, where K is the number of classes in the prediction problem with an additional label representing a new "fake" class.
The outcome is a classifier that can outperform state-of-the-art solutions on Computer Vision problems such as MNIST when trained on few labeled data, as shown by the results obtained by Odena in [89] and reported in Figure 3.5.
Given its promising results in Computer Vision related tasks, the SGAN approach has been employed by Gao et al. [90], who tested out its effec-

EXAMPLES	CNN	SGAN
1000	0.965	0.964
100	0.895	0.928
50	0.859	0.883
25	0.750	0.802

Figure 3.5: OA results, Odena [89]

tiveness in the SAR imagery facing the MSTAR target recognition problem. Inspired by Salimans et al. [14], they proposed a variation of the SGAN which involves the addition of the unlabeled samples predicted as positive labels to the labeled set for the next round of training. To reduce the negative influence of those samples that are wrongly labeled, they also introduce a noisy data learning theory. The results are visible in Figure 3.6, where L stands for *Labeled data*, U for *Unlabeled data* and NDLT for *Noisy Data Learning Theory*.

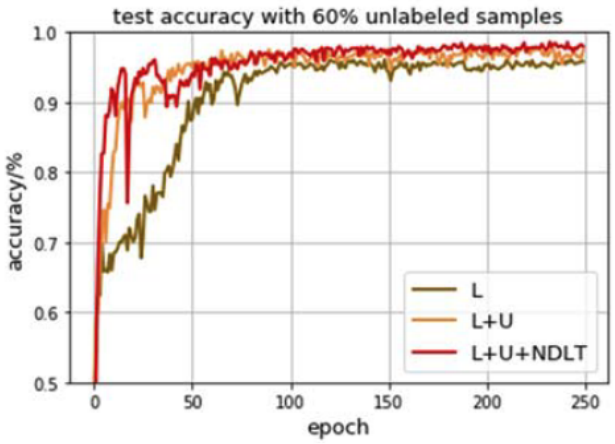

Figure 3.6: OA results, Gao et al. [90]

The encouraging results obtained by the cited works definitely make GANs worth considering as a promising solution to the problem of learning with labeled data in the sea ice classification problem on SAR imagery. Its implementation is presented and discussed in the next chapter of this work.

3.3 Related work of major interest

Analysing the related work that has been presented and its correlation with the problem under assessment, it is possible to identify which of the above-mentioned studies are the most inspiring for the task of this thesis.
In particular, the DeepSAR-Net method proposed by Li et al. [16] has been chosen as the CNN architecture that underlies the SL approach used in this work, due to the similarity between the two use cases. The DeepSAR-Net is also used in this study as the basis of the Discriminator architecture of the SGAN approach originally introduced by Salimans et al. [14], resulting in a extension of this model.
Furthermore, this work takes inspiration from the work of Wang et al. [75][84], that not only provide the TenGeoP-SARwv dataset useful for sea ice classification purposes, but also propose the application of an Inception-v3 [85]-based TL approach and its FT version on this dataset, leveraging on networks pre-trained on the ImageNet challenge. This work selects the most interesting classes for the task at hand from the TenGeoP-SARwv dataset and follows the stages of the above-mentioned TL approach to implement its VGG16 [17]-based variant. VGG16 is indeed a CNN architecture able to achieve very good results in the EO domain, as shown by Nogueira et al. [77].
The architectures involved in the cited works and their implementations are further presented and deepened in the next chapter.

Chapter 4

Deep Learning model architectures

This chapter introduces the architectures on which the whole research project is built, in such a way that they are understandable and reproducible.

The analysis of the related work and the study of the relevant literature have highlighted which of the existing models are the most promising in the context under assessment. As a consequence, these architectures have been implemented, validated and improved.
The following section introduces the DeepSAR-Net [16] for the purely SL approach, the VGG16 [17] for the TL approach and the SGAN [14] architecture for the SSL approach. In addition to the mentioned models, some variants are also presented: a modified version of each approach with the aim of including information such as geographic coordinates or incidence angle of the satellite images under evaluation; the FT version of the VGG16 in which the last few layers of the architecture are re-trained on the target dataset; the SGAN implementation which reuses the DeepSAR-Net architecture as the basis for its Discriminator. Every approach is also re-implemented substituting the final softmax layer with an SVM classifier.

4.1 Model architectures

The subsections below introduce the main NN architectures and methods on which the implementations proposed in this thesis are build on.

4.1.1 DeepSAR-Net

The DeepSAR-Net proposed by Lie et al. [16] is a CNN architecture that consists of five main building blocks composed by Convolutional (***C***), Max-pooling (***Mp***), Batch Normalization (***N***), Rectified Linear Unit (***ReLu***) and Fully-connected (***Fc***) layers. The CNN architecture built up by the authors is shown in Figure 4.1 along with the information related to the kernel size and the stride used, as well as the outputs obtained.

THE ARCHITECTURE OF DEEPSAR-NET

Type	Kernel size/Stride	Output size
C	5×5/1	20×124×124
N	- -	20×124×124
Mp	2×2/2	20×62×62
C	3×3/1	50×60×60
N	- -	50×60×60
Mp	2×2/2	50×30×30
C	3×3/1	100×28×28
N	- -	100×28×28
Mp	2×2/2	100×14×14
C	3×3/1	200×12×12
N	- -	200×12×12
Mp	2×2/2	200×6×6
C	3×3/1	400×4×4
N	- -	400×4×4
ReLU	- -	400×4×4
C	4×4/1	500×1×1
N	- -	500×1×1
ReLU	- -	500×1×1
Fc	3 or 10	#classes

Figure 4.1: DeepSAR-Net [16]

The Cross-Entropy function, already introduced in section **2.2.1**, is the cost function chosen by the authors to train their NN, whose output layer is represented by a softmax layer. The training algorithm adopted is the Mini-Batch Gradient Descent, previously described in section **2.3.2**.
The authors conduct their experiments on the MSTAR dataset [79], a group

of 128 x 128 pixels images with 1 ft x 1 ft resolution representing three different ground military targets and collected using an X-band SAR sensor. Figure 4.2 depicts some samples of the SAR images contained in the MSTAR dataset.

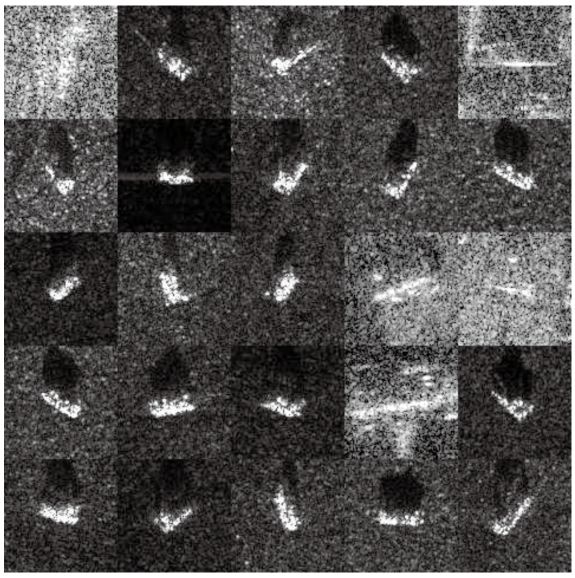

Figure 4.2: MSTAR dataset [79]

Given the promising results obtained by the just referred work in its use case, and its similarity with the task addressed by this thesis, it is reasonable to re-implement the original architecture introduced by the authors in such a way that the sea ice classification problem can be addressed effectively. The underlying hypothesis on which this choice is made is that, given the similarity between the MSTAR targets and the icebergs under a SAR perspective (the objects stand out from the background in a similar way), using the same cascade of filters and layers may result in a meaningful extraction of features, thus a meaningful classification. The resulting architectures implementations, adapted to the specific datasets under evaluation, are presented in the next chapter together with their outcomes.

4.1.2 VGG16

The VGG16 proposed by Simonyan et al. [17] is a CNN architecture which consists of twelve Convolutional layers, five Maximum-pooling layers, four Fully-connected layers and a final softmax classifier, as shown in Figure 4.3. The authors investigated the effect of the CNN depth on its accuracy during the 2014 large-scale image recognition challenge called ImageNet [82].

Layer		Feature Map	Size	Kernel Size	Stride	Activation
Input	Image	1	224 x 224 x 3	-	-	-
1	2 X Convolution	64	224 x 224 x 64	3x3	1	relu
	Max Pooling	64	112 x 112 x 64	3x3	2	relu
3	2 X Convolution	128	112 x 112 x 128	3x3	1	relu
	Max Pooling	128	56 x 56 x 128	3x3	2	relu
5	3 X Convolution	256	56 x 56 x 256	3x3	1	relu
	Max Pooling	256	28 x 28 x 256	3x3	2	relu
7	3 X Convolution	512	28 x 28 x 512	3x3	1	relu
	Max Pooling	512	14 x 14 x 512	3x3	2	relu
10	3 X Convolution	512	14 x 14 x 512	3x3	1	relu
	Max Pooling	512	7 x 7 x 512	3x3	2	relu
13	FC	-	25088	-	-	relu
14	FC	-	4096	-	-	relu
15	FC	-	4096	-	-	relu
Output	FC	-	1000	-	-	Softmax

Figure 4.3: VGG16 architecture [17], image available at [91]

As Nogueira et al. showed in their work [77], the VGG16 architecture performs well as features extractor and low and mid-level representations descriptor in the RS domain. Therefore, it is suitable to be exploited as the basis for a TL approach, especially for the low and mid-level features knowledge transfer.
Inspired by this result and by the work from Wang et al. [84] which leverages a TL approach to address the TenGeoP-SARwv classification task, this thesis validates two TL variants of the VGG16 architecture.
As visible in Figure 4.4, it is possible to freeze earlier weights in the NN ensuring that any low and mid-level features learned by VGG16 from the source domain are not destroyed, and re-training only the final Fully-connected layers responsible of performing the actual classification in the target domain through the softmax function. As a second option, it is possible to unfreeze some or all of the earlier layers in the NN and re-train them on the target task, providing a tailored or FT version of the model.
Depending on the dataset and the task under assessment, the FT-TL approach may perform better or worse than the traditional TL approach and it is usually worth validating both.
As a result, the TL approach and its FT version implemented in this thesis, adapted to the specific datasets under evaluation, are presented in the next chapter together with their outcomes.

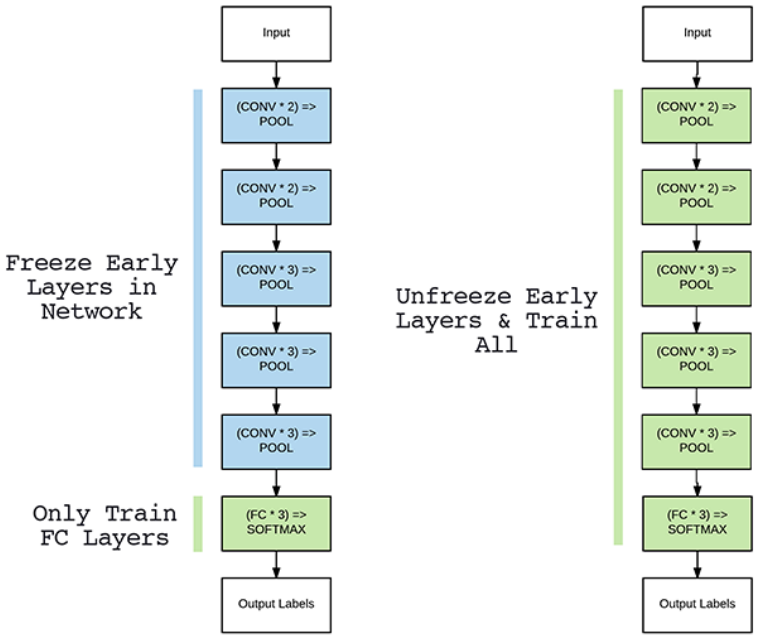

Figure 4.4: VGG16 Fine-tuning, image available at [92]

4.1.3 SGAN

SGAN is a SSL technique introduced by Salimans et al. [14] that leverages on a revised version of the GANs. This model represents an extension of the GAN architecture and implies the simultaneous training of a supervised Discriminator, an unsupervised Discriminator and a Generator model.
The proposed architecture ends up in a classification model that generalizes well on unseen samples thanks to the knowledge extracted from the unlabeled data: the Discriminator is updated to predict $K+1$ classes, where K is the number of classes in the prediction problem with an additional label representing a new "fake" class, as shown in Figure 4.5.
It is indeed possible to perform SSL with a standard classifier adding samples generated by the Generator to the dataset and labeling them with a new "fake" class $y = K+1$, while accordingly increasing the dimension of the classifier output from K to $K+1$. One can then use $p(y \in \{1, ..., K\}|x) = 1 - p(y = K+1|x)$ to represent the probability that a real instance x is real, corresponding to $D(x)$ in the original GAN (section **2.3.4**). By maximizing $p(y \in \{1, ..., K\}|x)$ is therefore possible to learn from unlabeled data, since they come from one of the K classes of real data.
Representing as $G(z)$ a fake instance and as x, y a real instance together with its label, the loss function to be minimized in order to train the classifier

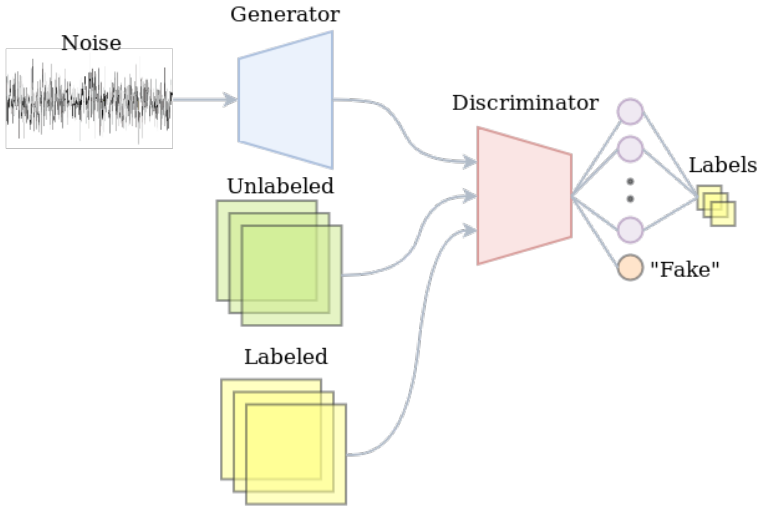

Figure 4.5: SGAN architecture, image available at [15]

becomes:

$$L = -E_{x,y}[log\, p(y|x)] - E_z[log\, p(y = K+1|G(z))] =$$

$$= L_{supervised} + L_{unsupervised}, \text{ where:}$$

$$L_{supervised} = E_{x,y}[log\, p(y|x, y < K+1)] \text{ and}$$

$$L_{unsupervised} = -E_x[1 - log\, p(y = K+1|x)] - E_z[log\, p(y = K+1|G(z))]$$

Once the total Cross-Entropy cost function is decomposed into the supervised loss $L_{supervised}$ and the unsupervised loss $L_{unsupervised}$, its equality with the standard cost function from section **2.3.4** becomes evident substituting $D(x) = 1 - p(y = K+1|x)$ into $L_{unsupervised}$:

$$L_{unsupervised} = E_x[log(D(x))] + E_z[log(1 - D(G(z)))]$$

$L_{unsupervised}$ has a positive impact if it is not too trivial for the classifier to minimize, thus the Generator is trained to best approximate the data distribution by minimizing the traditional GAN cost function through the Discriminator defined by the classifier.
The Discriminator undergoes some variations from its original GAN architecture: it is formed by two logically separated supervised and unsupervised models, attempting to reuse the output layers of the former as input to the latter. At first, the supervised model is created with K output classes and a softmax activation function, while in a second step the unsupervised model takes the output of the supervised one before the softmax and computes a

normalized sum of the exponential outputs to produce its own output "fake" or "real". This is done through a custom activation function, computed as follows:

$$D(x) = \frac{Z(x)}{Z(x)+1}, \text{ where:}$$

$$Z(x) = \sum_{k=1}^{K} e^{w_k^T x}$$

Once the loss function is defined and the model is built, the adversarial training between Generator and Discriminator takes place. This process can be formalized as follows:

Batch length: m
Iterations: I= epochs $*$ number of batches in each epoch;
for $i \leftarrow 1$ **to** I **do**
  Draw $\frac{m}{2}$ examples $\{(x^{(1)}, y^{(1)}), ..., (x^{(m)}, y^{(m)})\}$ from data generating distribution $p_d(x)$.
  Draw $\frac{m}{4}$ examples $\{x^{(1)}, ..., x^{(m)}\}$ from data generating distribution $p_d(x)$, coupled with true labels.
  Draw $\frac{m}{4}$ noise samples $\{G(z)^{(1)}, ..., G(z)^{(m)}\}$ from noise prior $p_g(z)$, coupled with false labels.
  Perform Gradient Descent on the parameters of the Discriminator w.r.t. the Cross-Entropy of the Discriminator outputs on the combined minibatch of size m.
  Draw m noise samples $\{G(z)^{(1)}, ..., G(z)^{(m)}\}$ from noise prior $p_g(z)$, coupled with true labels.
  Perform Gradient Descent on the parameters of the Generator w.r.t. the Cross-Entropy of the Discriminator outputs on the minibatch of size m.
end

Algorithm 1: SGAN Training Algorithm, revised from [89]

Every cycle includes first updating the supervised Discriminator with labeled examples and the unsupervised one with generated and unlabeled real examples, after which the is the turn of the Generator. This is done following the same labeling principles for both Discriminator and Generator trainings already described in section **2.3.4**.
Given the task faced by this thesis, the CNN architecture that underlies the Discriminator NN is chosen in such a way that it is capable of dealing effectively with SAR satellite images. In particular, the DeepSAR-Net architecture [16] is used to perform the role of the Discriminator and thus the classifier. For the Generator, a symmetric Deconvolutional version of the

DeepSAR-Net is provided.
The Generator is further modified compared to the common Deconvolutional version of the DeepSAR-Net approach to build stable CNN GANs. Relevant suggestions in this sense are provided by Radford et al. [93] and shown in Figure 4.6.

