Department of Information Engineering and Computer Science

Master's Degree in

Information Communication Engineering

FINAL DISSERTATION

SELF-SUPERVISED LEARNING FOR SEMANTIC SEGMENTATION OF POL-SAR IMAGES VIA SAR-TO-OPTICAL TRANSCODING

Co-Supervisor
Ronny Hänsch

Student
Alessandro Cattoi

Supervisor
Lorenzo Bruzzone

Academic year 2019/2020

La scivolata non è pericolosa, finché è controllata.

Ignoti

Abstract

In the remote sensing field, the data acquisition process has grown a lot in the last decades. Every year many new technologies are deployed, such as sensors, satellites, and space exploration missions. This effort in exploring the unknown produces a lot of valuable data. This amount of information, if left stored on hard drives, can become a liability instead of being an asset. For these reasons, it is of paramount importance to leverage the gathered data. This is the motivation why automatic data analysis is a trending topic in the scientific community.

The objective of this thesis is focused on Polarimetric Synthetic Aperture Radar (**Pol-SAR**) images semantic segmentation. The state-of-the-art techniques, to automatically interpret images, are called artificial neural networks. These techniques are algorithms trained to perform tasks learning by examples. This means that the most important requirement, to train such algorithms, is data annotations. Annotations are additional hand-crafted information that contains the data interpretation. The algorithms trained using annotated data are addressed as supervised. The optical images annotation process is simpler with respect to the one of **Pol-SAR** images. Every person observing optical images can recognize the content. Instead, the interpretation of **Pol-SAR** images requires experts. This makes the annotation process a costly and inefficient procedure. In literature, there are also techniques that do not need annotations. These techniques are addressed as unsupervised and they learn by autonomously finding patterns in data.

This thesis employs a technique that combines the capacities of supervised and unsupervised learning. This technique is called self-supervision. The self-supervised paradigm automatically generates the annotations for the training process. In this work, the self-supervision is achieved by transcoding **Pol-SAR** images to optical images. Transcoding algorithms do not need human supervision to learn, they just need optical and radar image couples to learn to perform the transformation from a domain to the other. The self-supervised frameworks are based on two tasks that require two training phases. The first one is called proxy task and in this work it is implemented as **Pol-SAR** to optical transcoding. This phase is designed to learn strong representative features for radar images. The second one is called downstream task and in this thesis is implemented by retraining some layers of the transcoders to perform the semantic segmentation task. This phase aims to adapt the knowledge, learned employing the proxy task, to the above-mentioned downstream task.

In this work are implemented three different transcoders to perform the proxy task. They are based on three different training strategies. One is based on a regressive loss. One is implemented using the conditional adversarial generative paradigm. One is based on a cycle consistent generative adversarial paradigm. These three methods are of increasing complexity. The first two are supervised, since they need paired radar and optical images. The last one, unlike the first two, is fully unsupervised, since it could be trained using unpaired datasets. In this thesis, this framework has been trained only using paired datasets.

This work demonstrates that the self-supervised learning paradigm can potentially leverage unused data, achieving high-quality semantic segmentation accuracy results. Moreover, this thesis evidences the potential of image transcoding employed as proxy task for **Pol-SAR** images semantic segmentation. Finally, in scenarios with very few annotated data, this framework shows how pre-trained algorithms maintain high-performance levels compared to their fully supervised baseline.

This thesis analyses the implemented algorithm architectures to find the optimal configuration. It is shown that the networks implemented perform better using deeper architectures and patch sizes of 128×128 pixels with respect to patch size of 256×256 pixels.

The three pre-trained algorithms implemented outperform the same architecture trained in a supervised fashion using all the labeled data available. In particular, the Conditional Adversarial Transcoder (**CAT**) implemented

is the best classifier: it reaches 89% of overall accuracy. The Regressive Transcoder (**RT**) achieves 87% of overall accuracy. The Cycle consistent adversarial transcoder (**Cycle-AT**) achieves 76% of overall accuracy. The benchmarking Baseline (**BL**) reaches 73% of overall accuracy.
This thesis also investigates the classifiers strength in scenarios with very few annotations. The **CAT** and the **RT**, when fine-tuned with only 5% of the total available labels, still outperform the **BL** capacity trained using all the available annotations. In this situation, they classify **Pol-SAR** images with an accuracy higher than 80%. All the pre-trained networks, including the **Cycle-AT**, are very resilient to annotated data scarcity. In fact, their semantic segmentation capacity decreases only by around 6 percentage points when they are trained with very few labels. The fully supervised framework implemented, if trained using 5% of available data, breaks down to 40% of classification accuracy.
This framework also demonstrates that a very simple training strategy as the **RT** achieves very interesting results, even if it is not the strongest features extractor. The **RT** is conceptually and computationally simpler, faster in reaching its optimal, and has a more stable learning process with respect to the other transcoders.
These results demonstrate how the transcoding from **Pol-SAR** to optical images employed as proxy task, is a very strong technique to achieve high-performance leveraging unlabeled data in scenarios where annotations are scarce and their procurement is costly.

Keywords: Self-Supervised Learning, Radar to Optical Images Transcoding, Generative Adversarial Networks, Semantic Segmentation, Pre-Training, Polarimetric Synthetic Aperture Radar Images.

Contents

1 Introduction **1**
1.1 Context . . . 1
1.2 Objective . . . 2
1.3 Contribution . . . 3

2 Foundations and Related Works **5**
2.1 Radar Systems . . . 5
2.2 Synthetic Aperture Radars . . . 6
2.2.1 Installation . . . 6
2.2.2 Resolution . . . 7
2.2.3 Polarimetric Synthetic Aperture Radars . . . 8
2.2.4 Synthetic Aperture Radar Images Singularities . . . 9
2.3 Deep Learning . . . 10
2.3.1 Supervised Learning . . . 10
2.3.2 Unsupervised Learning . . . 10
2.3.3 Weakly-Supervised Learning . . . 11
2.3.4 Transfer Learning and Fine-Tuning . . . 11
2.3.5 Semi-Supervised Learning . . . 11
2.3.6 Self-Supervised Learning . . . 12
2.4 Convolutional Neural Networks . . . 12
2.4.1 Convolution Operation . . . 12
2.4.2 Convolution Motivation . . . 13
2.4.3 Convolution Architecture . . . 13
2.5 Generative Adversarial Networks . . . 15
2.5.1 GAN Objective . . . 16
2.5.2 GAN Training . . . 17
2.6 Conditional Generative Adversarial Networks . . . 17
2.7 Cycle Consistent Generative Adversarial Networks . . . 18
2.8 Radar Images Classification . . . 19
2.9 Related works . . . 20

3 Dataset **23**
3.1 Data for Transcoding . . . 23
3.2 Data for Classification . . . 25
3.3 Data Splitting . . . 25
3.4 Data Preparation . . . 26

4 Methodology **27**
4.1 Network Architectures . . . 27
4.1.1 Generator . . . 27
4.1.2 Discriminator . . . 30
4.1.3 Classifier . . . 31
4.1.4 Implementation Details . . . 32

4.2 Networks Training . . . 32
4.2.1 Baseline . . . 33
4.2.2 Regressor Transcoder . . . 34
4.2.3 Conditional Adversarial Transcoder . . . 35
4.2.4 Cycle Consistent Adversarial Transcoder . . . 37
4.2.5 Fine-Tuning . . . 39
4.2.6 Networks Complexity . . . 39

5 Pipeline Bug Discovery **41**
5.1 Bug Explanation . . . 41
5.2 Wrongly Filtered Data Analysis . . . 42
5.3 The Role of the Incorrect Data in this Work . . . 43
5.4 The Effects on the Results . . . 43

6 Experiments and Networks Tuning **45**
6.1 Transcoding Evaluation . . . 45
6.2 Semantic Segmentation Metric . . . 45
6.3 Computational Power and Execution Time . . . 46
6.4 Networks Tuning . . . 47
6.4.1 Generated Samples Bufferization . . . 47
6.4.2 Reduced Discriminator Optimization Step . . . 47
6.4.3 Channels Dropping . . . 48
6.5 Network Experiments . . . 49
6.5.1 Patch Size . . . 49
6.5.2 Generator Deepness . . . 51
6.5.3 Discriminator Deepness . . . 52

7 Results and Discussions **53**
7.1 Final Configuration . . . 53
7.2 Transcoding Results . . . 54
7.2.1 Transcoders Comparison . . . 56
7.2.2 Cycle Consistent Adversarial Transcoder Analysis . . . 58
7.3 Semantic Segmentation Results . . . 59
7.3.1 Semantic Segmentation Accuracy Analysis . . . 60
7.3.2 Accuracy Results Statistical Validation . . . 62
7.3.3 Labeled Samples Sensitivity Analysis . . . 63
7.3.4 Classification Maps Analysis . . . 66

8 Conclusions **71**
8.1 Final Considerations . . . 71
8.2 Future Works . . . 72

Appendices **73**

A Scattering Matrix Filtering **75**
A.1 Correct Filter Usage . . . 75
A.2 Wrong Filter Usage . . . 76

B Original Data Tiles **77**
B.1 Pol-SAR Train Tile 78
B.2 Optical Train Tile 78
B.3 Label Map Train Tile 79
B.4 Pol-SAR Test Tile 79
B.5 Optical Test Tile 80
B.6 Label Map Test Tile 80