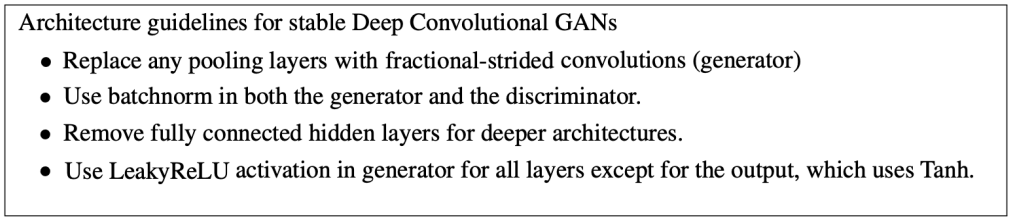

Figure 4.6: GANs convergence suggestions, revised from [93]

As a result of the SGAN dependency to the DeepSAR-Net architecture and these convergence suggestions, the final SGAN implementation, adapted to the specific datasets under evaluation, is presented in the next chapter together with its outcomes.

4.2 Special configurations

The subsections below present the main NN modifications that have been made to the architectures introduced in the section above to provide alternative approaches and possible performance enhancements.

4.2.1 Combining CNN features with scalar inputs

The first variation of the approaches that have been earlier introduced consists of concatenating the features given in output by the CNN architecture before the final classification with additional scalar information given in input to the model, when available.
This scalar information may consist of geographic coordinates when dealing with geo-referenced images, or additional data such as incidence angles for SAR satellite images. These data, pre-processed and normalized to interact properly with the features given in output by the CNN, are concatenated in a Dense layer and later used for the final classification task, as shown in Figure 4.7 .
This concatenation often results in improved classification performance, depending on the actual usefulness of the additional information provided for the task under evaluation. More specifically, it turns out to be very effective

when few samples are given in input to the CNN architecture and thus this additional information helps the model to enhance the classification performance. As a result, this approach, implemented and adapted to the specific datasets under evaluation in this thesis, is presented in the next chapter together with its outcomes.

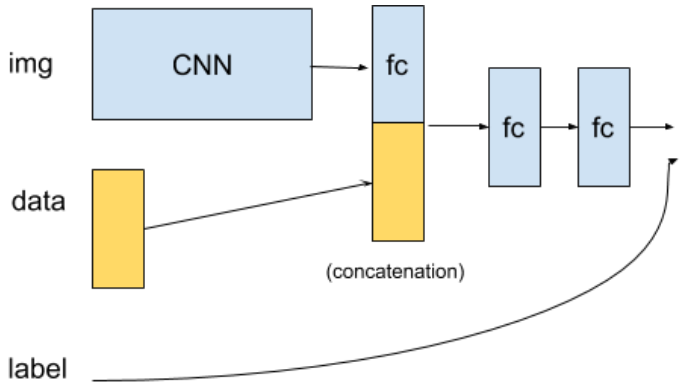

Figure 4.7: CNN features and scalar inputs concatenation [17], image available at [94]

4.2.2 Substituting the softmax layer with an SVM classifier

The second variation of the approaches that have been earlier introduced consists in substituting the final softmax classifier with an SVM classifier, with the aim of achieving high classification performance even when few samples are given in input to the CNN architecture. This is mainly due to the capability of the SVM of generalizing with datasets of small size: it is a kernel-based method and as such works better with few data with a lot of features than parametric methods such as the Logistic Regression.
At first, the CNN architecture is trained with the softmax classifier, then the features obtained before the softmax layer are extracted and given in input to the SVM classifier, which is trained on them. The overall idea is shown in Figure 4.8.
This approach, implemented and adapted to the specific datasets under evaluation in this thesis, is presented in the next chapter together with its outcomes.

4.3 Summary of the compared architectures

Table 4.1 gives an idea of how the main comparison among the previously introduced architectures is conducted and presented in the next chapter.

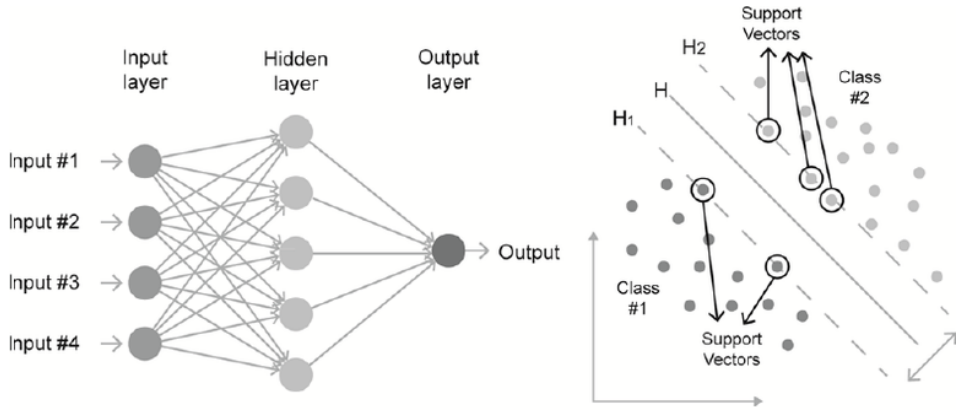

Figure 4.8: Softmax layer substituted by an SVM, image available at [95]

Models / #Labels	**SL (DeepSAR-Net) [16]**	**TL (VGG16-ImageNet [17])**	**TL-FT (VGG16-ImageNet [17])**	**SSL (DeepSAR-Net-SGAN)**
L	SL performance with L labels	TL performance with L labels	TL-FT performance with L labels	SSL performance with L labels
L_1	SL performance with L_1 labels	TL performance with L_1 labels	TL-FT performance with L_1 labels	SSL performance with L_1 labels
L_2	SL performance with L_2 labels	TL performance with L_2 labels	TL-FT performance with L_2 labels	SSL performance with L_2 labels

Table 4.1: Comparison of the models at different labeled samples sizes L_n

The models (on the X axis) are compared based on the classification performance that they achieve downgrading the amount of labeled samples at their disposal at training time (on the Y axis), thus each row of the table represents a different experimental configuration.
In particular, the amount of labeled samples used for training the models gets progressively decreased from the whole labeled dataset size (**L**) to smaller amounts of labeled data ($\mathbf{L_1}$ and $\mathbf{L_2}$, with $\mathbf{L_2} < \mathbf{L_1} < \mathbf{L}$), showing the related performance degradation. The resulting table is filled with the classification performance (mainly OA but also F_1 score) achieved by each model in each experimental configuration.
The same table structure is also used for enriching the quantitative analysis of the architectures under consideration with the comparison among their training times and their related CO_2 emissions.
Furthermore, additional analyses for each of the compared models are presented, exploring the special configurations of such architectures which involve the concatenation of additional information to the models or the substitution of the final softmax layer with an SVM classifier.
Finally, a qualitative analysis of the results obtained by the assessed models is included to complete the experimental evaluation of the research project.
The next chapter delves into the evaluation of the experiments conducted in the thesis, showing the experimental setups, the implementation of the presented models and their results through the above-described comparative framework.

Chapter 5

Experimental evaluation

This chapter aims at fully introducing the experimental setup of this thesis, including the datasets under assessment, the implementations of the model architectures tailored on these datasets and their results, which are presented through a comparative framework.

5.1 Datasets

The subsections below illustrate the two datasets that have been used for each experiment run during this project.

5.1.1 TenGeoP-SARwv

This is a labeled dataset of 37560 SAR images provided by Wang et al. [75] which contains Sentinel-1A WV acquisitions representing ten observed atmospheric and ocean-related physical phenomena.
Sentinel-1 WV vignettes, as already introduced in section **2.1.2**, are acquired alternating center incidence angles of 23.8° (WV1) and 36.8° (WV2) every 100 km along the satellite flight track. Each vignette has a 5 m spatial resolution with a 20 x 20 km footprint. This dataset focuses on the VV (default) polarized SAR vignettes.
The images are derived from the SLC processing of Sentinel-1 WV and are provided in both PNG and GeoTIFF versions, with PNG pixel values normalized into an 8 bits greyscale ([0, 255]) while GeoTIFF pixel values are instead normalized on 16 bits ([0, 65535]). Despite the higher precision maintained by the GeoTIFF files, PNGs are more suitable for visual interpretation and fulfill the input requirements for the CNN models implemented in this thesis.

Given the 8 bits greyscale ([0, 255]) nature of the PNG images, these needed to be pre-processed before being processed by any CNN architecture. In particular, this step consisted in normalizing the inputs in the range {-1,+1}. Besides, given the different sizes of the original images (varying around 500 x 500), they have been first resized to size 480 x 480 and then downsampled through the PIL library to fixed-size 240 x 240. This was done to guarantee the proper input dimensions to the models that have been used in the experiments. The original ten classes denoted by Wang et al. are depicted in Figure 5.1 together with their class names.

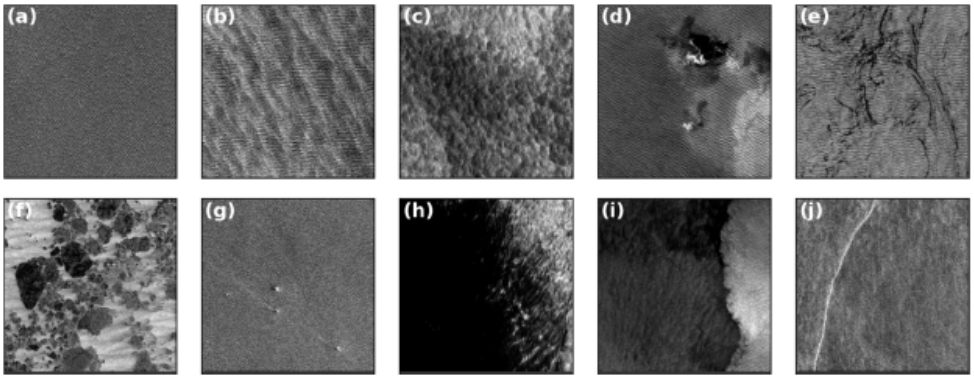

Figure 5.1: From (a) to (j): Pure Ocean Waves (POW), Wind Streaks (WS), Micro Convective cells (WC), Rain Cells (RC), Biological Slicks (BS), Sea Ice (SI), Icebergs (IB), Low Wind area (LW), Atmospheric Front (AF) and Oceanic Front (OF), image available at [75]

As visible in Figure 5.1, the vignettes are labeled based on the geophysical phenomenon that dominates with its specific pattern. In this regard, POW signatures usually exist in SAR images as background for the other nine classes.
Among the ten classes presented above, this thesis selects the three of greatest interest to tackle the iceberg and ice caps detection problem: Icebergs (IB), Pure Ocean Waves (POW) and Sea Ice (SI).
The overall amount of images composed of these three classes is 11.250, distributed as follows: 1980 IB, 4900 POW and 4370 SI images. Samples drawn from the dataset are shown in Figure 5.2.
Moreover, geographic coordinates related to these vignettes are available. Since there is no WV acquisition in areas such as the Arctic Ocean or the Mediterranean, Red, Caribbean and Black seas, the dataset does not cover every existing latitude and longitude bands. Existing ice caps and icebergs in the Arctic ocean are indeed not present in the dataset, as shown in Figure

Figure 5.2: The three TenGeoP-SARwv classes under assessment

5.3. Looking at the Figure, which depicts the geographic normalized coordinates of the three classes, it is also noticeable that, as expected, several icebergs positions almost overlap with those of the ice caps.

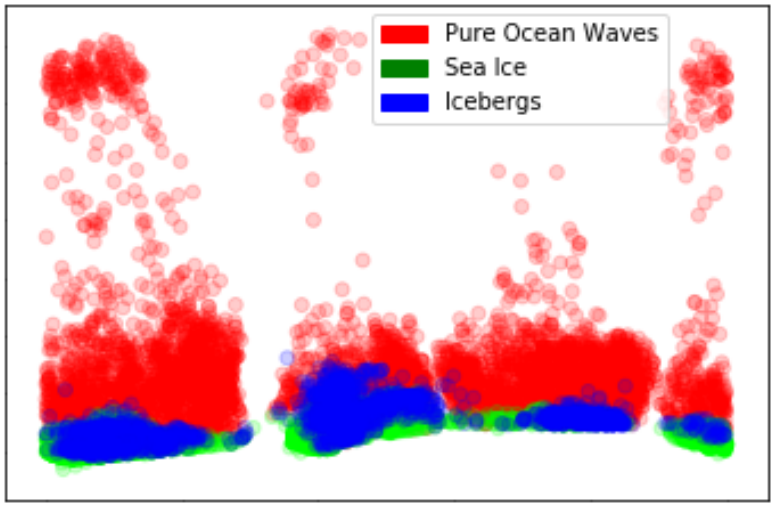

Figure 5.3: Geographic coordinates of the three TenGeoP-SARwv classes

Given the usefulness of the geographic coordinates of the three classes, especially in case of a model that monitors the Antarctic area, these are included in a variation of the models proposed in this thesis which concatenate this scalar information with the CNN features extracted from the images. The obtained results are shown in the next sections.

5.1.2 C-CORE

This dataset has been created by C-CORE [96], a R&D corporation experienced in the satellite data domain, in collaboration with the international energy company Equinor (ex Statoil) [97] to detect drifting icebergs that can threaten navigation in polar areas. They leveraged on the crowdsourcing platform for predictive modeling Kaggle to make the dataset publicly available to the data scientists eager to tackle the problem [98].
The dataset is composed of 1604 labeled images acquired by the Sentinel-1 satellite in Strip Map mode and representing either a ship or an iceberg (respectively 851 and 753 examples for each class). Each image is composed of the following fields:

- Band_1, Band_2: 75 x 75 pixel values representing the signals produced by SAR backscatter from different polarizations and measured in dB. Band_1 corresponds to HH polarization, while Band_2 corresponds to HV polarization;
- Band_3: 75 x 75 pixel values not originally present in the dataset and produced by the author of this thesis as the average of the two previous Bands: $Band\_3 = \frac{Band\_1+Band\_2}{2}$ to deal with the resulting images as they were 3-channels images, such as Colour Composite RGB ones;
- Inc_angle: the angle between the vertical to the terrain and the vector to the location on the Earth where the radar points;
- Is_iceberg: the label or target value, set to 1 in case of icebergs and 0 in case of ships;

Examples retrieved from the dataset are depicted in Figure 5.4.
As visible, these are challenging objects to classify and the polarization and the incidence angle of the images play an important role in the categorization since the icebergs and ships tend to reflect energy differently. Figure 5.5 presents a 3D visualization of the objects at hand which may help to grasp the main difference between the two: the ships follow a way more regular reflecting behaviour than the icebergs.
Band_1,Band_2, and Band_3 have physical meaning and thus their values are not the non-negative integers commonly used in image files. These are float numbers that need to be pre-processed before being processed by any CNN architecture. In particular, this step consisted of normalizing the inputs in the range {-1,+1}, as visible from Figure 5.5.
The incidence angle field, similar to the role that the geographic coordinates have played in the TenGeoP-SARwv dataset, is used as further information

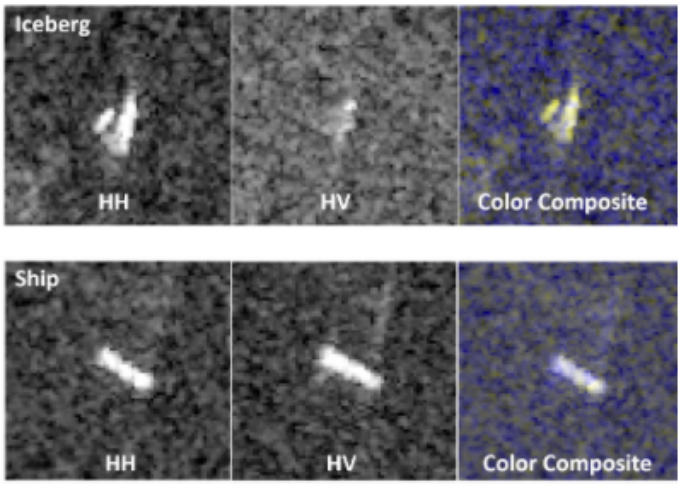

Figure 5.4: Iceberg vs Ship, image available at [98]

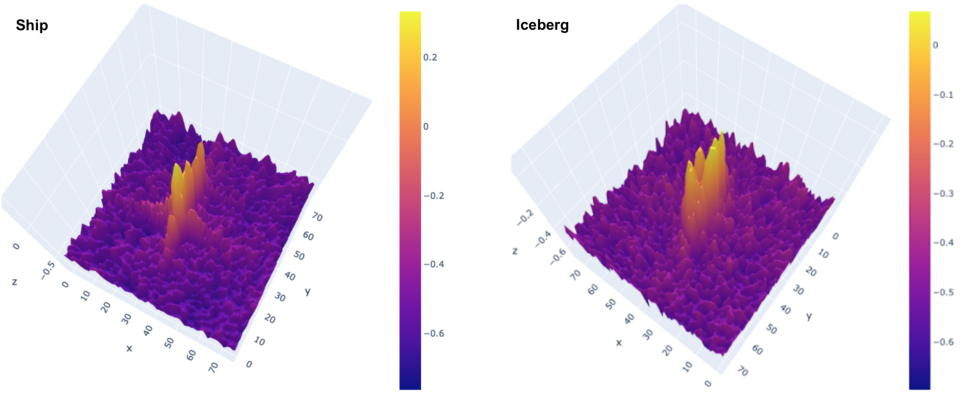

Figure 5.5: 3D visualization of Icebergs and Ships

used for improving the classification performance. Thus, it is included in a variation of the models proposed in this thesis concatenating this normalized scalar information with the CNN features extracted from the images. The obtained results are shown in the next sections.

5.2 Experimental settings

All the experiments included in the following subsections have been conducted on Google Colaboratory, a free platform that offers computational resources useful for ML and DL purposes. In particular, the Python 3.0

Jupyter Notebooks hosted on Google Colaboratory which have been used are provided with a Tesla P100 16GB PCIe GPU.
The main libraries that have been used include the Keras library on a Tensorflow backend v. 1.15.2 and the NumPy, Pandas, Matplotlib, SciPy and Scikit-Learn libraries.
The following subsections fully introduce the experimental setup and the architectures' implementations.

5.2.1 Experimental setup

This subsection aims at presenting the setup of the experiments for each of the two datasets, whose pre-processing has been already introduced in section **5.1**.

First setup: TenGeoP-SARwv

The 3 classes used are Icebergs (IB), Pure Ocean Waves (POW) and Sea Ice (SI), in total 11250 images. The dataset has been split in train, validation and test set to comply with ML and DL good practices. The split of train, validation and test sets is 70%, 15%, 15% of the total (7875, 1688, 1687), maintaining the same proportion of samples for each class performing a stratification procedure. In particular:

- POW: 4900 images (3455, 723, 722);
- IB: 1980 images (1408, 286, 286);
- SI: 4370 images (3012, 679, 679).

The architectures that have been implemented, validated and compared are the DeepSAR-Net for the SL approach, the VGG16 and its FT version for the TL approach, the DeepSAR-Net-SGAN for the SSL approach. The geo-based versions of each of these approaches have been tested out for benchmarking purposes, as well as their SVM-based classifier versions.
Experiments have been conducted selecting at training time respectively 7875, 4000, 2000, 500, 300, 100, 70, 40, 10 labeled images from the dataset, reproducing every above-mentioned approach each time. The SGAN approach, in particular, has taken in input both the selected labeled data and the whole dataset treating it as unlabeled data.
Moreover, for each configuration of selected labels used, the experiments are reproduced three times to counterbalance the stochasticity of the training process and the maximum value is recorded. Also, three different train-validation splits have been created to cross-validate the model, tune the

related hyperparameters and select the model with maximum validation performance as model to be finally evaluated on the test set. The evaluation of such metrics is done on the test set, while the training process mostly involves both training and validation sets since it implements the Early Stopping technique. This is valid for every approach except the SGAN, which, based on the adversarial training between Discriminator and Generator, could not exploit this technique.
Furthermore, experiments have been mostly conducted with maximum batch size = 100 and maximum number of epochs = 100. The batch size parameter has been reduced accordingly when dealing with labeled dataset sizes lower than 100 samples, while the epochs parameter has been subject to the Early Stopping behaviour in both the SL and the TL approaches, concluding the training after just a few epochs when the labeled data at disposal were not many.
At last, given the dataset unbalanced classes distribution, the OA may not be enough to evaluate the goodness of the models. Therefore Precision, Recall, F_1 score and Confusion Matrix are used. In addition, Class Activation Maps have been used to further compare the behaviour of the several approaches, since they indicate the image regions that the CNN activates along the NN to discriminate each of the categories under assessment.

Second setup: C-Core

The 2 classes are Icebergs and Ships, in total 1604 images. As already done in section **5.2.1**, this dataset has been split in train, validation and test to comply with ML and DL good practices. The split of train, validation and test sets is 70%, 15%, 15% of the total (1124, 240, 240), maintaining the same proportion of samples for each class. In particular:

- Icebergs: 753 images (541, 106, 106).
- Ships: 851 images (583, 134, 134).

To check the consistency of the results obtained in the first use case (section **5.2.1**), the architectures that have been implemented, validated and compared in this use case are the same ones: the DeepSAR-Net for the SL approach, the VGG16 and its FT version for the TL approach, the DeepSAR-Net-SGAN for the SSL approach. The incidence angle-based versions of each of these approaches have been tested out for benchmarking purposes, as well their SVM-based classifier versions.
Experiments have been conducted selecting at training time respectively

1124, 800, 500, 300, 100, 70, 40, 10 labeled images from the dataset, reproducing every above-mentioned approach each time. The SGAN approach, in particular, has taken in input both the labeled data and the unlabeled data that for each configuration are excluded from the labels' selection.
As well as in section **5.2.1**, three experiments for each of the configurations have been repeated, three different train-validation splits have been created and the results have been computed in the same way. Besides, the Early Stopping technique has been used during the training phase, whose parameters have been the following: maximum batch size = 100 and maximum number of epochs = 100.
At last, given the dataset balanced classes distribution, the OA has been considered enough to evaluate the goodness of the models. Class Activation Maps have been then used to further compare the behaviour of the approaches.

5.2.2 Implementation details

This section aims at showing the implementation details of the architectures presented in chapter **4**, tailored on the datasets described in the previous section and enriched with specific implementation choices such as hyperparameters selection.

DeepSAR-Net

The standard DeepSAR-Net input shape is (124,124,1), hence it has been necessary to adapt this architecture to the TenGeoP-SARwv (240,240,1) and C-CORE (75,75,3) shapes respectively. In particular, the DeepSAR-Net architecture dedicated to the TenGeoP-SARwv use case has been enriched with a Max-pooling layer before the first Convolutional layer, while the number of channels has been extended to 3 in order to deal with the C-CORE use case.
For what concerns the training process, the standard Stochastic Gradient Descent optimizer ($lr = 0.01$) has been used. To avoid overfitting, the Early Stopping technique ($min\_delta = 0.01$, $patience = 10$) has been implemented through the `keras.callbacks` functions.
The DeepSAR-Net geo-based version tailored for the TenGeoP-SAR use case includes the latitude and longitudes information separately into the model through the `keras.layers.concatenate` function. The DeepSAR-Net incidence angle-based version tailored for the C-CORE use case, similarly, includes the incidence angle information. In particular, the concatenation takes place before the softmax layer, when the image high-level features are

fully extracted from the CNN.
At last, the SVM-based classifier version of the DeepSAR-Net involves the substitution of the final softmax layer with the SVM classifier imported from the `sklearn.svm.SVC` class. In detail, a first training process based on the softmax layer is computed, then the softmax layer is removed and the features obtained from the Dense layer prior to the softmax one are given in input to the SVM classifier, which is trained from scratch based on the RBF kernel.
The standard model architecture, its geo-based and incidence angle-based versions are included in **Appendix A**, together with their number of trainable parameters.

VGG16

The standard VGG16 input shape is (224,224,3), hence it has been necessary to adapt this architecture to the TenGeoP-SARwv (240,240,1) and C-CORE (75,75,3) shapes respectively. In particular, the TenGeoP-SARwv images have undergone a replication of the greyscale channel in order to deal with the 3 input channels required by the architecture.
For what concerns the training process, the parameters of the VGG16 layers prior to the softmax layer have been transferred from its ImageNet challenge pre-trained version, while the softmax layer has been re-trained from scratch. In this configuration, the RMSProp optimizer ($lr = 0.001$ for TenGeoP-SARwv and $lr = 0.01$ for C-CORE, $decay = 1^{-6}$) has been used. To avoid overfitting, a Dropout layer ($\alpha = 0.5$) has been included before the softmax layer and the Early Stopping technique ($min\_delta = 0.01$, $patience = 10$) has been used.
A FT version of the VGG16 has also been proposed, freezing the pre-trained weights up to the penultimate Convolutional layer and Fine-tuning the remaining layers from scratch. This has been done using a $lr = 0.0001$ for TenGeoP-SARwv and $lr = 0.01$ for C-CORE.
The VGG16 geo-based and incidence angle-based versions, along with its SVM-based classifier version, have been managed exactly as already done with the DeepSAR-Net architecture.

DeepSAR-Net-SGAN

The DeepSAR-Net-SGAN approach leverages the DeepSAR-Net architecture for implementing its Discriminator, hence it must be tailored on the datasets similarly as it has been done with the DeepSAR-Net. The standard DeepSAR-Net input shape is (124,124,1), hence it has been necessary

to adapt this architecture to the TenGeoP-SARwv (240,240,1) and C-CORE (75,75,3) shapes respectively. In particular, the DeepSAR-Net-SGAN architecture dedicated to the TenGeoP-SARwv use case has been enriched with an additional Max-pooling layer at the beginning of the NN, while the number of channels has been extended to 3 in order to deal with the C-CORE use case.
Moreover, given the convergence suggestions presented in section **4.1.3**, LeakyReLu activation functions have been used instead of the DeepSAR-Net traditional ReLu activation functions for the Generator. The Generator architecture has been properly defined as the symmetric version of the Discriminator, characterised by Deconvolutional layers `keras.layers.Conv2DTranspose` instead of the Convolutional and Max-pooling ones.
For what concerns the training process, the Adam optimizer ($lr = 0.002$, $beta\_1 = 0.5$) has been used for both the unsupervised Discriminator (loss = binary_crossentropy)and the supervised Discriminator (categorical_crossentropy); the Generator (binary_crossentropy) uses a slightly different Adam optimizer ($lr = 0.02$, $beta\_1 = 0.5$). As already introduced in section **4.1.3**, a custom activation function generates the output of the unsupervised Discriminator taking in input the outputs of the supervised one prior to the softmax layer. In addition,to avoid overfitting, a Dropout layer ($\alpha = 0.4$) has been included before the Discriminator softmax layer.
The geo-based and incidence angle-based versions of the DeepSAR-Net-SGAN, along with its SVM-based classifier version, have been managed in a slightly different way compared to the DeepSAR-Net architecture. These variants are only related to the supervised Discriminator, which after being normally trained on images following the SGAN approach undergoes some changes. In particular, each of the variants re-train just the final layers: in case of geo-based and incidence angle-based versions, this information are concatenated to the features extracted prior the softmax layer and then the classifier is re-trained; in case of SVM-based classifier, the solution is the same as previous approaches and it substitutes the softmax layer with the SVM classifier, which takes in input the features extracted prior the softmax and train the SVM on them.

5.3 Analysis of the results

This section aims at presenting the results of the experiments. The results are obtained exploiting the models on the test set and evaluated through a quantitative analysis based on the evaluation metrics introduced in section **2.2.2** and through a qualitative analysis.
The majority of the experiments that have been conducted give in output the value of the chosen evaluation metrics at the variation of the amount of labeled data received in input by the assessed models during the training phase. Evaluation metrics such as OA and F_1 score have been monitored during the labeled dataset size variation, and interesting outcomes have been revealed. Also, training times have been recorded. Besides, Class Activation Maps of the NNs under comparison are shown when useful to enhance the interpretability of the results.
Moreover, the impact of the additional information given in input to the models (e.g. geographic coordinates or incidence angle) has been experienced, as well as the SVM classifier impact.
At last, a focus on the DeepSAR-Net-SGAN approach has been done, testing different sizes of unlabeled data given in input at training time and monitoring how the evaluation metrics are impacted. In addition, images generated by the DeepSAR-Net-SGAN Generator are introduced.
In all the experiments the DeepSAR-Net approach is referred to as SL, the VGG16 approach as TL, the VGG16 FT approach as TL-FT, the DeepSAR-Net-SGAN approach as SSL.
In the following subsections the results of the experiments run on each dataset are presented, shown in both tabular and graphic representations.

5.3.1 TenGeoP-SARwv quantitative analysis

Here follow the experiments and the related performance metrics results chosen to address the TenGeoP-SARwv classification problem. A quantitative analysis of the results is provided for each experiment.

Overall Accuracy

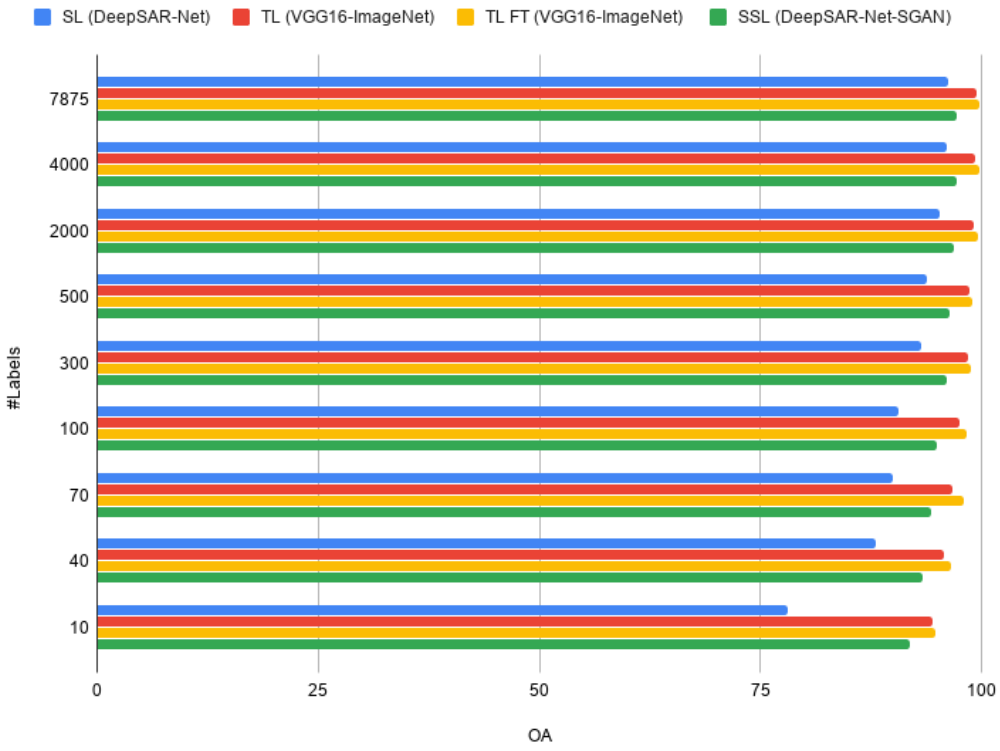

Figure 5.6: OA score at different labeled samples sizes

Models / #labels	**SL (DeepSAR-Net) [16] (%)**	**TL (VGG16-ImageNet [17]) (%)**	**TL-FT (VGG16-ImageNet [17]) (%)**	**SSL (DeepSAR-Net-SGAN) (%)**
7875 (whole dataset)	96.29	99.44	99.76	**97.29**
4000	96.19	99.37	99.71	**97.25**

2000	95.36	99.15	99.64	**96.89**
500	93.88	98.74	99.05	**96.42**
300	93.25	98.48	98.87	**96.13**
100	90.30	97.53	98.43	**94.98**
70	89.77	96.75	98.14	**94.37**
40	88.17	95.87	96.65	**93.41**
10	72.40	94.47	94.81	**91.94**

Table 5.1: OA score at different labeled samples sizes

Figure 5.6 and Table 5.1 show the OA results obtained by the models under assessment varying the size of the labeled data given in input to them at training time. For each configuration, the SSL approach exploits not only the restricted pool of selected labeled data, but also the whole dataset removing its labels and treating it as unlabeled data.
As visible in the last stages (when very few labeled data are available), the TL and the SSL approaches outperform the traditional SL approach. In the early stages, instead, the SL model is almost as accurate as the other approaches thanks to the bigger amount of labeled information given to the models.
The TL approach is very effective in every configuration. In particular, its FT version improves as more labeled information is given to the model since it is tailored to the higher-level features of the specific use case. This boosts its performance even more.
The SSL approach is way more effective than the SL approach with few labeled data. As more information is given, it improves and almost achieves the TL results, while the gap with the SL approach is reduced due to the high amount of labeled data given during the training phase.
The results are computed on the test set, as previously described in **5.2.1**. Further details on the classification performance obtained by the assessed models are given in the sections below.