C Full Transcoded Images **81**
C.1 Train Tile Transcoding Result - Regressive Transcoder 82
C.2 Test Tile Transcoding Result - Regressive Transcoder 82
C.3 Train Tile Transcoding Result - Conditional Adversarial Transcoder 83
C.4 Test Tile Transcoding Result - Conditional Adversarial Transcoder 83
C.5 Train Tile Transcoding Result - Cycle Consistent Adversarial Transcoder 84
C.6 Test Tile Transcoding Result - Cycle Consistent Adversarial Transcoder 84

D Confusion Matrices **85**
D.1 Confusion Matrix - Baseline 86
D.2 Confusion Matrix - Regressive Transcoder 87
D.3 Confusion Matrix - Conditional Adversarial Transcoder 88
D.4 Confusion Matrix - Cycle Consistent Adversarial Transcoder 89

List of Figures **95**

List of Tables **98**

List of Acronyms **99**

Bibliography **101**

Chapter 1

Introduction

Contents

1.1 Context 1
1.2 Objective 2
1.3 Contribution 3

1.1 Context

Information is becoming the most powerful asset for humanity. In the last decades, the progress has been focused on gathering and leveraging information to improve every aspect of human life. Information permeates everything, everywhere, from health to war, from climate change to universe exploration.
To be informed is crucial to manage and optimize every single thing, information management is a key factor to improve our society.
This field is enormously expanding, the biggest and richest companies in the world work with information. As a result, every year new systems are deployed to gather additional information, to monitor the climate, to map unexplored universe areas. There are plenty of companies and governments which are investing in this direction.
Even if we pay attention only to image acquisition systems, almost every electronic device has at least one sensor onboard: smartphones, cars, laptops, fridges, doorbells, etc.; without taking into account all monitoring systems such as security cameras, traffic control systems, or industrial automation machines. Besides, all around the globe are orbiting around 4300 satellites [Wood 2020] for earth observation purposes. Moreover, there are space missions to observe the universe, imaging systems on airplane, drones, etc.
All this huge amount of information has to be used to be transformed from a liability to an asset.
To extract valuable information from raw data is one of the most complex step in the whole process, which starts with data acquisition and ends with meaningful output based on processed data. This work is focused on data that comes from satellites for earth observation.
Satellites such as Sentinel-1, Sentinel-2 (SEN1-2) of the European Space Agency (ESA) [esa.int a, esa.int b] or Landsat-8 of the National Aeronautics and Space Administration (NASA) [nasa.gov], continuously rotate around the globe acquiring data. As mentioned above this very large amount of data are continuously being acquired by many different systems, but if they are not used is a loss of resources.

1.2 Objective

As explained in Section 1.1 there are a lot of data coming from satellites, the crucial question concerns how to use this amount of data. There are a lot of possible applications, an example could be to measure the extension of a desert or monitor a forest to check if there is a fire. Imaging sensors installed on satellites, even if used for very different purposes, can be divided into two groups:

- *Passive sensors*: passively measure light waves backscattered from the earth illuminated by the sun, for example, Red, Green, Blue (RGB) images are passive signals.
- *Active sensors*: actively illuminate the earth sending an electromagnetic wave, and measuring the backscattered echoes by the surface, for instance, aircrafts traffic control is managed by means of active sensors.

Figure 1.1: The left half of the image is a passive RGB image, the right half is an active radar image.

This thesis focuses on active data analysis. Radar systems are particularly interesting because they do not rely on the sun to illuminate their target, and they are resistant to cloud [Hänsch & Hellwich 2017]. Unlike, optical systems are blind with bad weather conditions and during the night. The employment of radar systems involves not only advantages but, as it is possible to see in *Figure 1.1*, some disadvantages: the interpretation of images that come from active systems is not trivial [Merkle *et al.* 2017]. Radar interpretation requires experts to be performed [Wang *et al.* 2019]. This, combined with the huge quantity of data, makes the processing step a very costly and time-demanding task. For these reasons automatic data analysis is a trending topic, a lot of the effort of the scientific community is going in this direction.
Automatic data analysis has the ability to find patterns in data. When the input and the expected output are given, they are called supervised learning algorithms, otherwise, whether only the input is given and the algorithm tries to find patterns autonomously, they are called unsupervised learning algorithms. Typically, supervised learning outperforms unsupervised learning [Jing & Tian 2020]. Unluckily, but obviously, to train a supervised algorithm is necessary to manually annotate a lot of data, which, as mentioned above, is very costly, in particular for active systems. The interpretation of radar images is very complicated and requires a lot of experience. This is particularly true for Polarimetric Synthetic Aperture Radar (Pol-SAR) images, for which the procurement of labeled data is very difficult [Hua *et al.* 2019].
This work explores an innovative learning paradigm that tries to leverage the potentiality of the unsupervised techniques to improve the capacity of the supervised ones. This family of techniques, in literature, is called

semi-supervised or self-supervised learning but it can even be addressed more generally as features learning [Raghu & Schmidt 2020].
The core idea is to leverage the big amount of unannotated data to achieve the state of the art results using less labeled data.

1.3 Contribution

This work aims to apply a self-supervised learning paradigm to the classification of Pol-SAR images. The idea is to use the large availability of radar and optical data of the earth's surface to develop an unsupervised algorithm. This algorithm is then used to extract meaningful features helpful for image classification. Using features extraction algorithms to perform the classification task is convenient because it allows the development of high-quality classifiers using fewer annotated data. In this thesis, the features extraction training phase has been performed using different types of architectures of increasing complexity. The simplest is a regressor, then a Generative Adversarial Network (GAN) has been implemented, up to the most complex implementation, which is based on a Cycle Consistent Generative Adversarial Network (Cycle-GAN). Cycle-GAN is one of the latest algorithms for image domain adaptation, which unlike the other two concepts, can convert images from a domain to another without necessarily using paired images [Zhu *et al.* 2017a]. These pre-training strategies are compared to investigate which performs better in extracting representative features for the classification task. To investigate which algorithm achieves the best performances all the implementations are trained and tested using the same data. Besides the framework has been investigated to study the classification capacity varying the availability of annotated data. Finally, the three networks used as proxy tasks are used to perform preliminary investigations to explore their sensibility to specific hyperparameters, such as, patch size and architecture deepness. The attempt to use the Cycle-GAN should bring a big advantage: it is not necessary to have two versions of the same image perfectly aligned to train the network. It is sufficient to have two sets of images that describe well the two domains. This big plus requires a much higher complexity with respect to a common regressor or to a GAN.
The framework implemented in this work is a clear example that transforms the stream of data coming from satellites in an asset to improve a common task, such as image classification, that normally requires a big quantity of expansive annotated data.

CHAPTER 2

Foundations and Related Works

Contents

2.1 Radar Systems **5**
2.2 Synthetic Aperture Radars **6**
2.2.1 Installation 6
2.2.2 Resolution 7
2.2.3 Polarimetric Synthetic Aperture Radars 8
2.2.4 Synthetic Aperture Radar Images Singularities 9
2.3 Deep Learning **10**
2.3.1 Supervised Learning 10
2.3.2 Unsupervised Learning 10
2.3.3 Weakly-Supervised Learning 11
2.3.4 Transfer Learning and Fine-Tuning 11
2.3.5 Semi-Supervised Learning 11
2.3.6 Self-Supervised Learning 12
2.4 Convolutional Neural Networks **12**
2.4.1 Convolution Operation 12
2.4.2 Convolution Motivation 13
2.4.3 Convolution Architecture 13
2.5 Generative Adversarial Networks **15**
2.5.1 GAN Objective 16
2.5.2 GAN Training 17
2.6 Conditional Generative Adversarial Networks **17**
2.7 Cycle Consistent Generative Adversarial Networks **18**
2.8 Radar Images Classification **19**
2.9 Related works **20**

2.1 Radar Systems

A radar is an antenna that radiates electromagnetic energy into space. Some of the radiated energy is reflected by an object called target. The target reflects the energy in many directions, some of these, so-called echoes, are intercepted by the radar antenna. The antenna, used as a receiver now, after a processing and amplification step, can measure the reflected wave which is represented by an amplitude and a phase. Nowadays many types of radars exist, this work concentrates on radar for imaging. One of the fundamental parameters of radars for imaging is resolution. Radar image resolution is typically defined in two directions, azimuth, and range. The resolution of each pixel of a radar image can not be squared due to the nature of

radar image acquisition. The range resolution refers to the resolution perpendicularly acquired with respect to the installation platform. The azimuth resolution refers to the resolution in parallel acquired with respect to the installation platform. In *Subsection* 2.2.2 radar resolution is deeply discussed. Radars for imaging can be divided into these categories:

- *Imaging Radars*: acquire two-dimensional images such as a portion of the planet's surface.
- *Side Looking Radars*: acquire high-resolution images in range and obtain high-resolution in azimuth using a narrow beam antenna.
- *Synthetic Aperture Radars*: are a coherent imaging systems installed on moving platforms, which acquire high-resolution images by combining signals with different phase information.
- *Inverse Synthetic Aperture Radars*:are coherent imaging systems that acquire images by using the doppler effect to achieve high-resolution in range.

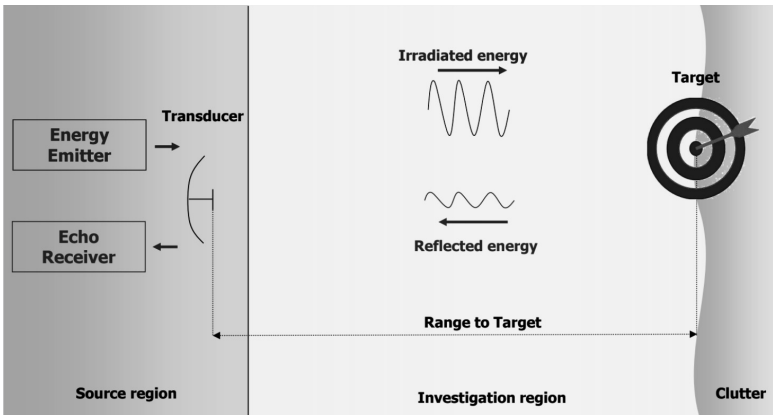


Figure 2.1: A general scheme on the radars functioning [Bruzzone 2019].

Each type of radar has a very specific field of application, that can vary from military surveillance to undersurface planet exploration. This work focuses on the remote sensing of the environment. This category of radars are used to monitor the environment: for example, they are used to measure vegetation status or glacier advance [Skolnik 2008].

2.2 Synthetic Aperture Radars

The synthetic aperture is an advanced processing technique for radar images. Radars for imaging have some specific characteristics: they employ the synthetic aperture to achieve a big improvement of the image quality.

2.2.1 Installation

A radar is an antenna that emits and receives an electromagnetic field, if it is installed on a static platform it continuously monitors a specific target. Otherwise, radars can be installed on moving platforms, such as aircrafts. The former can be used for surveillance: the sensor always monitors the same place. The latter can map very large areas. For example, satellites can map the entire earth surface as it is possible to see in *Figure* 2.2.

Figure 2.2: The image shows a satellite which rotates around the earth. In blue the areas interested by its sensor [Bruzzone 2017].

Radars do not acquire images illuminating the target perpendicularly. As it is possible to see in *Figure* 2.3 the radar is positioned side-looking [Merkle *et al.* 2017]. If the radar is perpendicularly positioned with respect to the surface it would be impossible to distinguish all the echoes. Echoes returning from two targets, equally distanced from the antenna, in two different halves of the beam footprint, will arrive at the sensor at the same time. As a result, there are ambiguities, not all the resolution cells of an image can be distinguished [Skolnik 2008].

2.2.2 Resolution

Normally the spatial resolution of a radar is directly related to the footprint of the beam on the surface. The resolution is defined in two directions: azimuth and range, shown in *Figure* 2.3 as x and y, respectively. The resolution in range depends on the pulse duration, the shorter it is the better the resolution. But this is in contrast with the power of the signal (the power of the pulse depends directly on its duration), which is fundamental to cover a long distance between the antenna and the target. Using specific types of waveform for the radar pulses can strongly mitigate the problem [Bruzzone 2017].
The azimuth resolution is directly related to the ratio between the sensor wavelength and the length of the sensor's antenna. For a given wavelength, the longer the antenna, the higher the spatial resolution. From a satellite in space operating at a wavelength of about 5 centimeter (C-band radar), to get a spatial resolution of 10 meters, it would be necessary a radar antenna about 4.250 meters long. That is unfeasible. To overcome this issue the so-called synthetic aperture is employed. In this concept, a sequence of acquisitions from a shorter antenna are combined to simulate a much larger antenna, thus providing higher resolution data [Flores-Anderson *et al.* 2019].

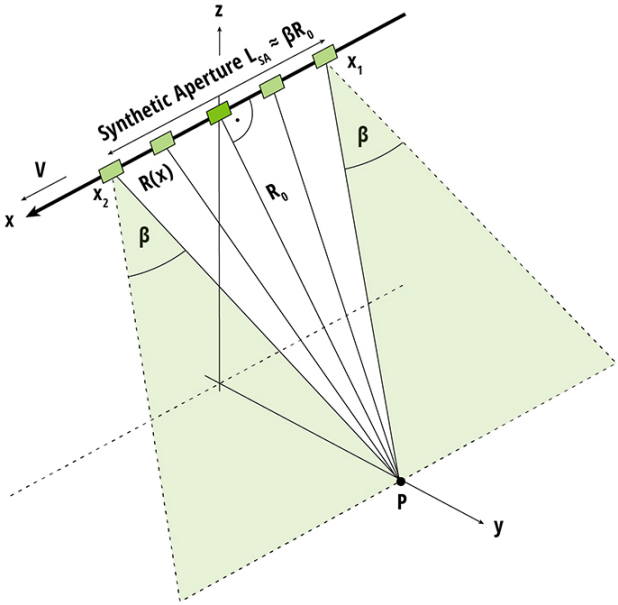


Figure 2.3: Each position of the radar antenna (green squares) contributes to the quality of the single point *"P"*. Normally, the information belonging to the point *"P"* would be only acquired by the antenna in the central position. Applying the synthetic aperture is possible to achieve the resolution as if the antenna would be as long as from *x1* to *x2*. Actually, the synthetic aperture combines the echoes coming from the point *"P"* at each position of the antenna.[Flores-Anderson *et al.* 2019].

2.2.3 Polarimetric Synthetic Aperture Radars

Pol-SAR belongs to the family of multimodal Synthetic Aperture Radar (SAR) systems, which enforce their effectiveness by using more than one frequency or more than one polarization. Polarization is the orientation of the propagating wave. When a target is illuminated with different polarizations more information can be gathered. Diverse polarizations represent a type of information different from any other. A fully polarized radar can transmit and receive microwaves vertically and horizontally polarized. The measured echoes can be represented using a scattering matrix characterized by four scattering coefficients: two co-polarized complex scattering amplitudes and two complex cross-polarized amplitudes. Pol-SAR images are often represented using the covariance matrix, since normal-complex distributed targets are fully described by it.
The information gathered by this technique depends on moisture, surface roughness, geometry, etc. All these properties are very useful in classification. Nowadays many sensors provide Pol-SAR data such as TerraSAR-X (TSX) from German Aerospace Center (DLR). Due to the increasing amount of data availability and the difficulties of human analysis, automatic analysis of Pol-SAR data is strongly required [Hänsch & Hellwich 2017].

2.2.4 Synthetic Aperture Radar Images Singularities

Due to the nature of the side-looking acquisition, there are three limitations in SAR images [Flores-Anderson *et al.* 2019]:

- *Foreshortening*: equally distanced points can appear nearer or farther in the image.

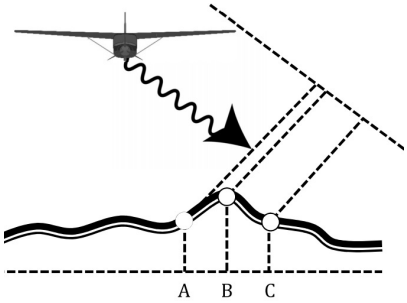


Figure 2.4: Point A, B, C are equidistant on the target surface but due to the surface morphology they appear not equidistant to the sensor [Bruzzone 2017].

- *Shadows*: some parts of the images can be shadowed.

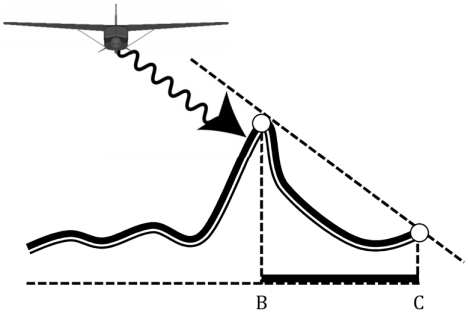


Figure 2.5: The area between point B and C is not illuminated by the sensor [Bruzzone 2017].

- *Layover*: some points can appear beyond the ones that were originally behind.

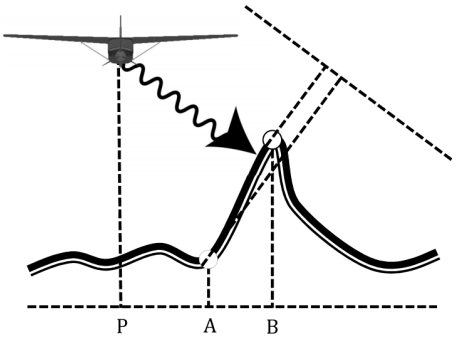


Figure 2.6: Point A results farther than point B in the SAR image [Bruzzone 2017].

2.3 Deep Learning

A Artificial Neural Network (ANN) is a computer science technique. ANNs autonomously extract relationships in a set of data. This set of algorithms is called Machine Learning (ML), since machines learn to perform a specific task. A lot of steps forward have been done in this field in the last few years: the main significant one is Deep Learning (DL), an ANN with many hidden layers, potentially, with a much stronger learning capacity. The strength of deep learning is the capacity to autonomously extract complex features by composing elementary ones. Deep learning solves the difficulties by splitting a complex representation into a composition of simpler ones. One big perk of autonomously extracting features is to avoid the process of designing hand-crafted features. Hand-crafted features are costly because they require a lot of effort in terms of both expertise and time. Another perk of DL features is that they are optimal for the learning target, since they are directly extracted optimizing the objective of the algorithm [Goodfellow *et al.* 2016]. Deep Learning offers several advantages:

- *Self-extracted features*: more complex and more representative features than hand-crafted one.
- *Unsupervised learning*: leverage unlabeled data, which are more abundant and cheaper.
- *Multiple levels of representation*: each level represents different and more complex features combining the ones extracted in the previous levels.
- *Multi-tasking*: the same architecture can perform for different problems.
- *Invariant and disentangling*: avoids the curse of dimensionality.