Confusion Matrix, OA, Precision, Recall, F_1 score: SL (DeepSAR-Net [16]) vs SSL (DeepSAR-Net-SGAN) with 10 labeled samples

DeepSAR-Net results with 10 labels given

	precision	recall	f1-score	support
POW	0.75	0.97	0.84	727
SI	0.99	0.56	0.71	685
IB	0.36	0.47	0.41	276
accuracy			0.72	1688
macro avg	0.70	0.66	0.65	1688
weighted avg	0.78	0.72	0.72	1688

Confusion Matrix

	Predicted POW	SI	IB
POW	702	0	25
SI	93	382	210
IB	144	2	130

Figure 5.7: Confusion Matrix, OA, Precision, Recall, F_1 score: SL (DeepSAR-Net [16]) with 10 labeled samples

DeepSAR-Net-SGAN results with 10 labels given

	precision	recall	f1-score	support
POW	0.92	0.93	0.93	727
SI	0.98	0.98	0.98	685
IB	0.76	0.75	0.75	276
accuracy			0.92	1688
macro avg	0.89	0.89	0.89	1688
weighted avg	0.92	0.92	0.92	1688

Confusion Matrix

	Predicted POW	SI	IB
POW	673	0	54
SI	1	671	13
IB	54	14	208

Figure 5.8: Confusion Matrix, OA, Precision, Recall, F_1 score: SSL (DeepSAR-Net-SGAN) with unlabeled and 10 labeled samples

Figure 5.7 and 5.8 show the Confusion Matrix, F_1 score, Precision and Recall of the DeepSAR-Net and the DeepSAR-Net-SGAN approaches trained with 10 labels. This comparison clearly shows the contribution given by the unlabeled data in the SSL performance.
As visible, both OA and F_1 score results of the two approaches are different. In particular, it is worth focusing on the icebergs F_1 score, which is a very relevant metric for the problem at hand. Due to the big similarity between POW and IB images, icebergs are the most difficult class to distinguish and thus the most prone to suffer from false negatives (real IB misclassified as belonging to another class) and false positives (classified as IB but belonging to another class) issues.
In this scenario, a big improvement in terms of icebergs F_1 score is done from the SL to the SSL approach thanks to the exploitation of unlabeled data. This helps in understanding the different OA results obtained by the two approaches.
An overall improvement in moving from the SL approach to the SSL approach can be also seen in the other metrics shown above. The same behaviour can be noticed in other configurations characterised by a bigger amount of labeled data available, but it becomes less noticeable the more labeled data are given to the models.

IB class F_1 score

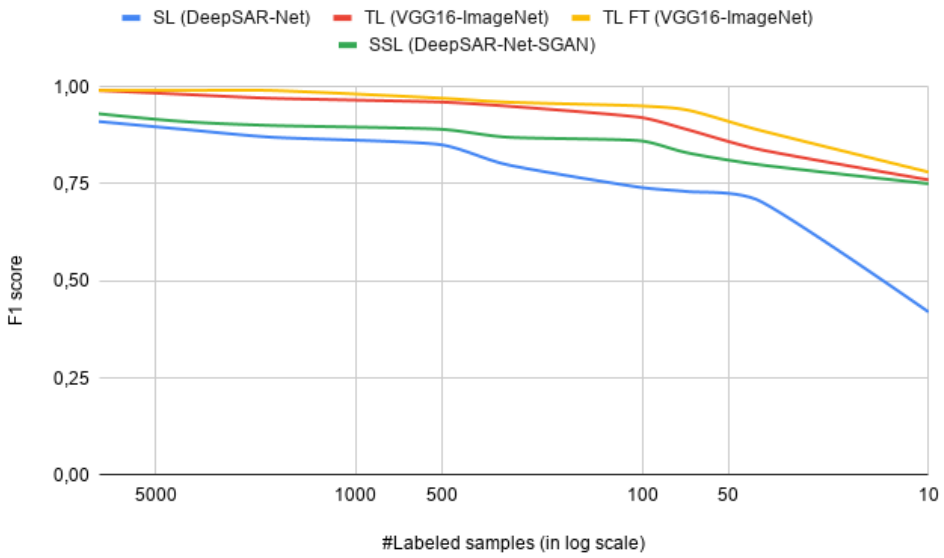

Figure 5.9: IB F_1 score at different labeled samples sizes

Models / #labels	**SL (DeepSAR-Net) [16]**	**TL (VGG16-ImageNet [17])**	**TL-FT (VGG16-ImageNet [17])**	**SSL (DeepSAR-Net-SGAN)**
7875 (whole dataset)	0.91	0.99	0.99	**0.93**
4000	0.89	0.98	0.99	**0.91**
2000	0.87	0.97	0.99	**0.90**
500	0.85	0.96	0.97	**0.89**
300	0.80	0.95	0.96	**0.86**
100	0.74	0.92	0.95	**0.86**
70	0.73	0.89	0.94	**0.83**
40	0.71	0.84	0.89	**0.80**
10	0.42	0.76	0.78	**0.75**

Table 5.2: IB F_1 score at different labeled samples sizes

As visible from Figure 5.9 and Table 5.2, the IB F_1 score metric results are lower in absolute value than the corresponding OA results. As previously stated, IB F_1 score metric may be considered the main cause of OA loss since the IB detection is the most difficult task in the problem under investigation. Anyway, the trend is very similar to the OA behaviour: the SL approach is always outperformed by the other three methods at hand.
In particular, the TL approach is very good at handling the IB detection even with fewer labeled data than the SSL, due to its ability to detect small features such as edges gained from the ImageNet challenge problem.

TenGeoP-SARwv training times and CO_2 consumption

Models / #labels	**SL (DeepSAR-Net) [16] (s)**	**TL (VGG16-ImageNet [17]) (s)**	**TL-FT (VGG16-ImageNet [17]) (s)**	**SSL (DeepSAR-Net-SGAN) (s)**
7875 (whole dataset)	284.66	1961.20	17040.50	1989.58

4000	174.10	1211.30	9157.20	1968.43
2000	99.25	806.21	4830.50	2017.25
500	54.16	575.62	3082.80	2135.67
300	49.83	531.48	2587.20	2361.40
100	43.21	482.63	2007.60	2578.56
70	38.63	447.81	1843.80	2212.63
40	35.02	412.62	1578.60	2267.18
10	24.75	362.64	1327.20	2342.35

Table 5.3: Training times comparison

Table 5.3 aims at showing the time necessary for training each of the different models under assessment. The results are considerably different from one approach to another due to some of the implementation choices already introduced in section **5.2.2**. In particular:

- the SL approach uses a Stochastic Gradient Descent optimizer with $lr = 0.01$, resulting in a relatively quick training phase although it has 4.16M parameters to train;
- the TL approach uses a RMSProp optimizer with $lr = 0.001$ for its standard version and a $lr = 0.0001$ for its FT version, resulting in a long computation even if the parameters to train are not too many (0,26M for the TL and 2,62M for the TL-FT approaches respectively). Moreover, it must be considered that every TL approach consists of pre-training the network on another problem before transferring the acquired knowledge: this could take days of training. In the case of VGG16, training this network on the ImageNet challenge leveraging on a Tesla P100 16GB PCIe GPU takes 277 ms / batch with batch size=64, as reported in [99]. This means that in this work fully training such a network would have taken approximately 89 hours of computation.
- the SSL approach must train a 7,02M parameters Generator and a 4,16M parameters Discriminator, using an Adam optimizer with $lr = 0.02$ and $lr = 0.002$ respectively. This results in a long computation, also due to the adversarial behaviour of the two NNs. The SSL times that are shown in Table 5.3 represent the duration needed (on average 10-15 epochs) for the Discriminator to achieve the best classification performance with a batch size of 16. Different batch sizes can be tested

out, varying the training time. Bigger batch sizes reduce training time but may let the Discriminator outperform the Generator too early compromising its learning procedure.
It is noticeable that on average with more labeled data the system receives in input, the sooner it achieves the best classification performance thanks to the boost received by the labeled data.

Batch sizes also impact training times: small batch sizes have been used when few labeled data were available, reaching a maximum size of 100 starting from the 300 labeled samples configuration.
As already anticipated above, the SSL batch size deserves a separate discussion, since it has been noticed the Generator and Discriminator behaviours usually relate to its choice. Smaller batch sizes help the GAN convergence and allow the Generator to generate good synthetic images after about 10 epochs, but this not always leads to an improvement of the classification results. Instead, bigger batch sizes allow the Discriminator to outperform the Generator from the very beginning of the training thanks to the bigger amount of real samples seen right from the start, resulting in high-quality classification results from the early stages. A good batch size trade-off can be 16, according to the conducted experiments.
Given the training times shown in Table 5.3 and the information of the GPU and ML platform that have been used (section **5.2**), it would be also possible to estimate the CO_2 emissions of the models under evaluation thanks to the work of Lacoste et al. [100].
For this work, experiments were conducted using the Google Colab platform (provided by Google Cloud) and in the Europe-West1 computation region, resulting in carbon efficiency of 0.27 $kgCO_2eq/kWh$. A cumulative of 0.65 hours of computation was performed to train the SSL architecture with the 10 labels configuration on hardware of type Tesla P100, characterised by a TDP of 250W. This results in an estimation of total emissions produced equivalent to 0.05 $kgCO_2eq$.
Such computations can be repeated for every approach in each of the configurations, including also the pre-training time in case of the TL approach. For each architecture, its CO_2 consumption would be estimated as a quantity that is directly proportional to its training time. Thus the TL approach would result as the less CO_2 consumption efficient method overall due to its computational-heavy pre-training phase.

SSL (DeepSAR-Net-SGAN) OA varying unlabeled samples size at fixed labeled samples sizes

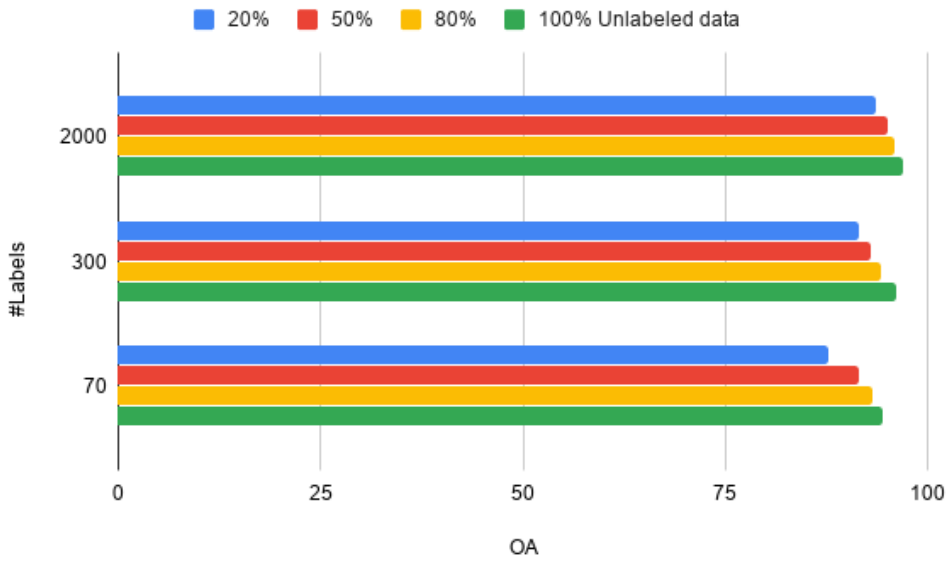

Figure 5.10: SSL (DeepSAR-Net-SGAN) OA results varying unlabeled samples size

%Unlabeled data \ #Labels	**70**	**300**	**2000**
100% (whole dataset)	94.37	96.13	96.89
80%	93.06	94.10	95.87
50%	91.51	92.86	94.90
20%	87.60	91.44	93.56

Table 5.4: SSL (DeepSAR-Net-SGAN) OA (%) results varying unlabeled samples size at fixed labeled samples sizes

Figure 5.10 and Table 5.4 show the variation of the OA performance obtained by the SSL approach varying the size of the unlabeled dataset given in input at fixed sizes of labeled data chosen as representatives. As visible, from 20% to 100% dataset size there is a consistent performance difference for almost every fixed labeled samples size configuration, with the gap that slowly shrinks when more labeled data are given to the models.
This leads to the deduction that more unlabeled data there are at disposal, the better performance results are reachable by the SSL approach, especially

with few labeled data.

5.3.2 TenGeoP-SARwv qualitative analysis

This section provides a qualitative analysis of the DeepSAR-Net-SGAN model achievements on the TenGeoP-SARwv dataset.

Class Activation Maps: SL (DeepSAR-Net) [16] vs SSL (DeepSAR-Net-SGAN) with 10 labeled samples

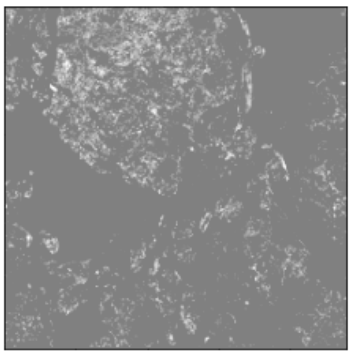

Figure 5.11: A sample of SI class drawn from TenGeoP-SARwv

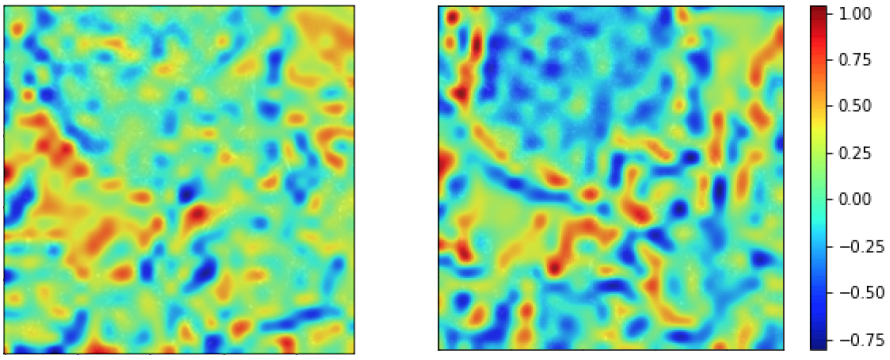

Figure 5.12: SL (DeepSAR-Net) [16] vs SSL (DeepSAR-Net-SGAN) Class Activation Maps of class SI with 10 labeled samples

Given the original image in Figure 5.11, Figure 5.12 shows the comparison of its Class Activation Maps produced by the SL (on the left) and the SSL (on

the right) approaches, which indicate the image regions that the CNN activates along the NN to discriminate each category under assessment. Here, activations of the SI class image after the third DeepSAR-Net Convolutional layer have been chosen for the comparison of the two approaches.
Thanks to the unlabeled data provided, the SSL approach is way more capable of activating the correct pixels in the input image and then classify it. In the example, both the models produce a correct classification of SI, but the SSL model outputs a probability for the correct class that is higher (0.99976) than the SL one (0.93425).

Synthetic images generated by the SSL (DeepSAR-Net-SGAN) approach

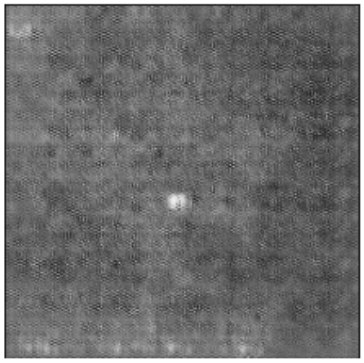

Figure 5.13: Samples of IB class generated by the DeepSAR-Net-SGAN

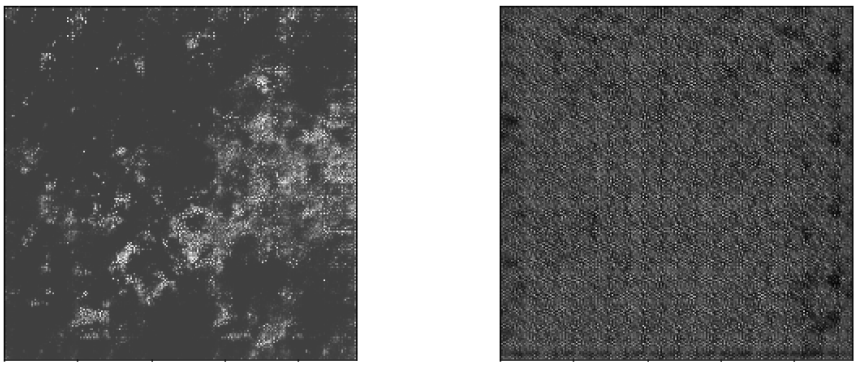

Figure 5.14: Two samples of SI and POW classes generated by the DeepSAR-Net-SGAN

Figure 5.13 and 5.14 depict some images generated by the DeepSAR-Net-SGAN Generator after about 15 epochs of training, proving that not only

the DeepSAR-Net-SGAN can be successfully used to train a supervised Discriminator for classification purposes, but it can be also exploited to perform Data Augmentation or Label Refinery techniques once high-quality images are generated.
There is still room for further improvement in this sense and problems such as Mode Collapse have to be faced appropriately. At the current state, Mode Collapse is not treated and as a consequence each Generator that is saved during the training phase turns out to be very good at reproducing images similar to the ones it has seen in input shortly before, but not those that has seen earlier.