2.3.1 Supervised Learning

Supervised learning is the most common, the most straightforward and the most performing training methodology for artificial neural networks. It is based on a fundamental requirement: the dataset must be made of input data accompanied with labels. For example, the *MNIST handwritten digit database* [LeCun *et al.* 2010] is composed of images of handwritten digits. Each data sample is accompanied by a value that labels the handwritten digit in the image. This paired information is fundamental because the network is fed with input data, a handwritten digit, and it generates an output: the value corresponding to the digit. The network is then optimized based on the difference between the label and the prediction of the artificial neural network. In theory, the more the data, the better the quality of the training should be. One of the most critical steps of designing a supervised algorithm is the balancing between problem complexity and data availability. The architecture of the algorithm has to be strong enough to represent the target task, and the data have to be sufficient to train the network designed [Goodfellow *et al.* 2016].

2.3.2 Unsupervised Learning

Unsupervised learning is a training strategy that uses unlabeled data to extract strongly representative features or statistics that characterize the input data distribution. It is a very potential field. The capacity of developing an algorithm able to learn powerful data representation without any additional aid, could be disruptive for developing new technologies.
In unsupervised learning, the algorithm is fed with many samples without any other additional input. The training procedure updates the algorithm trying to find patterns in data. To perform this operation the computer does not need human support. The ANNs try to find by itself peculiarities in data. There are many examples of unsupervised frameworks in literature, one typical architecture are the autoencoders, an artificial neural network that encodes and decodes an image. It works by compressing and then decompressing an image using

two ANNs called: encoder and decoder. The output of the encoder is a compressed representation of the input image which should enclose all its fundamental characteristics. This process is performed in a completely unsupervised fashion, simply by learning to reproduce the input. Another typical application of unsupervised learning is clustering algorithms: algorithms that find similarities in data and cluster them in coherent groups [Goodfellow *et al.* 2016].

2.3.3 Weakly-Supervised Learning

When collecting high quality and high quantity labels is not a trivial task, a possible solution is to apply weak supervision. In this scenario, the algorithm is trained using noisy labels which might be relatively easy to obtain. In computer vision, some works have proposed to use Instagram tags as image labels [Mahajan *et al.* 2018] or to use classification labels to perform segmentation [Pathak *et al.* 2015]. This class of algorithms are trained in a typical supervised fashion (see *Subsection* 2.3.1). The labels employed are defined as weak or noisy that means that are not very accurate.

2.3.4 Transfer Learning and Fine-Tuning

Transfer learning is a process for training deep learning algorithms. It allows to transfer the knowledge learned from a proxy task to a related but different target task. More in detail, a randomly initialized ANN is trained to perform a pre-training task, for example using an unsupervised paradigm. With an appropriate pre-training task that is generic and complex enough, the features learning step allows the artificial neural network to learn strong representations. The features extraction capability is stored in its parameters, and it can be reused for the second step, the fine-tuning [Raghu & Schmidt 2020]. In fine-tuning the pre-trained ANN is retrained, often with minor modifications, to perform the final objective task. This approach has shown very interesting results in computer vision [Miyato *et al.* 2019].

2.3.5 Semi-Supervised Learning

Semi-supervised learning is a technique that combines the supervised and the unsupervised learning paradigms. This method fuses the advantages of leveraging many unlabeled data with the capacity of supervised learning. It proposes to extract meaningful features from a scenario in which there are few labeled data and many unlabeled data. The goal is to find a way to extract strongly representative features from unlabeled data. Then it tries to employ the features extracted to improve the final objective. The combination of unsupervised features and the labeled data can improve the performances [Raghu & Schmidt 2020]. Here below some semi-supervised techniques are described:

- *Self-training*: also known as *bootstrapping* or *pseudo-labels*, is a deep artificial neural network firstly trained with labeled data. Then the network is used to auto-generate its own labels. In the last phase, the generated predictions and the original labels are used in combination to iteratively train the network [Lee 2013].

- *Co-training*: two networks are trained on different parts of a dataset. Then the two networks are used to generate predictions of unlabeled data. Predictions with high confidence from each model are used as labels for the other [Blum & Mitchell 1998].

- *Enforcing consistency*: trains a network to recognize an image and its perturbed version. This is a strategy to enforce the network. Using the algorithm to recognize a more complex version of the same image force the network reinforce its recognition capacities [Miyato *et al.* 2019].

2.3.6 Self-Supervised Learning

In self-supervision frameworks, the algorithms learn to perform a specific task by autonomously generating labels for raw data. This unsupervised learning task is usually called *pretext task* or *proxy task*. A typical use case of self-supervised learning is composed of two phases [Jing & Tian 2020]:

- *Features extraction*: in which the algorithm performs the proxy task learning strongly representative features.
- *Fine-tuning*: in which the algorithm transfers the capacity learned by means of the self-supervised task using few labeled data.

An example is the work of [Gidaris *et al.* 2018], in which a network learns to recognize the degree of rotation of input raw data. The degree of rotation is arbitrarily decided, so it is known, and it is used as the label for the prediction. The pretext task teaches the network good representation without the need for any human labeling effort. In the Pol-SAR images classification scenario can be a very powerful tool. Taking into consideration all the problems related to the labels procurement phase as discussed in *Chapter* 1. Here below are described some possible methods for deploying proxy tasks:

- *Generation-Based Methods*: such as image inpainting [Pathak *et al.* 2016], image colorization [Zhang *et al.* 2016], or image generation [Goodfellow *et al.* 2014]. This group of methods applies a self-supervision by reducing the amount of information available, for example by removing the colors from an image. The network is trained to retrieve the missing information.
- *Context-Based Methods*: such as image jigsaw puzzle [Noroozi & Favaro 2016], geometric transformations [Gidaris *et al.* 2018], image clustering [Caron *et al.* 2018]. These types of pretext tasks employ context features of images such as spatial structures, shapes continuity, or context similarities to implement a self-supervision.
- *Free Semantic Label-Based Methods*: generate semantic labels using contour detection [Li *et al.* 2016], moving objects segmentation [Pathak *et al.* 2017], or game engines [Ren & Lee 2018] to generate data labels.

2.4 Convolutional Neural Networks

Convolutional Neural Network (CNN) is the most well-known and successful artificial neural network to deal with image data. It has shown significant results both in the field of remote sensing [Zhu *et al.* 2017b] and in the field of computer vision [Raghu & Schmidt 2020]. CNN architectures are characterized by many convolutional layers which extract features at different contextual levels. Convolutional layers are able to apply the same operation in different positions of the image. As a result, the network is able to learn strong and efficient features for image processing. For instance a CNN can autonomously learn *Gabor filters*, which is a traditional image processing technique [Khaleefah *et al.* 2020].

2.4.1 Convolution Operation

In mathematics, convolution is an operation that combines two functions x and w. The result of this operation is a third function which represents how the shape of x is modified by w. The convolution operation is expressed using the operator $*$, in the *Equation* 2.1 the basic formulation of the convolution is shown. This expression is valid for one-dimensional time-series.

$$f(t) = (x * w)(t) := \int_{-\infty}^{\infty} x(\tau)w(t-\tau)d\tau \tag{2.1}$$

In a convolutional neural network, the function x is called input, and the function w is called kernel or weights. The output can be referred to as a features map. In image processing, the input is a multidimensional array of data, and the kernel is a multidimensional array of parameters. The learning algorithm works to optimize the parameters tensor.
In image processing I is defined as a two-dimensional tensor (an image) and K is defined as a two-dimensional kernel that processes the image I. Unlike in the *Equation* 2.1, the operation in this case is defined over a finite discrete set of data. This is necessary because the computer works with discrete numbers and images have finite dimensions. The resulting operation is expressed by the *Equation* 2.2

$$F(i,j) = (I * K)(i,j) = \sum_m \sum_n I(m,n)K(i-m, j-n) \tag{2.2}$$

2.4.2 Convolution Motivation

Convolution operation in ANN has three main motivations:

- *Sparse interactions*: convolution operation is implemented using a small kernel with respect to the input image size. Even if an image is huge, analyzing a small part of it allows to extract meaningful features, such as edges. This can be performed using a small kernel. A small kernel has a lot of advantages: less parameters are stored, less memory is used and fewer operations need to be performed. As a result, the algorithm efficiency increases.

- *Parameter sharing*: means that each kernel (each parameters set) is used over the whole input. As a fact, the convolution passes the same filter/kernel over the whole image. This allows us to get better generalization capabilities instead of learning a specific parameter for each image position.

- *Equivariance*: is a direct consequence of parameter sharing. It consists in the fact that if the input changes, the output changes in the same way. This means that whether a feature is moved from a position to another in the input images, the output reports the same variation. In other words, the output excitation will be accordingly moved with respect to what happened in the input image.

2.4.3 Convolution Architecture

Typically, convolutional layers are composed of three parts, as it is possible to see in *Image* 2.7. The first operation performs several convolutions in parallel to produce a set of linear activations. The second one processes the output of the first applying a non-linear activation function, such as a sigmoid. The last stage applies a pooling function. A pooling function is an operation that replaces the actual values using summary statistics. For instance, max-pooling extracts the maximum value from a sliding window of a certain size. The window is passed through the whole image extracting the salient features. Convolution is not always applied in conjunction with an activation and a pooling layer, it can be used even alone or in conjunction with other operations.

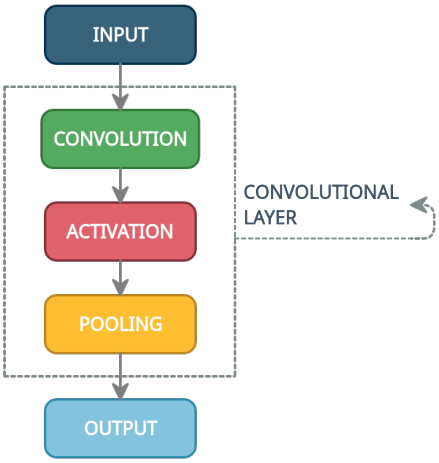


Figure 2.7: The image shows the detail of how a convolutional layer is composed.

2.5 Generative Adversarial Networks

Generative models are a branch of unsupervised machine learning techniques that aim to imitate any data distribution. They work opposite with respect to a common classification algorithm. They generate features modifying pseudorandom input data, instead of analyzing features to infer labels [Salimans *et al.* 2016]. GANs are feedforward ANNs composed of two contesting networks, in which one supervises the other in a zero-sum game. The concept was introduced in 2014 by [Goodfellow *et al.* 2014]. The final objective is to learn to mimic the input data distribution, to generate outputs that are new samples belonging to that data distribution. The discriminator is a classifier which states if images are real or fake. In *Figure* 2.8 it is possible to see the main elements of a GAN.
A common analogy, for GAN, is to consider one network as an art forger and the other as an art expert. The forger, known in the GAN literature as the generator, G, creates forgeries, intending to make realistic images. The expert, known as the discriminator, D, receives both forgeries and real images and tries to distinguish them. D and G are trained simultaneously, and in competition with each other. The generator has no direct access to real samples, the only way it learns is by interacting with the discriminator.

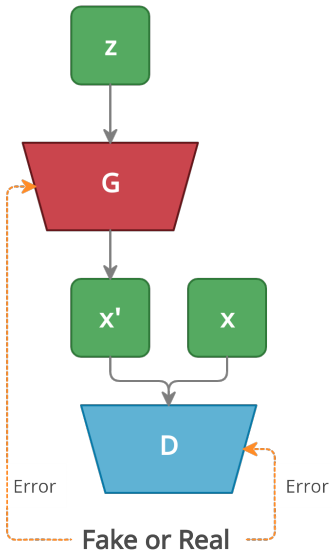


Figure 2.8: The scheme shows the main elements of a GAN. z is the input noise to the generator. G is the generator network. G is fed with z and it outputs x'. x' is the synthetic/generated image and it is inputted to the discriminator. D is the discriminator which is fed with x and x' and it outputs if samples are real or fake. Based on its decisions the network is optimized.

2.5.1 GAN Objective

Here below some terminology for GANs is introduced:

$$p_z \rightarrow \textit{ data distribution over noise input } z \tag{2.3}$$

$$p_g \rightarrow \textit{ generator distribution over data } x \tag{2.4}$$

$$p_r \rightarrow \textit{data distribution over real samples } x \tag{2.5}$$

$$\Theta_{\mathcal{D}} \rightarrow \mathcal{D} \textit{ parameter set} \tag{2.6}$$

$$\Theta_{\mathcal{G}} \rightarrow \mathcal{G} \textit{ parameter set} \tag{2.7}$$

$$\mathcal{D} : \mathcal{D}(x; \Theta_{\mathcal{G}}) \rightarrow (0, 1) \textit{ probability of the input sample } x \textit{ to be real or fake} \tag{2.8}$$

$$\mathcal{G} : \mathcal{G}(z; \Theta_{\mathcal{D}}) \rightarrow x \textit{ generated sample} \tag{2.9}$$

As introduced by [Goodfellow *et al.* 2014] the original GAN objective is:

$$\begin{aligned} \min_{\mathcal{G}} \max_{\mathcal{D}} \mathcal{V}(\mathcal{D}, \mathcal{G}) = \mathbb{E}_{x \sim p_r(x)}[\log \mathcal{D}(x)] \\ + \mathbb{E}_{z \sim p_z(z)}[\log(1 - \mathcal{D}(\mathcal{G}(z)))] \end{aligned} \tag{2.10}$$

The objective is to find the optimal parameter set for both G and D. The optimal D can be obtained maximizing its capacity to distinguish real samples $D(x)$ and maximizing the capacity to distinguish fake samples $1 - D(G(z))$. At the same time G is optimized to minimize the function $1 - D(G(z))$. This results in a min-max problem.

Given a fixed generator G, D^* is optimal when it is able assign probability one to data coming from p_{r} and to assign probability zero to generated samples belonging to p_g. It is possible to demonstrate that the optimal D^* for a fixed G is:

$$\mathcal{D}^* = \frac{p_r(x)}{p_r(x) + p_g(x)} \tag{2.11}$$

Now, finding the optimal generator $\mathcal{G}^*$ for the optimal discriminator $\mathcal{D}^*$ shown in *Equation* 2.11 it is possible to demonstrate that is:

$$\mathcal{V}(\mathcal{D}^*, \mathcal{G}^*) = 2\mathcal{JSD}(p_r \| p_g) - \log 4 \tag{2.12}$$

Where the $\mathcal{JSD}$ term is the Jensen-Shannon Divergence which is known to be optimal for $p_r = p_g = \frac{1}{2}$. Hence, as already mentioned, the optimality for the GANs is achieved when the discriminator is maximally confused between real and fake samples.

2.5.2 GAN Training

The training process of a GAN consists of training the discriminator D to be an optimal classifier on a fixed version of G. Then D is fixed and G is trained to its optimal. This optimization process is iterated back and forth. The optima is theoretically reached, when D can no longer distinguish between real and fake samples. D will output a fifty percent probability for every sample it encounters [Goodfellow *et al.* 2014]. In practice, the training of a GAN is quite complex because it is very unstable for the nature of the minimax game implemented in the optimization function. Nevertheless, generative adversarial networks have shown very interesting results in the last few years. The main reasons why GANs training is very challenging are:

- *Instabilities*: the GANs training is composed of two contesting networks which can start oscillating compromising the performances.
- *Mode collapse*: is a scenario in which only a few modes of the data distribution are generated, as a consequence the model produces only a limited variety of samples.
- *Difficulties in evaluation*: due to the lack of a robust stopping criteria, it is difficult to know when exactly the GANs training is finished.
- *Catastrophic forgetting*: is the tendency of an artificial neural network to completely and abruptly forget previously learned information upon learning new information.
- *Non-convergence*: the training procedure does not converge.
- *Vanishing gradient*: the gradient to train the networks tends to become smaller and smaller making the learning process unfeasible.

2.6 Conditional Generative Adversarial Networks

In the field of image processing, translating an image from a domain to another is a common task. For example, converting gray scale images to RGB images, or extracting edges from RGB images, is a usual task in computer vision. In the remote sensing field, the transformation of radar images to optical images is another quite typical image processing application.
GANs framework does not allow to control the modes in which images are generated. To overcome this limitation has been proposed an improved concept of the GANs. Instead of generating an image from noise, this class of algorithms is conditioned to the input. The input is conditioned feeding the network with additional informative content, which influences the generation of the corresponding outputs. This variation of the GANs is called Conditional Generative Adversarial Network (cGAN) [Mirza & Osindero 2014]. The objective is very similar to the original GANs, in which the final scope is to mimic a data distribution. The most important difference is that both discriminator and generator in cGANs are conditioned with additional information. This makes this concept supervised. In fact, to train these ANNs it is necessary to have paired couples of input and output data, in *Figure* 2.9 the main elements of cGANs are presented.
[Isola *et al.* 2017a] have proposed an interesting work that has exploited this technique. This paper demonstrates that cGANs can be applied to a wide variety of problems. They also provide the framework used to achieve their results. cGANs have been successfully employed even in the field of remote sensing [Ley *et al.* 2018]. One of the main limitations of cGANs is the need for paired images. This means that the same version of an image is required. In the scenario of black and white images, the procurement of a paired training sets is simple, because there are many colored images that can be used. But in other application scenarios, such as radar to optical images transcoding, aligning optical and radar images is not a trivial task [Merkle *et al.* 2017].

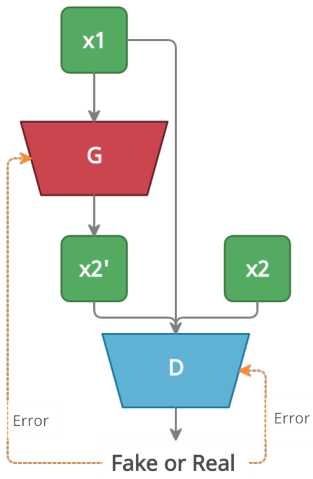


Figure 2.9: The scheme shows the main elements of a cGAN. $x1$ is the input sample from one data distribution (1) given to the generator. G is the generator network fed with $x1$ that outputs $x2'$. $x2'$ is the synthetic/generated image that should belong to the second data distribution (2) and it is inputted to the discriminator. D is the discriminator which is fed with $x1$, $x2$ and $x2'$ and it outputs if samples are real or fake. Based on its decisions the network is optimized.