5.3.3 TenGeoP-SARwv special configurations

This section shows the impact of the additional information given in input to the models (geographic coordinates for the TenGeoP-SARwv use case), as well as the SVM classifier impact. Among the assessed models, the SL (DeepSAR-Net) [16] approach is the architecture which experiences the biggest improvements.

Geo-based SL vs base SL (DeepSAR-Net) [16]

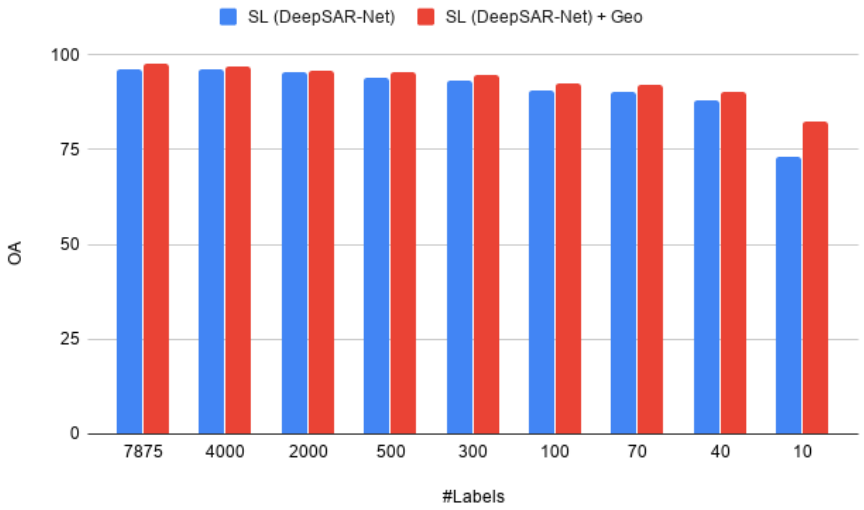

Figure 5.15: OA comparison between Geo-based SL and base SL (DeepSAR-Net) [16]

As visible from Figure 5.15, concatenating the geographic coordinates with the CNN features boosts the SL performance and achieves higher quality

results also when few labeled data are available. This highlights the correlation existing between the distribution of the geographic coordinates (shown in Figure 5.3) and the distribution of the classes, given the different world map positions of the targets to be categorized. The concatenation gives a relevant contribution mainly in the SL approach, allowing this technique to narrow the gap with the other approaches. Still, it is not able to reach TL and SSL performance. The TL and SSL already have additional information to deal with, therefore the geographic coordinates do not seem to affect their performance in a relevant way.

SVM-based SL vs base SL (DeepSAR-Net) [16]

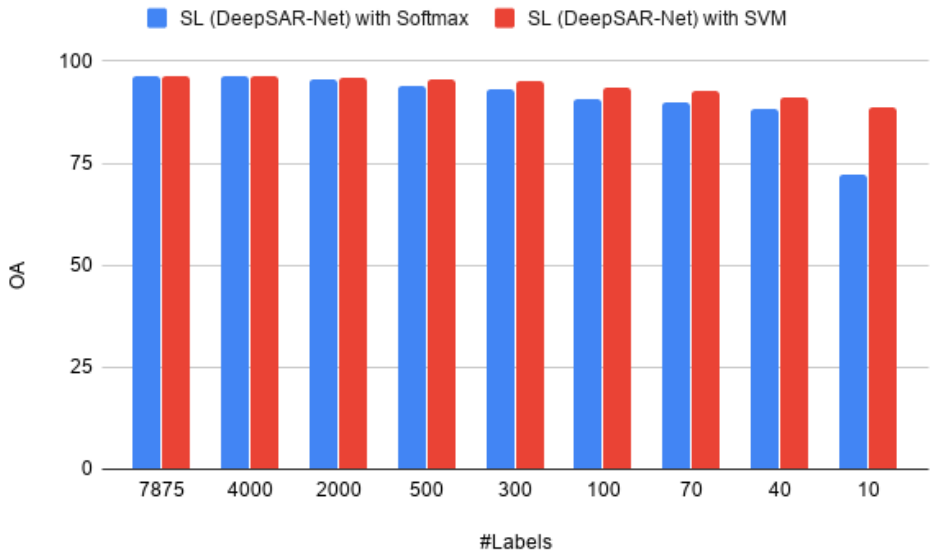

Figure 5.16: OA comparison between SVM-based SL and base SL (DeepSAR-Net) [16]

As shown in Figure 5.16, substituting the Softmax classifier with the SVM classifier allows the traditional SL approach to achieve high-quality results even with few labeled data used. This is mainly due to the capability of the SVM of generalizing with datasets of small size: it is a kernel-based method and as such works better with few data with a lot of features than parametric methods such as the Logistic Regression. The improvements in the inclusion of the SVM are not that effective in the other architectures, which already achieve high performance with the Logistic Regression. Therefore, the SVM substitution allows the SL approach to narrow the gap with the other approaches, but still obtaining lower performance than TL and SSL.

5.3.4 C-CORE quantitative analysis

Here follow the experiments and the related performance metrics results chosen to address the C-CORE classification problem. A quantitative analysis of the results is provided for each experiment.

Overall Accuracy

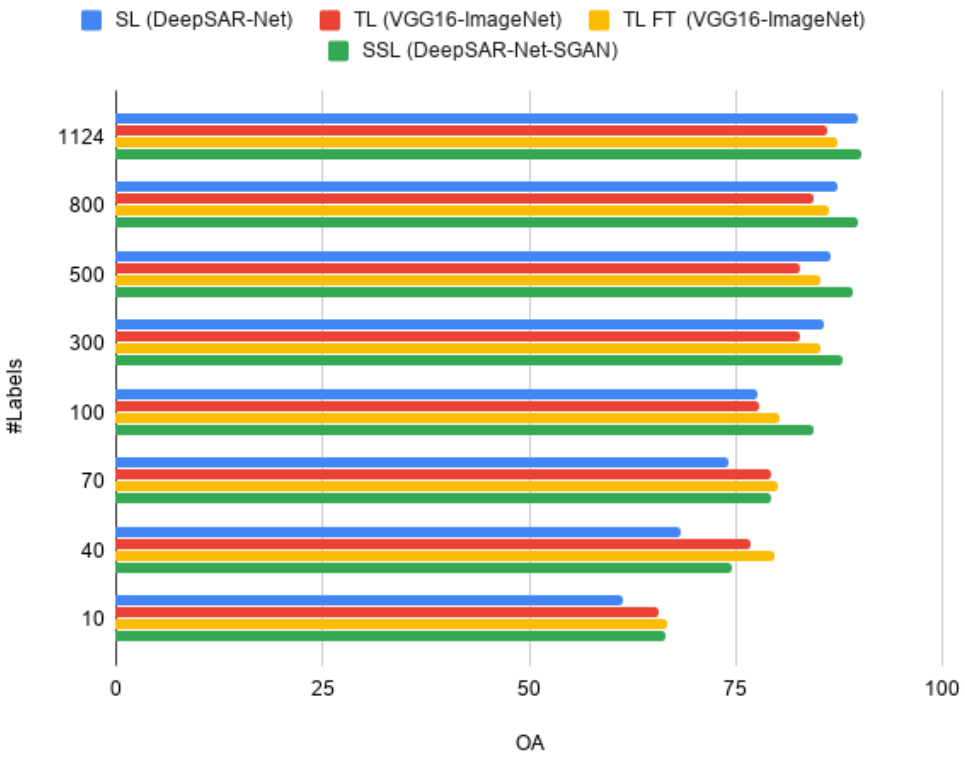

Figure 5.17: OA score at different labeled samples sizes

Models / #labels	**SL (DeepSAR-Net) [16] (%)**	**TL (VGG16-ImageNet [17]) (%)**	**TL-FT (VGG16-ImageNet [17]) (%)**	**SSL (DeepSAR-Net-SGAN) (%)**

1124 (whole dataset)	89.67	86.10	87.33	**90.15**
800	87.33	84.33	86.19	**89.74**
500	86.45	82.67	85.30	**89.11**
300	85.63	81.60	84.67	**87.96**
100	77.64	77.80	80.38	**84.45**
70	74.10	79.29	80.17	**79.33**
40	68.24	76.67	79.70	**74.55**
10	61.34	65.66	66.78	**66.50**

Table 5.5: OA score at different labeled samples sizes

Figure 5.17 and Table 5.5 show the OA results obtained by the models under assessment varying the size of the labeled data given in input to them at training time. For each configuration, the SSL approach exploits not only the restricted pool of selected labeled data, but also the whole dataset removing its labels and treating it as unlabeled data.
As visible in the last stages (very few labeled data available), the TL and the SSL approaches outperform the traditional SL approach. In the early stages, instead, the SL model is almost as accurate as the SSL approach and does even better than the TL approach thanks to the bigger amount of labeled information given to the models.
The TL approach is effective in every configuration, but not as much as it was the previous use case (section **5.3.1**). This is due to the fact that the C-CORE pixel values represent the signals produced by SAR backscatter and are measured in dB, differing a lot from the ImageNet images. Moreover, its FT version improves as more labeled information is given to the model since it is tailored to the higher-level features of the specific use case, which boosts its performance.
The SSL approach is more effective than the SL approach with few labeled data. As more information is given, it improves and outperforms the TL results, while the gap with the SL approach is reduced due to the high amount of labeled data given during the training phase.
The results are computed on the test set and no cross-validation is performed due to the small size of the dataset at hand.
The overall performance obtained in this use case are on average lower than the ones obtained in (section **5.3.1**), due to the higher difficulty of the problem. Pictures shown in the next subsection, related to the Class Activation

Maps, clarify how difficult is to distinguish a Ship from an Iceberg in 2D images.

C-CORE training times and CO_2 consumption

Models / #labels	**SL (DeepSAR-Net) [16] (s)**	**TL (VGG16-ImageNet [17]) (s)**	**TL-FT (VGG16-ImageNet [17]) (s)**	**SSL (DeepSAR-Net-SGAN) (s)**
1124 (whole dataset)	54.16	66.39	81.22	63.12
800	40.79	51.38	60.19	58.88
500	28.60	38.62	45.61	87.17
300	17.32	19.53	34.88	70.20
100	13.45	21.33	25.27	65.36
70	11.89	19.21	23.60	80.66
40	10.21	17.39	21.41	75.98
10	8.86	14.43	17.23	88.35

Table 5.6: Training times comparison

Table 5.6 aims at showing the time necessary for training each of the different models under assessment. The results are considerably different from one approach to another due to some of the implementation choices already introduced in section **5.2.2**. In particular:

- the SL approach uses a Stochastic Gradient Descent optimizer with $lr = 0.01$, resulting in a relatively quick training phase although it has 4.16M parameters to train;
- the TL approach uses a RMSProp optimizer with $lr = 0.01$ for both its standard version and FT version, resulting in a similar computation time compared to SL even if the parameters to train are less (0,26M for the TL and 2,62M for the TL-FT approaches respectively). Moreover, it must be considered that every TL approach consists of pre-training the network on another problem before transferring the acquired knowledge: this could take days of training. In the case of

VGG16, training this network on the ImageNet challenge leveraging on a Tesla P100 16GB PCIe GPU takes 277 ms / batch with batch size=64, as reported in [99]. This means that in this work fully training such a network would have taken approximately 89 hours of computation.

- the SSL approach must train a 4,42M parameters Generator and a 4,16M parameters Discriminator, using an Adam optimizer with $lr = 0.002$. This results in a longer computation than SL, also due to the adversarial behaviour of the two NNs. The SSL times that are shown in Table 5.3 represent the duration needed (on average 10-15 epochs) for the Discriminator to achieve the best classification performance with a batch size of 16. Different batch sizes can be tested out, varying the training time. Bigger batch sizes reduce training time but may let the Discriminator outperform the Generator too early compromising its learning procedure.
  It is noticeable that on average with more labeled data the system receives in input, the sooner it achieves the best classification performance thanks to the boost received by the labeled data. Still, the training times are quite unstable due to the adversarial training between Generator and Discriminator.

Batch sizes also impact training times: small batch sizes have been used when few labeled data were available, reaching a maximum size of 100 starting from the 300 labeled samples configuration.
Given the training times shown in Table 5.3 and the information of the GPU and ML platform that have been used (section **5.2**), it would be also possible to estimate the CO_2 emissions of the models under evaluation thanks to the work of Lacoste et al. [100], as already proposed in **5.3.1**.
In this case, given the similar training times needed by the assessed models, their related CO_2 emissions estimates are also comparable. In any case, in such comparison it must be also taken into account the TL pre-training phase, that is computationally-heavy and hence results in much higher CO_2 consumption than the other assessed approaches.

5.3.5 C-CORE qualitative analysis

This section provides a qualitative analysis of the DeepSAR-Net-SGAN model achievements on the C-CORE dataset.

Class Activation Maps: Ice vs Ship with SSL (DeepSAR-Net-SGAN)

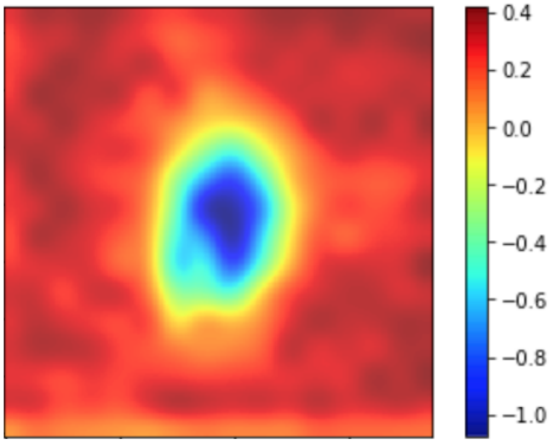

Figure 5.18: SSL (DeepSAR-Net-SGAN) Class Activation Maps of class Iceberg with 10 labeled samples

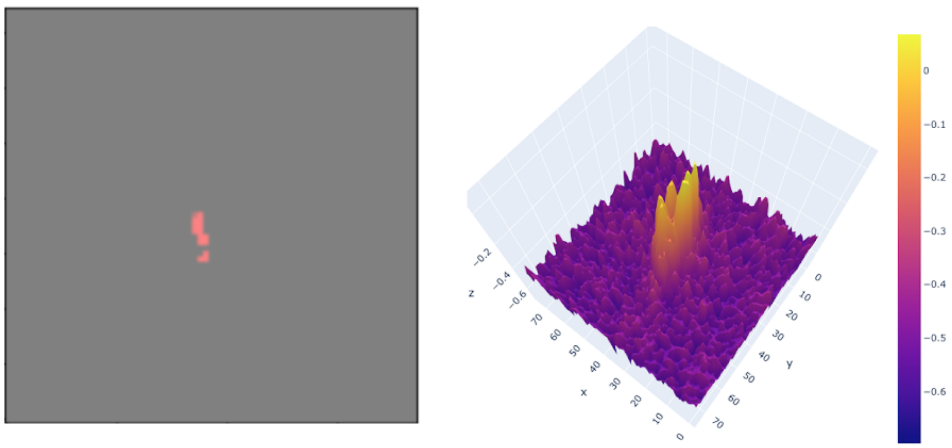

Figure 5.19: 2D and 3D representations of a sample of class Iceberg

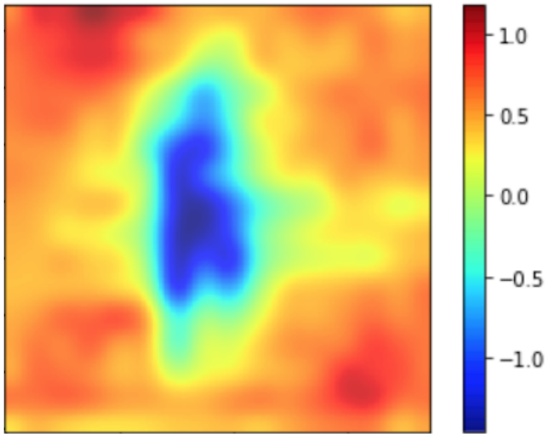

Figure 5.20: SSL (DeepSAR-Net-SGAN) Class Activation Maps of class Ship with 10 labeled samples

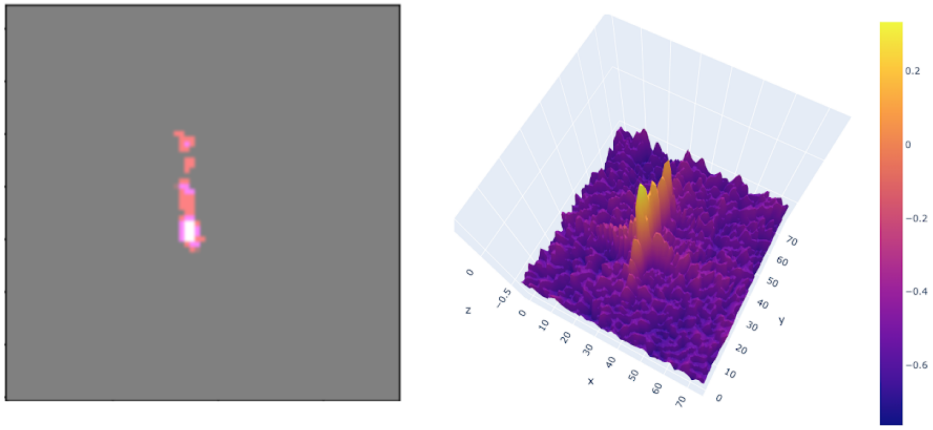

Figure 5.21: 2D and 3D representations of a sample of class Ship

Figure 5.18, 5.19, 5.20 and 5.21 show the comparison of the SSL Class Activation Maps of the two classes under assessment Ship and Iceberg. In particular, the activations indicate the image regions that the CNN activates along the NN to discriminate each category under assessment. Here, activations of the Ship and Iceberg images after the third DeepSAR-Net Convolutional layer have been chosen for the comparison of the two Class Activation Maps.
In the examples, it is noticeable that the Ship activation is more regular than the Iceberg activation, given its nearly rectangular shape. To understand if activations are correctly computed, 3D images are provided (in the cen-

tre). These images show more effectively the backscatter difference between a Ship and an Iceberg, whose reflectivity properties are different. Hence 3D images are better suited for human visual inspection than the 2D images (on the left), whose content is difficult to distinguish.