2.7 Cycle Consistent Generative Adversarial Networks

A further step forward in image generation has been done introducing the so-called Cycle-GAN, or image to image unpaired transcoder [Zhu *et al.* 2017a]. This network is very similar to the work of [Isola *et al.* 2017a]. It brings the same capacity of converting images from a domain to another, but this technique eliminates the constraint of having paired images. The relaxation of having an unpaired dataset makes this formulation very powerful, the same method can be used to tackle a variety of problems. Besides, data procurement is much easier since it is not necessary to have a direct relation between samples in the datasets. The main feature of Cycle-GANs lies in being able to learn transformations without one-to-one mapping between the source domain and the target domain.
The need for paired images is eliminated by making a two-step transformation. Data are firstly mapped to the target domain and then mapped back to the source domain. The objective function forces the original image and the reconstructed one to be as consistent as possible between each other. The loss function of the Cycle-GANs is composed of two terms: the above-mentioned one, and an adversarial term which is based on the capacity of the generator to fool a discriminator. For each domain is implemented a specifically designed discriminator. Cycle-GAN framework is composed of two generator-discriminator couples, one for each domain. The four ANNs interact at each optimization step. Each training sample flows through the first generator. The output is a transformed version from one domain to the other. Then the output is passed through the second generator to be transformed back to the original domain. In the end, the generator is optimized by comparing the similarity between the reconstructed image and the input original sample, and based on the capacity to fool the discriminator with the transformed version of the input sample. This process is performed for both the generators. Simultaneously, the discriminators are trained to recognize fake and real samples. This process involves four different ANNs and requires to correctly apply two transformations to the input image. As a result, this framework is much more complex with respect to the cGAN. In practice the capacity to translate images without using paired datasets introduce a complexity that is not negligible. Finally, in *Figure* 2.10 the main elements of cGANs are presented.

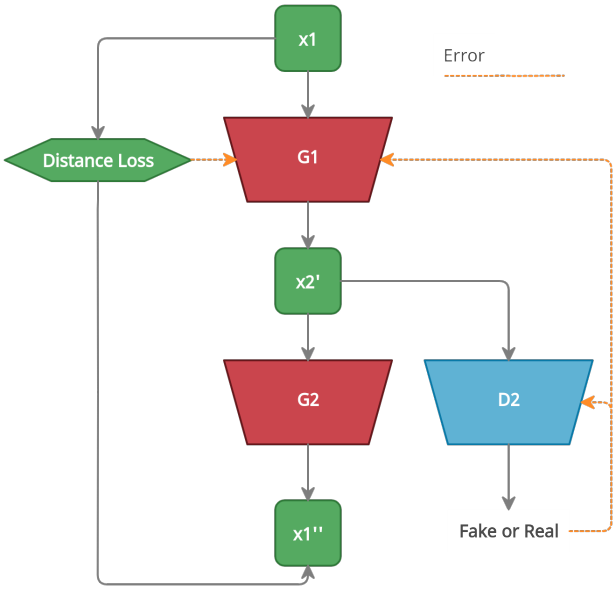


Figure 2.10: The figure shows the main elements for training the Cycle-GANs. $x1$ is the input sample from data distribution (1) given to the generator $G1$. There are two generators $G1$ and $G2$. $G1$ converts samples belonging to data distribution (1) to data distribution (2). $G2$ converts samples belonging to data distribution (2) to data distribution (1). $x2'$ is a synthetic sample that is given to $G2$. $G2$ converts $x2'$ back to domain (1). $G2$ outputs $x1''$, which is the reconstructed sample used to generate the cycle loss comparing it with the input $x1$. $D2$ is a discriminator trained to recognize if samples coming from data distribution (2) are real or fake. Based on the $D2$ output $G2$ parameters are optimized. This is the so-called adversarial loss. This image represents only the process related to the training of $G1$, but it is necessary to perform the same operations for $G2$. $G2$ requires a cyclic loss between $x2$ and $x2''$. Besides an additional discriminator, called $D1$, used to recognize if samples coming from (2) are real or fake.

2.8 Radar Images Classification

Radar images classification can be performed using parametric approaches. Parametric approaches rely on assumptions and try to classify images by describing specific features and modeling data statistical properties. Some examples are: generalized gaussian [Moser *et al.* 2006] or ginite mixtures [Moser *et al.* 2010]. These are the firstly implemented methods and have some limitations related to the required assumptions. In general, they tend to fail when applied to modern high-resolution images.
Then, another technique is to employ general purposes classifiers which are composed of two steps. They firstly extract hand-crafted features, then they apply a classifier to the extracted features. Some examples of these methods are: Multilayer Perceptrons (MLP) [Bruzzone *et al.* 2004] or Random Forest (RF) [Hänsch & Hellwich 2010]. This set of methods is more recent and has better generalization capabilities. They have two main limitations: they are based on handcrafted features, and such classifiers are not easily extendable to complex data, which is the case of Pol-SAR data. As a result, these approaches require a projection of complex-valued Pol-SAR data to the real domain. Typical techniques to project complex-valued data into real features are: polarimetric decompositions or feature extraction methods especially designed for Pol-SAR images. Both these methods introduce a dependency on the classification results to these projections. Applying machine learning algorithms, directly on complex-valued data, allows us to exploit the features extraction process in a very generalized way. The optimization process working directly on raw data does not make any

assumption or apply any synthetic pattern to data [Hänsch 2010, Hua *et al.* 2019]. This last approach has still a significant disadvantage, being a supervised learning algorithm it requires a lot of labeled data. In particular, deep learning, which in the field of images classification has shown very good results, requires an even larger amount of labeled data to perform well [Raghu & Schmidt 2020]. This means that this improvement comes at a price.

2.9 Related works

Convolutional artificial neural networks are the state-of-the-art in remote sensing images classification. The work of [Hjelm *et al.* 2019] offers an exhaustive review of remote sensing techniques. This thesis concentrates on Polarimetric Synthetic Aperture Radar images semantic segmentation. Typically, this task is composed of a manual features extraction phase and a classifier trained on the handcrafted features. Deep learning is a technique that automatically extracts the required features [Goodfellow *et al.* 2016]. Manual features extraction techniques cause information loss. Instead, Deep Artificial Neural Network (DANN)s, directly working on raw data, exploit all the information coming from sensors. Moreover, manually extracted features provide less generalization and often introduce dependency to the extraction methods or to the initial assumptions [Hänsch & Hellwich 2017].
Deep learning directly working on raw data can avoid all the above-mentioned drawbacks. DL needs a large amount of data to perform well. Pol-SAR images annotation is a costly and time-consuming task. As a result, the procurement of annotated data in this scenario is challenging. In this work, to overcome the mentioned issue, the training of the algorithms is performed using a combination of supervised and unsupervised techniques. Semi-supervised learning is a technique which involves the two above-motioned. Self-supervised learning is a subset of semi-supervised learning methods. It is a technique that extracts features by leveraging unlabeled data. Unlike pure unsupervised learning, the self-supervision automatically generates labels and learns to predict them.
Self-supervision is a topic largely studied in computer vision. For example, [Jenni & Favaro 2018] propose to pre-train an ANN, learning to distinguish between real images, and images altered with synthetic artifacts. They state that an artifacts discriminator can learn strong representative features for object classification, detection, and segmentation. To train such a discriminator they propose a framework that generates altered images. They modify the input images at the feature level. The discriminator network is forced to output the probability of a sample to be real or altered, and a map that locates which areas have been altered. The proposed architecture is made of two parallel pre-trained autoencoders, which are used to process the images. One is employed to encode and decode the original images, the other performs the same operation but it introduces the alteration. The alteration is applied after the encoding phase. The position of the altered areas is propagated to the last layer to be employed as label.
The authors set up their experiments using unlabeled data to perform the pre-training task, and the labeled part to transfer the knowledge to the downstream task. The authors state that the classification network, based on the pre-trained features, achieves state of the art performances on several benchmarks for self-supervised learning in the field of computer vision. They compare their results even with a supervised framework reaching comparable results. They state that one of the main limitations of their work is that the comparisons they propose are based on different models. The results they achieve could be related both to the different architectures or to the pre-training strategy. By the way, this work is a clear example of how the self-supervision can boost the capacity of the downstream task, in particular in situations where there are few labeled data.
Another example that confirms the potential of self-supervised learning is presented in the work of [Singh *et al.* 2018]. In this work, the authors apply the concept of self-supervision to overhead imageries. They state that overhead imageries annotation is quite difficult and requires experts. To overcome this problem