5.3.6 C-CORE special configurations

This section shows the impact of the additional information given in input to the models (incidence angles for the C-CORE use case), as well as the SVM classifier impact. Among the assessed models, the SL (DeepSAR-Net) [16] approach is the architecture which experiences the biggest improvements.

Incidence angle-based SL vs base SL (DeepSAR-Net) [16]

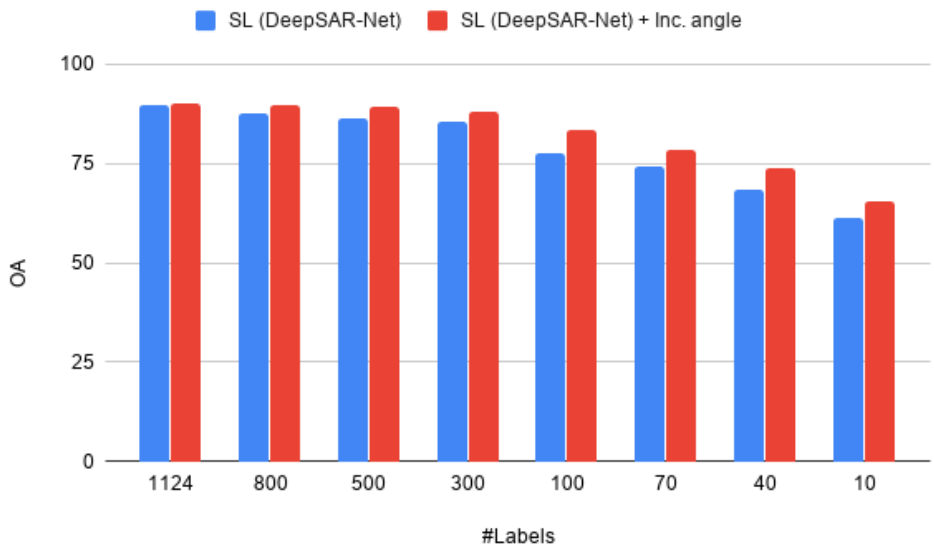

Figure 5.22: OA comparison between Inc. angle-based SL and base SL (DeepSAR-Net) [16]

As visible from Figure 5.15 and similarly to what happened with the previous dataset (section **5.3.3**), concatenating additional information such as incidence angle with the CNN features boosts the SL performance and achieves higher quality results also when few labeled data are available. This highlights the correlation existing between the distribution of the incidence angles and the distribution of the classes, given the different reflectivity properties of the targets to be categorized. The concatenation gives a relevant

contribution mainly in the SL approach, allowing this technique to narrow the gap with the other approaches. The TL and SSL already have additional information to deal with, therefore the incidence angle does not seem to affect their performance in a relevant way.

SVM-based SL vs base SL (DeepSAR-Net) [16]

As shown in Figure 5.16 and as already experienced with the previous dataset (section **5.3.3**), substituting the Softmax classifier with the SVM classifier allows the traditional SL approach to achieve high-quality results even with few labeled data used. This is mainly due to the capability of the SVM of generalizing with datasets of small size: it is a kernel-based method and as such works better with few data with a lot of features than parametric methods such as the Logistic Regression. The improvements in the inclusion of the SVM are not that effective in the other architectures, which already achieve high performance with the Logistic Regression.

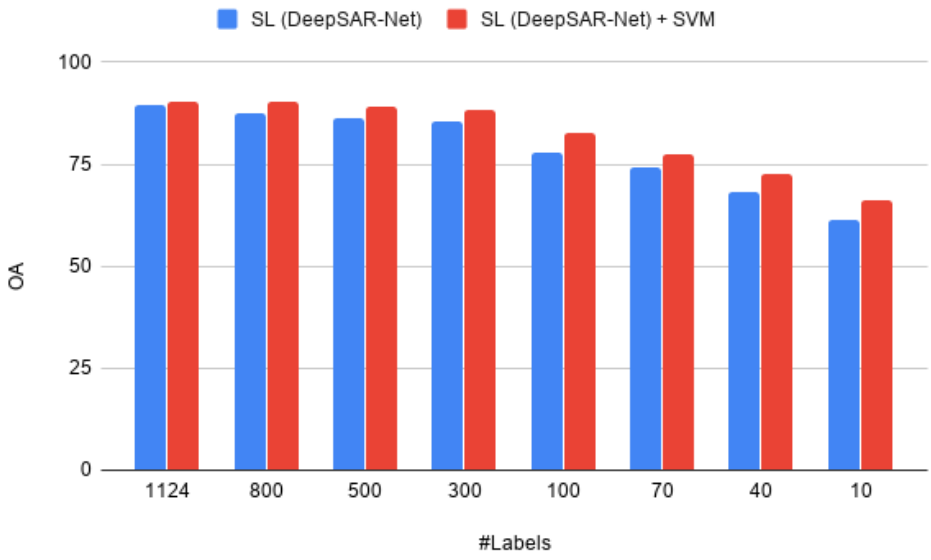

Figure 5.23: OA comparison between SVM-based SL and base SL (DeepSAR-Net) [16]

5.4 Discussion

The experiments that have been conducted show that the DeepSAR-Net-SGAN architecture for the SSL approach is capable of exploiting the knowledge hidden in unlabeled data outperforming the traditional SL approach and thus representing a great resource for both the EO and SAR domains. The performance results reached by the assessed methods on the two datasets and presented in sections **5.3.1** and **5.3.4** are consistent with each other and in line with the original hypothesis that underlies this research work, guaranteeing the robustness of the obtained results.
Indeed, the SSL approach outperforms the traditional SL approach in every assessed setting, even in special configurations when additional information such as geographic coordinates of the images is included, representing a valid Representation Learning alternative to the TL approach based on pre-trained NNs.
In particular, SSL improves the OA achieved by the SL approach by at least 5% in configurations with less than 100 labeled samples available at training time in both the use cases, as visible from Table 5.7 for the TenGeoP-SARwv use case and from Table 5.8 for the C-CORE use case.

Models / #labels	**SL (DeepSAR-Net) [16] (%)**	**TL (VGG16-ImageNet [17]) (%)**	**TL-FT (VGG16-ImageNet [17]) (%)**	**SSL (DeepSAR-Net-SGAN) (%)**
100	90.30	97.53	98.43	**94.98**
70	89.77	96.75	98.14	**94.37**
40	88.17	95.87	96.65	**93.41**
10	72.40	94.47	94.81	**91.94**

Table 5.7: OA comparison at different labeled samples sizes on TenGeoP-SARwv use case, extracted from Table 5.1

Models / #labels	**SL (DeepSAR-Net) [16] (%)**	**TL (VGG16-ImageNet [17]) (%)**	**TL-FT (VGG16-ImageNet [17]) (%)**	**SSL (DeepSAR-Net-SGAN) (%)**
100	77.64	77.80	80.38	**84.45**

70	74.10	79.29	80.17	**79.33**
40	68.24	76.67	79.70	**74.55**
10	61.34	65.66	66.78	**66.50**

Table 5.8: OA score comparison at different labeled samples sizes on C-CORE use case, extracted from Table 5.5

These results show that the OA gap between SSL and SL grows as less labeled data are given to the models at training time, while SSL and TL achieve pretty similar results in both the use cases. In the first one, TL achieves higher performance thanks to a very positive knowledge transfer from the source domain (ImageNet [82]) to the target domain, while in the second case the transfer is less effective due to the fact that C-CORE pixel values represent the signals produced by SAR backscatter measured in dB and are thus different from the ImageNet images.
Still, SSL in both the use cases has a great advantage compared to TL: it does not need any pre-training procedure. This avoids long training times and consequent large CO_2 emissions produced by the computation engine, as discussed in **5.3.1** and **5.3.4**. Furthermore, with SSL it is not needed to find a wide labeled dataset on which the pre-training phase must be performed, saving manpower: SSL leverages unlabeled data to boost the performance of the supervised classification, therefore there is no need to hand-label a huge amount of data except those few that are necessary.
Specifically, it has been shown in section **5.3.1** that more unlabeled data are available, the higher is the performance achieved by the SGAN, especially with few labeled data. This means that in a domain as EO it could be enough to have a pool of few labeled data available and a large amount of unlabeled data to obtain high classification performance.
At last, further evidence of the validity of the results and the effectiveness of the SSL approach has been given in sections **5.3.2** and **5.3.5**, through the use of the Class Activation Maps along the NNs architectures and through the generation of synthetic images produced by the the DeepSAR-Net-SGAN Generator approach. The generated images prove the quality of the approach and opens up to various possibilities for future work, as delved in the next chapter.

Chapter 6

Conclusions and future work

This chapter concludes the thesis by summarizing the work done and advising the possible future work.

6.1 Conclusions

This thesis has addressed the problem of performing Sea-Ice classification on polar SAR satellite images. Given the scarce availability of labeled data in the EO and satellite domain, addressing such a task with the traditional SL approach has revealed to be unfavourable.
Prompted by the huge availability of EO unlabeled data, this work has explored a SSL solution based on GANs (SGAN [14]) to address the problem at hand, benchmarking its performance with a SL approach that shared its same CNN architecture (DeepSAR-Net [16]) and a TL approach based on VGG16 [17] pre-trained on the ImageNet challenge.
The comparison has been performed taking into account widely used classification metrics such as OA and F_1 score, but also other parameters such as training times and related CO_2 emissions.
The benchmarking procedure has revealed the effectiveness of the SSL approach based on GANs, showing its ability to extract knowledge from unlabeled data and exploit it outperforming the traditional SL approach. In particular, it improves the OA achieved by the SL approach by at least 5% in configurations with less than 100 labeled samples available at training time in both the use cases under evaluation. Moreover, the SSL approach obtains performance comparable to the TL one, with the great advantage of avoiding the TL pre-training step, hence saving big human effort for data hand-labeling, long training time and great CO_2 emission.
Besides, such an architecture is able not only to perform an accurate classifi-

cation through its modified Discriminator but also to produce good quality synthetic images through its Generator, which can be used to implement techniques such as Data Augmentation and Label Refinery with the aim of enhancing the classification performance even further. This capability opens up to the outlining of the possible future work, discussed in the next section.
At last, some variations of the above-mentioned architectures have been tested out, concatenating additional information such as geographic coordinates to the CNN features extracted from the images or substituting the final softmax classifier with an SVM. These variations resulted in relevant performance enhancement especially for the SL approach, therefore able to narrow the gap with SSL and TL approaches.
To sum up, the main contributions provided by this work are the following:

- The extensive study and comparison of existing DL approaches in the EO domain have been performed, with the focus on the Sea-Ice classification problem with scarcely available labeled data;
- The extension of the SGAN architecture [14] based on the DeepSAR-Net [16] has been introduced. It outperforms the traditional DeepSAR-Net [16] SL approach by at least 5% in configurations with less than 100 labeled samples available at training time in both the use cases under evaluation, and competes with the TL approach based on VGG16 [17];
- The comparison of the training times and CO_2 emissions related to the SL, SSL and TL approaches has been proposed, pointing out the great advantage of the SSL approach of avoiding the TL pre-training step, that is computationally-heavy;
- The generative capabilities of the DeepSAR-Net-SGAN approach have been highlighted, opening up new Data Augmentation and Label Refinery possibilities;
- The performance enhancement related to DeepSAR-Net-SGAN when increasing the size of unlabeled data available has been shown;
- The SVM classifier capability of generalizing with datasets of small size has been proved, enhancing the SL approach performance especially when few labeled data are available;
- The performance enhancement in the traditional SL approach due to additional information given (e.g. geographic coordinates) has been highlighted, especially when few labeled data are available.

6.2 Future work

The future work may be dedicated to giving a deeper emphasis on a few points that can provide an interesting advancement of this research, such as:

- Study and evaluate the deployment of the model on AI modules-equipped satellites, so that once images get analyzed just geographic coordinates of detected icebergs and ice caps could be passed to the ground stations to notify their existence, saving bandwidth;
- Refine the quality of synthetic images generated by the SGAN Generator through architectures modifications (e.g. treating Mode Collapse with the inclusion of a replay buffer) and implement a Label Refinery technique to perform a Data Augmentation of the existing labeled dataset;
- Compare the SGAN approach with another Semi-Supervised Learning approach (e.g. Variational Autoencoder), evaluating the differences and similarities between the approaches to delve more into the possible SSL solutions to the problem of scarce availability of labeled data;
- Evaluate the use of an alternative Representation Learning approach based on UL methods such as the Deep Clustering technique for exploiting unlabeled data before the final Supervised classification;
- Test the generality of the developed models in different use cases such as the Food Security TEP, with the aim of performing tasks such as crop classification and crop yield estimation, or continuously monitoring the water and nutrient availability in crops development.

Bibliography

[1] Aurélien Géron. *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow - 2nd Edition*. O'Reilly, 2019.

[2] European Commission and European Space Agency. *Copernicus Marine Environment Monitoring Service*. 2015. URL: http://marine.copernicus.eu/.

[3] UoA et al. *ExtremeEarth*. 2019. URL: http://earthanalytics.eu/index.html.

[4] Ramon Torres et al. "GMES Sentinel-1 mission". In: *Remote Sensing of Environment* 120 (2012), pp. 9–24. ISSN: 0034-4257. DOI: https://doi.org/10.1016/j.rse.2011.05.028. URL: http://www.sciencedirect.com/science/article/pii/S0034425712000600.

[5] Olivier Chapelle, Bernhard Schölkopf, and Alexander Zien. *Semi-Supervised Learning*. London, U.K.: MIT Press, 2006.

[6] Xiaojin Zhu. *Semi-Supervised Learning Literature Survey*. Tech. rep. University of Wisconsin – Madison, 2008.

[7] Lisa Torrey and Jude Shavlik. *Transfer learning*. Handbook of Research on Machine Learning Applications, Trends: Algorithms, Methods, and Techniques: IGI Global, 2009.

[8] Vincent Vanhoucke. *The Quiet Semi-Supervised Revolution*. 2019. URL: https://towardsdatascience.com/the-quiet-semi-supervised-revolution-edec1e9ad8c.

[9] Tatiana Tommasi, Francesco Orabona, and Barbara Caputo. "Learning Categories From Few Examples With Multi Model Knowledge Transfer". In: *IEEE transactions on pattern analysis and machine intelligence* 36 (Oct. 2013). DOI: 10.1109/TPAMI.2013.197.

[10] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016.

[11] Yoshua Bengio. *Learning Deep Architectures for AI*. Tech. rep. Dept. IRO, Universite de Montreal, 2009.

[12] Julien Despois. *Memorizing is not learning! — 6 tricks to prevent overfitting in machine learning*. 2018. URL: https://hackernoon.com/memorizing-is-not-learning-6-tricks-to-prevent-overfitting-in-machine-learning-820b091dc42.

[13] Anne Håkansson. "Portal of Research Methods and Methodologies for Research Projects and Degree Projects". In: (2013).

[14] Tim Salimans et al. *Improved Techniques for Training GANs*. 2016. eprint: 1606.03498. URL: http://papers.nips.cc/paper/6125-improved-techniques-for-training-gans.pdf.

[15] Jos van de Wolfshaar. *Semi-supervised learning with GANs*. 2018. URL: https://medium.com/@jos.vandewolfshaar/semi-supervised-learning-with-gans-23255865d0a4.

[16] Yang Li et al. "DeepSAR-Net: Deep convolutional neural networks for SAR target recognition". In: (Mar. 2017), pp. 740–743. DOI: 10.1109/ICBDA.2017.8078734.

[17] Karen Simonyan and Andrew Zisserman. "Very Deep Convolutional Networks for Large-Scale Image Recognition". In: *arXiv 1409.1556* (Sept. 2014).

[18] Beauchamp Tom L. and Childress James F. *Principles of biomedical ethics*. Oxford University Press New York, 1979.

[19] United Nations. *Sustainable Development Goals*. 2015. URL: https://www.un.org/sustainabledevelopment/sustainable-development-goals/.

[20] Anthony J. Lewis Floyd M. Henderson. *Principles and Applications of Imaging Radar. Manual of Remote Sensing*. John Wiley and Sons, New York, 1998.

[21] Yee Kit Chan and Voon-Chet Koo. "An introduction to Synthetic Aperture Radar (SAR)". In: (2008). DOI: 10.2528/PIERB07110101.

[22] Mark A. Richards. *Fundamentals of Radar Signal Processing*. McGraw-Hill Professional, 2005. ISBN: 0071444742. DOI: 10.1036/0071444742. URL: https://mhebooklibrary.com/doi/book/10.1036/0071444742.

[23] *Radar Basics*. URL: https://training.weather.gov/nwstc/NEXRAD/RADAR/Section1-2.html.

[24] Alan Di Cenzo. "A new look at nonseparable synthetic aperture radar processing". In: *IEEE Transactions on Aerospace and Electronic Systems* 24.3 (May 1988), pp. 218–223. DOI: 10.1109/7.192089.

[25] Ian G. Cumming and Frank H. Wong. *Digital Processing of Synthetic Aperture Radar Data*. 2004.

[26] Robert N. McDonough John C. Curlander. *Synthetic Aperture Radar. Systems and Signal Processing*. 1991.

[27] Riccardo Lanari and Giorgio Franceschetti. *Synthetic Aperture Radar Processing*. 1999.

[28] *Acquisition Modes*. URL: https://sentinel.esa.int/web/sentinel/user-guides/sentinel-1-sar/acquisition-modes.

[29] *Level-1 SLC Processing Algorithms*. URL: https://sentinel.esa.int/web/sentinel/level-1-slc-processing-algorithms.

[30] *Polarimetric observation by PALSAR*. URL: https://www.eorc.jaxa.jp/ALOS/en/img_up/pal_polarization.htm.

[31] Sotiris Kotsiantis. "Supervised Machine Learning: A Review of Classification Techniques". In: (2007), pp. 3–24.

[32] Corinna Cortes and Vladimir Vapnik. *Support-Vector Networks*. 1995, pp. 273–297.

[33] Anil Cheriyadat. "Unsupervised Feature Learning for Aerial Scene Classification". In: *IEEE Transactions on Geoscience and Remote Sensing* 52.1 (Jan. 2014), pp. 439–451. ISSN: 1558-0644. DOI: 10.1109/TGRS.2013.2241444.

[34] Trevor J. Hastie and Robert Tibshirani. "Classification by Pairwise Coupling". In: (1997). DOI: DOI:10.1214/aos/1028144844.

[35] Bernhard Scholkopf. *Learning with Kernels - Support Vector Machines, Regularization, Optimization and Beyond. MIT Press, Cambridge, MA*. Vol. 98. Jan. 2001.

[36] *The Radial Basis Function Kernel*. URL: http://pages.cs.wisc.edu/~matthewb/pages/notes/pdf/svms/RBFKernel.pdf.

[37] *Support Vector Machines*. URL: https://scikit-learn.org/stable/modules/svm.html.

[38] Sarang Narkhede. *Understanding Confusion Matrix*. 2018. URL: https://towardsdatascience.com/understanding-confusion-matrix-a9ad42dcfd62.

[39] Tom M. Mitchell. *Machine Learning*. McGraw-Hill, 1997.

[40] Warren S. McCulloch and Walter H. Pitts. “A logical calculus of the ideas immanent in nervous activity”. In: (1943).

[41] Frank Rosenblatt. “The Perceptron. A perceiving and recognizing automation.” In: (1957).

[42] Donald Olding Hebb. *The Organization of Behavior. A neuropsychologycal theory.* John Wiley & Sons, 1949.

[43] Marvin Minsky and Seymour Papert. *Perceptrons: An Introduction to Computational Geometry.* Cambridge, MA, USA: MIT Press, 1969.

[44] *Neural Networks.* URL: https://miro.medium.com/max/1302/1*UA30b0mJUPYoPvN8yJr2iQ.jpeg.

[45] Vinod Nair and Geoffrey E. Hinton. “Rectified Linear Units Improve Restricted Boltzmann Machines”. In: (2010).

[46] Balázs Csanád Csáji. “Approximation with Artificial Neural Networks”. In: (2001). URL: https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.101.2647&rep=rep1&type=pdf.

[47] Amir H. Payberah. *Deep Feedforward Networks.* URL: https://id2223kth.github.io/slides/2019/06_feedforward_networks.pdf.

[48] Ning Qian. “On the momentum term in gradient descent learning algorithms”. In: *Neural Networks* 12.1 (1999), pp. 145–151. ISSN: 0893-6080. DOI: https://doi.org/10.1016/S0893-6080(98)00116-6. URL: http://www.sciencedirect.com/science/article/pii/S0893608098001166.

[49] Geoffrey Hinton and Nitish Srivastava and Kevin Swersky. *Neural Networks for Machine Learning.* URL: http://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf.

[50] Diederik P. Kingma and Jimmy Ba. *Adam: A Method for Stochastic Optimization.* 2014. arXiv: 1412.6980 [cs.LG].

[51] Daniel Svozil, Vladimir Kvasnicka, and Jiří Pospíchal. “Introduction to multi-layer feed-forward neural networks”. In: *Chemometrics and Intelligent Laboratory Systems* 39 (Nov. 1997), pp. 43–62. DOI: 10.1016/S0169-7439(97)00061-0.

[52] Danilo P. Mandic and Jonathon A. Chambers. “Recurrent Neural Networks for Prediction: Learning Algorithms, Architectures and Stability”. In: (2001).

[53] Sepp Hochreiter and Jürgen Schmidhuber. "Long short-term memory". In: *Neural computation* 9.8 (1997), pp. 1735–1780.

[54] Yann Lecun and Yoshua Bengio. "Convolutional Networks for Images, Speech, and Time-Series". In: *The Handbook of Brain Theory and Neural Networks* (Jan. 1995).

[55] Ian Goodfellow et al. "Generative Adversarial Nets". In: (2014). Ed. by Z. Ghahramani et al., pp. 2672–2680. URL: http://papers.nips.cc/paper/5423-generative-adversarial-nets.pdf.

[56] Geoffrey Everest Hinton and Ruslan Salakhutdinov. "Reducing the Dimensionality of Data with Neural Networks". In: *Science (New York, N.Y.)* 313 (Aug. 2006), pp. 504–7. DOI: 10.1126/science.1127647.

[57] Yi Tao Zhou and Rama Chellappa. "Computation of optical flow using a neural network". In: *IEEE 1988 International Conference on Neural Networks* (1988), 71–78 vol.2. DOI: 10.1109/ICNN.1988.23914.

[58] Connor Bowley et al. "Toward Using Citizen Scientists to Drive Automated Ecological Object Detection in Aerial Imagery". In: (Oct. 2017), pp. 99–108. DOI: 10.1109/eScience.2017.22.

[59] David Smith and Brian Burke. *Hype Cycle for Emerging Technologies 2019*. 2019. URL: https://www.gartner.com/en/documents/3956015/hype-cycle-for-emerging-technologies-2019.

[60] Yoshua Bengio, Aaron Courville, and Pascal Vincent. "Representation Learning: A Review and New Perspectives". In: (2012). DOI: 10.1109/TPAMI.2013.50.

[61] Jason Yosinski et al. "How transferable are features in deep neural networks?" In: NIPS'14 (), pp. 3320–3328.

[62] integrate.ai. *Transfer Learning Explained*. 2018. URL: https://medium.com/the-official-integrate-ai-blog/transfer-learning-explained-7d275c1e34e2.

[63] *Semi-Supervised Learning for NLP*. URL: https://helicqin.github.io/2018/03/22/Semi-Supervised%20Learning%20for%20NLP/.

[64] Abdelmalek Amine et al. *Computational Intelligence and Its Applications 6th IFIP TC 5 International Conference, CIIA 2018, Oran, Algeria, May 8-10, 2018, Proceedings*. Apr. 2018. ISBN: 978-3-319-89742-4.

[65] Fan Hu et al. “Transferring Deep Convolutional Neural Networks for the Scene Classification of High-Resolution Remote Sensing Imagery”. In: *Remote Sensing* (2015).

[66] Svetlana Lazebnik, Cordelia Schmid, and Jean Ponce. “Beyond Bags of Features: Spatial Pyramid Matching for Recognizing Natural Scene Categories”. In: (2006). URL: http://www.csd.uwo.ca/~olga/Courses/Fall2014/CS9840/Papers/lazebnikcvpr06b.pdf.

[67] Yi Yang and Shawn Newsam. “Spatial pyramid co-occurrence for image classification”. In: *Proceedings of the IEEE International Conference on Computer Vision* (2011). DOI: 10.1109/ICCV.2003.1238663.

[68] Lionel Gueguen. “Classifying Compound Structures in Satellite Images: A Compressed Representation for Fast Queries”. In: *Geoscience and Remote Sensing, IEEE Transactions on* 53 (Mar. 2015). DOI: 10.1109/TGRS.2014.2348864.

[69] Josef Sivic and Andrew Zisserman. “Video Google: A Text Retrieval Approach to Object Matching in Videos”. In: *Proceedings of the IEEE International Conference on Computer Vision* 2 (Nov. 2003), 1470–1477 vol.2. DOI: 10.1109/ICCV.2003.1238663.

[70] Natalia Zakhvatkina, Vladimir Smirnov, and Irina Bychkova. “Satellite SAR Data-based Sea Ice Classification: An Overview”. In: *Geosciences* (2019).

[71] Wolfgang Dierking. “Mapping of Different Sea Ice Regimes Using Images From Sentinel-1 and ALOS Synthetic Aperture Radar”. In: *IEEE Transactions on Geoscience and Remote Sensing* 48.3 (Mar. 2010), pp. 1045–1058. ISSN: 1558-0644. DOI: 10.1109/TGRS.2009.2031806.

[72] Wolfgang Dierking — Alfred Wegener Institute for Polar and Germany Marine Research Bremerhaven. “Sea Ice Monitoring by Synthetic Aperture Radar”. In: *Oceanography* 26 (June 2013). URL: https://doi.org/10.5670/oceanog.2013.33.

[73] Bernd Scheuchl et al. “Potential of RADARSAT-2 data for operational sea ice monitoring”. In: *Canadian Journal of Remote Sensing* 30.3 (2004), pp. 448–461. DOI: 10.5589/m04-011. URL: https://doi.org/10.5589/m04-011.

[74] Christopher Jackson and John R. Apel. *Synthetic Aperture Radar Marine User’s Manual*. Jan. 2004.

[75] Chen Wang et al. “A labelled ocean SAR imagery dataset of ten geophysical phenomena from Sentinel-1 wave mode”. In: *Geoscience Data Journal* 6.2 (2019), pp. 105–115.

[76] Jiuxiang Gu et al. *Recent Advances in Convolutional Neural Networks*. 2015. arXiv: 1512.07108.

[77] Keiller Nogueira, Otávio A.B. Penatti, and Jefersson A. dos Santos. “Towards better exploiting convolutional neural networks for remote sensing scene classification”. In: *Pattern Recognition* 61 (Jan. 2017), pp. 539–556. ISSN: 0031-3203. DOI: 10.1016/j.patcog.2016.07.001. URL: http://dx.doi.org/10.1016/j.patcog.2016.07.001.

[78] Marco Castelluccio et al. *Land Use Classification in Remote Sensing Images by Convolutional Neural Networks*. 2015. arXiv: 1508.00092.

[79] *MSTAR*. URL: https://www.sdms.afrl.af.mil/index.php?collection=mstar.

[80] Dimitrios Marmanis et al. “Deep Learning Earth Observation Classification Using ImageNet Pretrained Networks”. en. In: *IEEE Geoscience and Remote Sensing Letters* 13.1 (Jan. 2016), pp. 105–109. DOI: 10.1109/LGRS.2015.2499239. URL: http://ieeexplore.ieee.org/document/7342907/.

[81] Pierre Sermanet et al. *OverFeat: Integrated Recognition, Localization and Detection using Convolutional Networks*. 2013. eprint: 1312.6229.

[82] *ImageNet*. URL: http://www.image-net.org/.

[83] *UC Merced Land Use*. URL: http://weegee.vision.ucmerced.edu/datasets/landuse.html.

[84] Chen Wang et al. “Classification of the global Sentinel-1 SAR vignettes for ocean surface process studies”. en. In: *Remote Sensing of Environment* 234 (Dec. 2019), p. 111457. ISSN: 00344257. DOI: 10.1016/j.rse.2019.111457. URL: https://linkinghub.elsevier.com/retrieve/pii/S0034425719304766.

[85] Christian Szegedy et al. “Going deeper with convolutions”. In: (June 2015), pp. 1–9. DOI: 10.1109/CVPR.2015.7298594.

[86] Zhongling Huang, Zongxu Pan, and Bin Lei. “Transfer Learning with Deep Convolutional Neural Network for SAR Target Classification with Limited Labeled Data”. en. In: *Remote Sensing* 9.9 (Aug. 2017), p. 907. ISSN: 2072-4292. DOI: 10.3390/rs9090907. URL: http://www.mdpi.com/2072-4292/9/9/907.

[87] Diederik P. Kingma et al. *Semi-Supervised Learning with Deep Generative Models*. 2014. arXiv: 1406.5298 [cs.LG].

[88] *MNIST*. URL: http://yann.lecun.com/exdb/mnist/.

[89] Augustus Odena. "Semi-Supervised Learning with Generative Adversarial Networks". en. In: *arXiv:1606.01583 [cs, stat]* (Oct. 2016). URL: http://arxiv.org/abs/1606.01583.

[90] Fei Gao et al. "A Deep Convolutional Generative Adversarial Networks (DCGANs)-Based Semi-Supervised Method for Object Recognition in Synthetic Aperture Radar (SAR) Images". en. In: *Remote Sensing* 10.6 (May 2018), p. 846. ISSN: 2072-4292. DOI: 10.3390/rs10060846. URL: http://www.mdpi.com/2072-4292/10/6/846.

[91] *VGG16 – Implementation Using Keras*. URL: https://engmrk.com/vgg16-implementation-using-keras/.

[92] Adrian Rosebrock. *Fine-tuning with Keras and Deep Learning*. 2019. URL: https://www.pyimagesearch.com/2019/06/03/fine-tuning-with-keras-and-deep-learning/.

[93] Alec Radford, Luke Metz, and Soumith Chintala. "Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks". In: *CoRR* abs/1511.06434 (2015).

[94] Raúl Gombru. *Concatenate layer output with additional input data*. 2018. URL: https://discuss.pytorch.org/t/concatenate-layer-output-with-additional-input-data/20462.

[95] Giuseppe Maccari, Riccardo Nifosì, and Mariagrazia Di Luca. "Rational development of antimicrobial peptides for therapeutic use: design and production of highly active compounds". In: Dec. 2013, pp. 1265–1277. ISBN: 978-84-942134-0-3. DOI: 10.13140/2.1.4408.6726.

[96] *C-CORE*. URL: https://www.c-core.ca/.

[97] *Equinor*. URL: https://www.equinor.com/.

[98] *Statoil/C-CORE Iceberg Classifier Challenge*. 2018. URL: https://www.kaggle.com/c/statoil-iceberg-classifier-challenge/.

[99] John Murphy. *Deep Learning Benchmarks of NVIDIA Tesla P100 PCIe, Tesla K80, and Tesla M40 GPUs*. 2017. URL: https://www.microway.com/hpc-tech-tips/deep-learning-benchmarks-nvidia-tesla-p100-16gb-pcie-tesla-k80-tesla-m40-gpus/.

[100] Alexandre Lacoste et al. "Quantifying the Carbon Emissions of Machine Learning". In: *ArXiv* abs/1910.09700 (2019).