they propose to pre-train a network using image inpainting as pretext task to learn representative features. The idea proposed is to train a *coach* network, which predicts increasingly difficult masks (empty areas), and an inpainting network trained to fill these gaps by analyzing the context of the image. The former is based on the Residual Network (ResNet) architecture ([He *et al.* 2016]) and is trained with the adversarial loss trying to output always more difficult masks. The latter is an encoder-decoder network based on the ResNet architecture. The authors state that one of the main limitations observed is that their framework does not exploit the large amount of unlabeled data. In fact, they show that the performances of self-supervised pre-training remains mostly the same, despite a significant increase in the number of unlabeled images used during the pre-training phase. On the other hand, they achieve good performances in different classification tasks, such as semantic segmentation and road extraction. To fine-tune their algorithm they use only 10% of labeled data. Besides, they highlight the strong capacity of the ResNet architecture implemented.
The idea to apply the self-supervised paradigm to extract features from radar images has been exploited in some works: [Ley *et al.* 2018, Saha *et al.* 2020]. This thesis tries to employ Pol-SAR to optical transcoding as self-supervised task to extract features for semantic segmentation.
[Ley *et al.* 2018] propose the same paradigm. The authors implement Pol-SAR to optical transcoding as proxy task. They develop a conditioned generative adversarial network that processes Pol-SAR images and transforms them into optical ones. The authors state that this support task allows to learn useful features for polarimetric synthetic aperture radar images classification. The algorithm has to recognize shapes and textures to correctly convert radar images to optical ones. They compare three supervised algorithms versus a self-supervised one. The three supervised algorithms are a random forest classification algorithm, a U-Net classification algorithm, and the ANN architecture used to transcode images. These three benchmark implementations are trained in a supervised fashion using annotated data. [Ley *et al.* 2018] have developed a dedicated architecture both for the generator and discriminator. The two networks have been trained in a generative adversarial fashion to perform the task of image transcoding. The self-supervision task is performed using paired radar and optical data, since the conditioning requires coupled images.
This work achieves very interesting results. The pre-trained algorithm is the strongest classifier with respect to the U-Net the random forest and the generator supervised version. It has shown an overall accuracy of over 80%. All the supervised versions have achieved less than 70%. Besides, the authors have studied the sensitivity of the algorithms to the amount of labeled data. The self-supervised version, fine-tuned with an amount of labeled data minor than 10%, loses only a few accuracy percentage points. The accuracy of the supervised counterparts breaks down to less than 50%.
The main limitations in this work are intrinsically related to the conditioned adversarial neural network. This training paradigm requires two equal versions of the same images, so basically the transcoding phase is supervised. This work has demonstrated that semantic segmentation can be performed even if the amount of annotated data is significantly low. On the other hand, it remains the complexity of the procurement of aligned Pol-SAR and optical images. Another limitation of this work is related to the features extraction method: the algorithm is not directly applied to the complex version of the Pol-SAR data, but there is a preprocessing step in which a real features vector is extracted.
The transcoding from Pol-SAR to optical images using unpaired datasets allows us to take advantage of much more data. The alignment of radar and optical images is not a trivial task. Radar to optical images transcoding is a common application in literature. In some cases it is specifically used as a features extractor for the target task, as proposed by [Saha *et al.* 2020]. In this work, the authors implement a transcoder using a Cycle-GAN, then they use the features extracted from the transcoder to perform change detection analysis. They state to use the Cycle-GAN because of its capability to work with spatially uncoupled images.
Another application quite spread in literature uses to transcode radar to optical images with the purpose to make radar products more accessible. For example, an optical image can be interpreted by non trained experts, unlike a radar one. In general, the ability to transform active sensor images to passive sensor images can be a way to

enlarge the possibilities for radar images.
Recently some works have tried to implement the radar to optical transcoding using the cycle consistent adversarial loss. This work [Wang *et al.* 2019] implements a Cycle-GAN to transcode images from SAR to optical images to better interpret them. The work of [Reyes *et al.* 2019] implements a cycle consistent adversarial transcoder to exploit the potentiality of radar to optical transcoding. They analyze the network capacity in terms of interpretation (capacity to simplify the understanding of original radar images), despeckling capabilities, and features extraction. To test the features extraction capabilities they implement a road extraction algorithm. Road extraction is different from semantic segmentation, but it is comparable, since semantic segmentation can also recognize roads. To perform road extraction, [Reyes *et al.* 2019] train a fully-convolutional neural network. This network is trained to classify raw SAR images and transcoded SAR images. The two classifiers are compared to understand how good the transcoded images are with respect to directly analysis of SAR iamges. The results are not very encouraging, since the direct process of SAR images shows better results. The authors state that translating images from radar to optical can cause the loss of sharp details, reducing the amount of information. This kind of information is very important in road extraction. Nevertheless, [Reyes *et al.* 2019] state that the transcoding capacity is promising. The authors specify that they noticed some transcoding limitations mainly related to urban areas. They state that, although the network can transcode independent crop fields, it shows difficulties to isolate buildings in large cities, where these can be blurred or merged into larger elements. With respect to the implementation of this thesis, there is a fundamental difference. In this thesis, the transcoder extracts features from Pol-SAR images and does not analyze the features contained in the transcoded images. The purpose of this work is to leverage the features extraction capacity of the transcoder, not the quality of the features contained in a transcoded optical image.
In this thesis, the final objective starts trying to demonstrate the capacity of the Pol-SAR to optical transcoding to be employed as the proxy task for image semantic segmentation. The scope is to confirm that it is possible to train a good classifier even in a scenario in which few labeled data are available. This is strongly motivated by the fact that Pol-SAR image labels procurement is a very costly task. Then, this work aims to employ different concepts for training the transcoder. The idea is to implement a transcoder based on a regressive loss, one based on a conditional adversarial loss, and one based on a cycle consistent adversarial loss. These three concepts are of increasing complexity. Besides, the implementation of the cycle consistent adversarial transcoder should exploit the capacity of this training paradigm to learn from unpaired datasets. The comparison of different training strategies helps to understand better the capacity of Pol-SAR to optical images transcoding used as pretext task for semantic segmentation.
The expectations are that the regressor transcoder can highlight good semantic segmentation performances with reduced complexity. The conditional adversarial transcoder should be able to perform better, but it requires a more complex framework. The cycle consistent adversarial transcoder, instead, could show competitive capacity. Moreover, it should empower the potentiality to learn from unpaired datasets. Finally, the last contribution of this thesis is the investigation of transcoders hyperparameters, such as architecture deepness and patch size.

CHAPTER 3

Dataset

Contents

3.1 Data for Transcoding 23
3.2 Data for Classification 25
3.3 Data Splitting 25
3.4 Data Preparation 26

3.1 Data for Transcoding

The data used for transcoding are Sentinel-1 (SEN-1) [esa.int a] and Sentinel-2 (SEN-2) [esa.int b] images. According to [esa.int a], SEN-1 is a constellation of two polar-orbiting satellites, which performs C-band SAR imaging and operates in dual polarization (HH+HV, VV+VH). Instead, SEN-2, according to [esa.int b], mounts a high-resolution, multi-spectral imaging sensor. SEN-2 is composed of a constellation of two polar-orbiting satellites. SEN-2 orbit is sun-synchronous. In this thesis, the data are the same used in the work of [Ley *et al.* 2018]. The resolution of both radar and optical data is ten meters. Data were preprocessed with ESA's Sentinel Application Platform (SNAP) software. Images come from an area with the following characteristics: minimal cloud coverage, image acquired at close dates, large spatial overlap between SAR and optical images. Optical images are composed of these four bands:

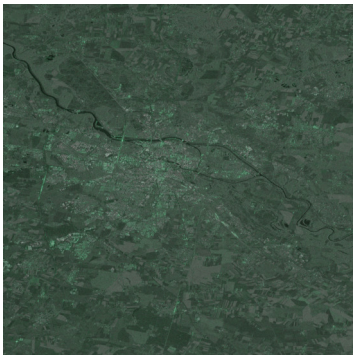
- *Red (R)*: Sentinel-2 band 4.
- *Green (G)*: Sentinel-2 band 3.
- *Blue (B)*: Sentinel-2 band 2.
- *Near-Infrared (NIR)*: Sentinel-2 band 8.

Pol-SAR data are Dual-Polarimetric single-look complex products. According to [Ley *et al.* 2018], Pol-SAR data were processed using standard Sentinel-1 pre-processing (e.g. TOPS deburstsing). Dual-polarimetric covariance matrices were computed from radar data to take advantage of polarimetric amplitude and phase information. Next, data were geo-coded based on SNAP's Digital Elevation Model (DEM) based Range-Doppler Terrain Correction step. The geocoding tool was applied in the original paper to align optical and radar images. From the raw Pol-SAR data have been calculated the covariance matrix C from where a five features vector v has been extracted. This vector represents the radar data and it is used both for transcoding and semantic segmentation. This vector is composed of the logarithm of the two diagonal elements and the magnitude of one of the off-diagonal elements. Then the last two elements represent the phase information. These two features are respectively the normalized real and imaginary part of one of the off-diagonal elements. The result is a real value vector as the one shown in the *Equation* 3.1.

$$v = \begin{pmatrix} log(C_{1,1}) \\ log(C_{2,2}) \\ log(|C_{1,2}|) \\ \frac{\mathbb{R}(C_{1,2})}{|(C_{1,2})|} \\ \frac{\mathbb{I}(C_{1,2})}{|(C_{1,2})|} \end{pmatrix} \tag{3.1}$$

All the images, both optical and radar, have been normalized before being used. The normalization has been applied separately to each of the four optical channels and to each of the five features of the vector v. The normalization used is a transformation to zero mean and unit variance.
In *Figure* 3.1 *a)* and *b)* is possible to see a sample data generated using the optical and radar data employed in this thesis. The *Figure* 3.1 *a)* represents a sample of the RGB channels of the optical data. It is a portion of the optical train tile shown in *Appendix* B.2. The *Figure* 3.1 *b)* represents a samples of the three first elements of the vector v (presented in *Equation* 3.1) visualized as an RGB image. This image is a portion of the Pol-SAR train tile shown in *Appendix* B.1.

a) Optical image over the area of Poznan.



b) Pol-SAR image over the area of Poznan.

Figure 3.1: *Figure a)* and *Figure b)* show a sample from the training data tiles.

3.2 Data for Classification

The data presented in the precedent sections are accompanied with labels. These annotations were manually created using the optical images. These semantic segmentation maps are composed of five classes:

- Band 0 is *Forests*.
- Band 1 is *Streets*.
- Band 2 is *Fields*.
- Band 3 is *Urban*.
- Band 4 is *Water*.

Labels are stored in a five channels vector and are one-hot encoded. The train data set is composed of around 2 million labeled pixels. The test dataset is composed of roughly 1.2 million labeled pixels. In *Table* 3.1 is possible to see the exact number of classified pixels per each class for the train and the test sets, respectively. As is possible to see the classes are balanced between the two sets of data, between classes instead the *Highways* has few occurrences with respect to the others. In *Figure* 3.2 is possible to see a sample of the label map train tile shown in *Appendix* B.3.