Appendix A

Appendix

A.1 DeepSAR-Net models

A.1.1 DeepSAR-Net - base version (TenGeoP-SARwv)

```
Model: "DeepSAR-Net"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         (None, 240, 240, 1)       0
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 120, 120, 1)       0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 120, 120, 20)      520
_________________________________________________________________
batch_normalization_1 (Batch (None, 120, 120, 20)      80
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 60, 60, 20)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 60, 60, 50)        9050
_________________________________________________________________
batch_normalization_2 (Batch (None, 60, 60, 50)        200
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 30, 30, 50)        0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 28, 28, 100)       45100
_________________________________________________________________
batch_normalization_3 (Batch (None, 28, 28, 100)       400
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 14, 14, 100)       0
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 12, 12, 200)       180200
_________________________________________________________________
batch_normalization_4 (Batch (None, 12, 12, 200)       800
_________________________________________________________________
max_pooling2d_5 (MaxPooling2 (None, 6, 6, 200)         0
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 4, 4, 400)         720400
_________________________________________________________________
batch_normalization_5 (Batch (None, 4, 4, 400)         1600
_________________________________________________________________
activation_1 (Activation)    (None, 4, 4, 400)         0
_________________________________________________________________
conv2d_6 (Conv2D)            (None, 1, 1, 500)         3200500
_________________________________________________________________
batch_normalization_6 (Batch (None, 1, 1, 500)         2000
_________________________________________________________________
activation_2 (Activation)    (None, 1, 1, 500)         0
_________________________________________________________________
flatten_1 (Flatten)          (None, 500)               0
_________________________________________________________________
dense_1 (Dense)              (None, 3)                 1503
=================================================================
Total params: 4,162,353
Trainable params: 4,159,813
Non-trainable params: 2,540
_________________________________________________________________
```

Figure A.1: DeepSAR-Net base architecture

A.1.2 DeepSAR-Net - Geo-based version (TenGeoP-SARwv)

Model: "DeepSAR-Net Geo-based"

Layer (type)	Output Shape	Param #	Connected to
input_1 (InputLayer)	(None, 240, 240, 1)	0	
max_pooling2d_1 (MaxPooling2D)	(None, 120, 120, 1)	0	input_1[0][0]
conv2d_1 (Conv2D)	(None, 120, 120, 20)	520	max_pooling2d_1[0][0]
batch_normalization_1 (BatchNor	(None, 120, 120, 20)	80	conv2d_1[0][0]
max_pooling2d_2 (MaxPooling2D)	(None, 60, 60, 20)	0	batch_normalization_1[0][0]
conv2d_2 (Conv2D)	(None, 60, 60, 50)	9050	max_pooling2d_2[0][0]
batch_normalization_2 (BatchNor	(None, 60, 60, 50)	200	conv2d_2[0][0]
max_pooling2d_3 (MaxPooling2D)	(None, 30, 30, 50)	0	batch_normalization_2[0][0]
conv2d_3 (Conv2D)	(None, 28, 28, 100)	45100	max_pooling2d_3[0][0]
batch_normalization_3 (BatchNor	(None, 28, 28, 100)	400	conv2d_3[0][0]
max_pooling2d_4 (MaxPooling2D)	(None, 14, 14, 100)	0	batch_normalization_3[0][0]
conv2d_4 (Conv2D)	(None, 12, 12, 200)	180200	max_pooling2d_4[0][0]
batch_normalization_4 (BatchNor	(None, 12, 12, 200)	800	conv2d_4[0][0]
max_pooling2d_5 (MaxPooling2D)	(None, 6, 6, 200)	0	batch_normalization_4[0][0]
conv2d_5 (Conv2D)	(None, 4, 4, 400)	720400	max_pooling2d_5[0][0]
batch_normalization_5 (BatchNor	(None, 4, 4, 400)	1600	conv2d_5[0][0]
activation_1 (Activation)	(None, 4, 4, 400)	0	batch_normalization_5[0][0]
conv2d_6 (Conv2D)	(None, 1, 1, 500)	3200500	activation_1[0][0]
batch_normalization_6 (BatchNor	(None, 1, 1, 500)	2000	conv2d_6[0][0]
activation_2 (Activation)	(None, 1, 1, 500)	0	batch_normalization_6[0][0]
input_2 (InputLayer)	(None, 1)	0	
flatten_1 (Flatten)	(None, 500)	0	activation_2[0][0]
dense_2 (Dense)	(None, 1)	2	input_2[0][0]
input_3 (InputLayer)	(None, 1)	0	
concatenate_1 (Concatenate)	(None, 501)	0	flatten_1[0][0] dense_2[0][0]
dense_3 (Dense)	(None, 1)	2	input_3[0][0]
concatenate_2 (Concatenate)	(None, 502)	0	concatenate_1[0][0] dense_3[0][0]
dense_4 (Dense)	(None, 500)	251500	concatenate_2[0][0]
dropout_1 (Dropout)	(None, 500)	0	dense_4[0][0]
dense_5 (Dense)	(None, 3)	1503	dropout_1[0][0]

Total params: 4,413,857
Trainable params: 4,411,317
Non-trainable params: 2,540

Figure A.2: DeepSAR-Net Geo-based architecture

A.1.3 DeepSAR-Net - Inc. angle-based version (C-CORE)

```
Model: "DeepSAR-Net Inc-based version"
_________________________________________________________________________________________________
Layer (type)                 Output Shape              Param #     Connected to
=================================================================================================
input_1 (InputLayer)         (None, 75, 75, 3)         0
_________________________________________________________________________________________________
conv2d_1 (Conv2D)            (None, 75, 75, 20)        1520        input_1[0][0]
_________________________________________________________________________________________________
batch_normalization_1 (BatchNor (None, 75, 75, 20)     80          conv2d_1[0][0]
_________________________________________________________________________________________________
max_pooling2d_1 (MaxPooling2D)  (None, 37, 37, 20)     0           batch_normalization_1[0][0]
_________________________________________________________________________________________________
conv2d_2 (Conv2D)            (None, 37, 37, 50)        9050        max_pooling2d_1[0][0]
_________________________________________________________________________________________________
batch_normalization_2 (BatchNor (None, 37, 37, 50)     200         conv2d_2[0][0]
_________________________________________________________________________________________________
max_pooling2d_2 (MaxPooling2D)  (None, 18, 18, 50)     0           batch_normalization_2[0][0]
_________________________________________________________________________________________________
conv2d_3 (Conv2D)            (None, 16, 16, 100)       45100       max_pooling2d_2[0][0]
_________________________________________________________________________________________________
batch_normalization_3 (BatchNor (None, 16, 16, 100)    400         conv2d_3[0][0]
_________________________________________________________________________________________________
max_pooling2d_3 (MaxPooling2D)  (None, 8, 8, 100)      0           batch_normalization_3[0][0]
_________________________________________________________________________________________________
conv2d_4 (Conv2D)            (None, 6, 6, 200)         180200      max_pooling2d_3[0][0]
_________________________________________________________________________________________________
batch_normalization_4 (BatchNor (None, 6, 6, 200)      800         conv2d_4[0][0]
_________________________________________________________________________________________________
conv2d_5 (Conv2D)            (None, 4, 4, 400)         720400      batch_normalization_4[0][0]
_________________________________________________________________________________________________
batch_normalization_5 (BatchNor (None, 4, 4, 400)      1600        conv2d_5[0][0]
_________________________________________________________________________________________________
activation_1 (Activation)    (None, 4, 4, 400)         0           batch_normalization_5[0][0]
_________________________________________________________________________________________________
conv2d_6 (Conv2D)            (None, 1, 1, 500)         3200500     activation_1[0][0]
_________________________________________________________________________________________________
batch_normalization_6 (BatchNor (None, 1, 1, 500)      2000        conv2d_6[0][0]
_________________________________________________________________________________________________
activation_2 (Activation)    (None, 1, 1, 500)         0           batch_normalization_6[0][0]
_________________________________________________________________________________________________
input_2 (InputLayer)         (None, 1)                 0
_________________________________________________________________________________________________
flatten_1 (Flatten)          (None, 500)               0           activation_2[0][0]
_________________________________________________________________________________________________
dense_2 (Dense)              (None, 1)                 2           input_2[0][0]
_________________________________________________________________________________________________
concatenate_1 (Concatenate)  (None, 501)               0           flatten_1[0][0]
                                                                   dense_2[0][0]
_________________________________________________________________________________________________
dense_3 (Dense)              (None, 500)               251000      concatenate_1[0][0]
_________________________________________________________________________________________________
dropout_1 (Dropout)          (None, 500)               0           dense_3[0][0]
_________________________________________________________________________________________________
dense_4 (Dense)              (None, 2)                 1002        dropout_1[0][0]
=================================================================================================
Total params: 4,413,854
Trainable params: 4,411,314
Non-trainable params: 2,540
_________________________________________________________________________________________________
```

Figure A.3: DeepSAR-Net Inc-based architecture

A.2 VGG16 models

A.2.1 VGG16 - base version (TenGeoP-SARwv)

```
Model: "VGG16"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         (None, 240, 240, 3)       0
_________________________________________________________________
block1_conv1 (Conv2D)        (None, 240, 240, 64)      1792
_________________________________________________________________
block1_conv2 (Conv2D)        (None, 240, 240, 64)      36928
_________________________________________________________________
block1_pool (MaxPooling2D)   (None, 120, 120, 64)      0
_________________________________________________________________
block2_conv1 (Conv2D)        (None, 120, 120, 128)     73856
_________________________________________________________________
block2_conv2 (Conv2D)        (None, 120, 120, 128)     147584
_________________________________________________________________
block2_pool (MaxPooling2D)   (None, 60, 60, 128)       0
_________________________________________________________________
block3_conv1 (Conv2D)        (None, 60, 60, 256)       295168
_________________________________________________________________
block3_conv2 (Conv2D)        (None, 60, 60, 256)       590080
_________________________________________________________________
block3_conv3 (Conv2D)        (None, 60, 60, 256)       590080
_________________________________________________________________
block3_pool (MaxPooling2D)   (None, 30, 30, 256)       0
_________________________________________________________________
block4_conv1 (Conv2D)        (None, 30, 30, 512)       1180160
_________________________________________________________________
block4_conv2 (Conv2D)        (None, 30, 30, 512)       2359808
_________________________________________________________________
block4_conv3 (Conv2D)        (None, 30, 30, 512)       2359808
_________________________________________________________________
block4_pool (MaxPooling2D)   (None, 15, 15, 512)       0
_________________________________________________________________
block5_conv1 (Conv2D)        (None, 15, 15, 512)       2359808
_________________________________________________________________
block5_conv2 (Conv2D)        (None, 15, 15, 512)       2359808
_________________________________________________________________
block5_conv3 (Conv2D)        (None, 15, 15, 512)       2359808
_________________________________________________________________
block5_pool (MaxPooling2D)   (None, 7, 7, 512)         0
_________________________________________________________________
global_average_pooling2d_1 ( (None, 512)               0
_________________________________________________________________
dense_1 (Dense)              (None, 512)               262656
_________________________________________________________________
dropout_1 (Dropout)          (None, 512)               0
_________________________________________________________________
dense_2 (Dense)              (None, 3)                 1539
=================================================================
Total params: 14,978,883
Trainable params: 264,195
Non-trainable params: 14,714,688
_________________________________________________________________
```

Figure A.4: VGG16 model architecture

A.2.2 VGG16 - FT version (C-CORE)

```
Model: "VGG16-FT"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_2 (InputLayer)         (None, 75, 75, 3)         0
_________________________________________________________________
block1_conv1 (Conv2D)        (None, 75, 75, 64)        1792
_________________________________________________________________
block1_conv2 (Conv2D)        (None, 75, 75, 64)        36928
_________________________________________________________________
block1_pool (MaxPooling2D)   (None, 37, 37, 64)        0
_________________________________________________________________
block2_conv1 (Conv2D)        (None, 37, 37, 128)       73856
_________________________________________________________________
block2_conv2 (Conv2D)        (None, 37, 37, 128)       147584
_________________________________________________________________
block2_pool (MaxPooling2D)   (None, 18, 18, 128)       0
_________________________________________________________________
block3_conv1 (Conv2D)        (None, 18, 18, 256)       295168
_________________________________________________________________
block3_conv2 (Conv2D)        (None, 18, 18, 256)       590080
_________________________________________________________________
block3_conv3 (Conv2D)        (None, 18, 18, 256)       590080
_________________________________________________________________
block3_pool (MaxPooling2D)   (None, 9, 9, 256)         0
_________________________________________________________________
block4_conv1 (Conv2D)        (None, 9, 9, 512)         1180160
_________________________________________________________________
block4_conv2 (Conv2D)        (None, 9, 9, 512)         2359808
_________________________________________________________________
block4_conv3 (Conv2D)        (None, 9, 9, 512)         2359808
_________________________________________________________________
block4_pool (MaxPooling2D)   (None, 4, 4, 512)         0
_________________________________________________________________
block5_conv1 (Conv2D)        (None, 4, 4, 512)         2359808
_________________________________________________________________
block5_conv2 (Conv2D)        (None, 4, 4, 512)         2359808
_________________________________________________________________
block5_conv3 (Conv2D)        (None, 4, 4, 512)         2359808
_________________________________________________________________
block5_pool (MaxPooling2D)   (None, 2, 2, 512)         0
_________________________________________________________________
global_average_pooling2d_1 ( (None, 512)               0
_________________________________________________________________
dense_1 (Dense)              (None, 512)               262656
_________________________________________________________________
dropout_1 (Dropout)          (None, 512)               0
_________________________________________________________________
dense_2 (Dense)              (None, 2)                 1026
=================================================================
Total params: 14,978,370
Trainable params: 2,623,490
Non-trainable params: 12,354,880
_________________________________________________________________
```

Figure A.5: VGG16-FT model architecture

A.3 DeepSAR-Net-SGAN models

A.3.1 Discriminator (TenGeoP-SARwv)

```
Model: "Discriminator"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         (None, 240, 240, 1)       0
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 120, 120, 1)       0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 120, 120, 20)      520
_________________________________________________________________
batch_normalization_1 (Batch (None, 120, 120, 20)      80
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 60, 60, 20)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 60, 60, 50)        9050
_________________________________________________________________
batch_normalization_2 (Batch (None, 60, 60, 50)        200
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 30, 30, 50)        0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 28, 28, 100)       45100
_________________________________________________________________
batch_normalization_3 (Batch (None, 28, 28, 100)       400
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 14, 14, 100)       0
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 12, 12, 200)       180200
_________________________________________________________________
batch_normalization_4 (Batch (None, 12, 12, 200)       800
_________________________________________________________________
max_pooling2d_5 (MaxPooling2 (None, 6, 6, 200)         0
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 4, 4, 400)         720400
_________________________________________________________________
batch_normalization_5 (Batch (None, 4, 4, 400)         1600
_________________________________________________________________
activation_1 (Activation)    (None, 4, 4, 400)         0
_________________________________________________________________
conv2d_6 (Conv2D)            (None, 1, 1, 500)         3200500
_________________________________________________________________
batch_normalization_6 (Batch (None, 1, 1, 500)         2000
_________________________________________________________________
activation_2 (Activation)    (None, 1, 1, 500)         0
_________________________________________________________________
flatten_1 (Flatten)          (None, 500)               0
_________________________________________________________________
dropout_1 (Dropout)          (None, 500)               0
_________________________________________________________________
dense_1 (Dense)              (None, 3)                 1503
_________________________________________________________________
activation_3 (Activation)    (None, 3)                 0
=================================================================
Total params: 4,162,353
Trainable params: 4,159,813
Non-trainable params: 2,540
_________________________________________________________________
```

Figure A.6: DeepSAR-Net-SGAN Discriminator

A.3.2 Generator (TenGeoP-SARwv)

```
Model: "Generator"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_2 (InputLayer)         (None, 500)               0
_________________________________________________________________
dense_2 (Dense)              (None, 500)               250500
_________________________________________________________________
reshape_1 (Reshape)          (None, 1, 1, 500)         0
_________________________________________________________________
conv2d_transpose_1 (Conv2DTr (None, 4, 4, 500)         4000500
_________________________________________________________________
batch_normalization_7 (Batch (None, 4, 4, 500)         2000
_________________________________________________________________
leaky_re_lu_1 (LeakyReLU)    (None, 4, 4, 500)         0
_________________________________________________________________
conv2d_transpose_2 (Conv2DTr (None, 9, 9, 400)         1800400
_________________________________________________________________
batch_normalization_8 (Batch (None, 9, 9, 400)         1600
_________________________________________________________________
leaky_re_lu_2 (LeakyReLU)    (None, 9, 9, 400)         0
_________________________________________________________________
conv2d_transpose_3 (Conv2DTr (None, 19, 19, 200)       720200
_________________________________________________________________
batch_normalization_9 (Batch (None, 19, 19, 200)       800
_________________________________________________________________
leaky_re_lu_3 (LeakyReLU)    (None, 19, 19, 200)       0
_________________________________________________________________
conv2d_transpose_4 (Conv2DTr (None, 39, 39, 100)       180100
_________________________________________________________________
batch_normalization_10 (Batc (None, 39, 39, 100)       400
_________________________________________________________________
leaky_re_lu_4 (LeakyReLU)    (None, 39, 39, 100)       0
_________________________________________________________________
conv2d_transpose_5 (Conv2DTr (None, 79, 79, 50)        45050
_________________________________________________________________
batch_normalization_11 (Batc (None, 79, 79, 50)        200
_________________________________________________________________
leaky_re_lu_5 (LeakyReLU)    (None, 79, 79, 50)        0
_________________________________________________________________
conv2d_transpose_6 (Conv2DTr (None, 240, 240, 20)      25020
_________________________________________________________________
batch_normalization_12 (Batc (None, 240, 240, 20)      80
_________________________________________________________________
leaky_re_lu_6 (LeakyReLU)    (None, 240, 240, 20)      0
_________________________________________________________________
conv2d_7 (Conv2D)            (None, 240, 240, 1)       501
=================================================================
Total params: 7,027,351
Trainable params: 7,024,811
Non-trainable params: 2,540
_________________________________________________________________
```

Figure A.7: DeepSAR-Net-SGAN Generator