Class Name	Number of Classified Pixels Train Set [k]	Number of Classified Pixels Test Set [k]
Forests	27	28
Streets	635	414
Fields	691	196
Urban	166	273
Water	577	376
Total	2096	1287

Table 3.1: The table shows the number of labeled pixels for each class in the dataset. The top column shows one next to the other the number of classified pixels in the train and test sets. In the last row the total number of labelled pixels is presented.

3.3 Data Splitting

The training phase has been performed using an image coming from the area of Poznan (Poland). The optical data have been acquired on 30/07/2017, the Pol-SAR data have been acquired on 29/07/2017. The test phase has been performed using an image coming from the area of Wroclaw (Poland). The optical data have been acquired on 30/07/2017, the Pol-SAR data have been acquired on 24/06/2017. In *Appendixes* B.1 and B.4 the original Pol-SAR tiles, used for training and testing, are shown. In *Appendix* B.2 and B.5 the original optical tiles, used for training and testing, are shown. In *Section* B.3 and B.6 the original label map tiles, used for training and testing, are shown.

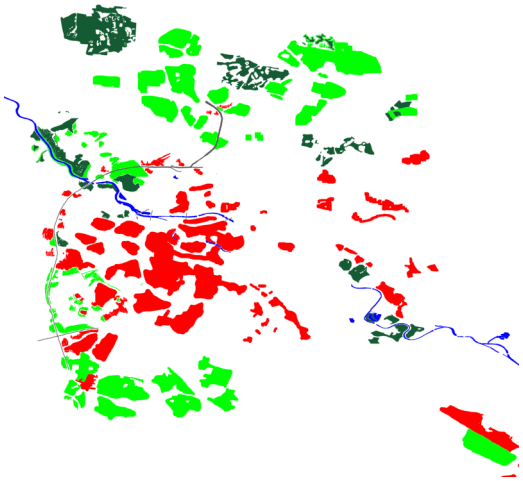


Figure 3.2: The image shows a sample from the label map train tile. Dark Green areas represent *Forests*, grey areas represent *Streets*, light green areas represent *Fields*, red areas represent *Urban*, blue areas represent *Water*.

3.4 Data Preparation

After the above-mentioned preprocessing operations, data tiles have been cut into patches. Each tile has been cut into patches of size that varies from 128×128 pixels to 256×256 pixels. The patches generated during the cutting process have been created using an overlap of the 50%. This means that information is redundant inside patches. Each patch has some information present even in other patches. This operation helps the network to process better the images. Without image overlapping, an element across two patches is cut in half. This could make the recognition of the targets more complex. Besides, when the prediction maps or the transcoded images are reconstructed, the use of overlapping is a possible solution to smooth the edges between different patches. If bordering patches contain completely different images, it is more likely that the reconstructed image will be affected by patch edges artifacts. Unlike using patch overlapping, the ANN receptive fields process two neighboring patches with common information. As a result, it is possible to better combine patches achieving smoother image compositions. These considerations are valid for borderline cases. In *Table* 3.2 is presented the resulting number of patches for each dataset employed in this thesis.

	Patch Size	**Number of Patches for Transcoding**	**Number of Patches for Classification**
Train	128	24702	2730
Test	128	25772	2340
Train	256	6341	1116
Test	256	6606	1012

Table 3.2: The table shows the number of samples for every dataset configuration.

Chapter 4

Methodology

Contents

4.1 Network Architectures **27**
4.1.1 Generator 27
4.1.2 Discriminator 30
4.1.3 Classifier 31
4.1.4 Implementation Details 32
4.2 Networks Training **32**
4.2.1 Baseline 33
4.2.2 Regressor Transcoder 34
4.2.3 Conditional Adversarial Transcoder 35
4.2.4 Cycle Consistent Adversarial Transcoder 37
4.2.5 Fine-Tuning 39
4.2.6 Networks Complexity 39

4.1 Network Architectures

This thesis is conceptually based on the work of [Ley *et al.* 2018]. Unlike the implementation code is based on the work of [Zhu *et al.* 2017a, Isola *et al.* 2017a], for which the code is freely available [Isola *et al.* 2017b]. Based on these two works, a new framework for transcoding and classifying images has been developed. The code is available here [Cattoi 2021]. To perform the images transcoding task, three different strategies have been implemented. These strategies are deeply discussed in the *Subsections* 4.2.2, 4.2.3, 4.2.4 . All the implementations are based on the same network architecture, deeply described in the next *Subsections*. These three implementations mainly differ in the training strategy.

4.1.1 Generator

The generator architecture is based on the Cycle-GAN architecture [Zhu *et al.* 2017a, Isola *et al.* 2017b]. The authors have based their ANN design on [Johnson *et al.* 2016]. This architecture is composed of three macroblocks: a couple of convolutional downsampling layers, several residual blocks and a couple of convolutional upsampling layers. The rationale behind this architecture is related to two main reasons. The downsampling-upsampling setup leverages the capacity of the convolutional layers to capture features coming from different receptive fields. The capacity to capture features coming from smaller and larger receptive fields allows the algorithm to extract different levels of features complexity. Each downsampling-upsampling operation introduces a factor of two to the dimensions of the receptive fields. The purpose of the residual blocks, in the body of the network, is to preserve low-level characteristics of the images processed. According to [He *et al.* 2016] residual blocks have the capacity to learn the input image identity function. Placing several residual blocks in

the backbone of the network allows to directly transfer some input images structures to the output. Since in images transformation input and output share some characteristics, residual blocks can be an effective strategy to employ [Johnson *et al.* 2016, He *et al.* 2016].

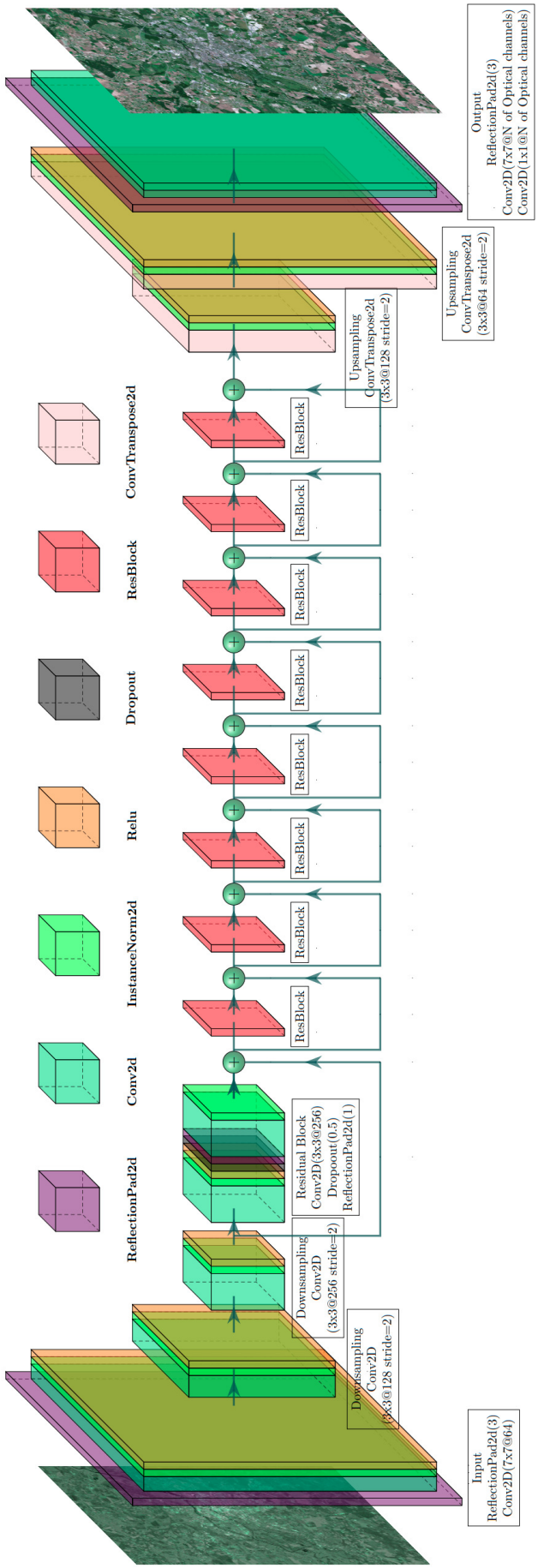


Figure 4.1: The image shows the architecture of the generator.

4.1.2 Discriminator

The discriminator architecture is implemented following the work of [Zhu *et al.* 2017a, Isola *et al.* 2017a]. [Isola *et al.* 2017a] state that the discriminator loss should only model high-frequency structures. As a fact, the Cycle-GAN relies on the cycle consistent regression term to model low-frequencies. To model high-frequencies, [Isola *et al.* 2017a] design a discriminator architecture that penalizes structure at the scale of patches. The discriminator applies several convolutional filters across the image. The extracted information is combined using the last convolutional layer. The discriminator output is represented by a map that states if an image is real or fake. The discriminator output is not a simple *true or false* bit. In fact, [Zhu *et al.* 2017a] affirm that employing a confidentiality map as output of the discriminator is more stable and allows to achieve better results. The informative content enclosed in a confidentiality map is higher than the one enclosed in a single pixel. As a result a richer output increases the discriminator capacity.

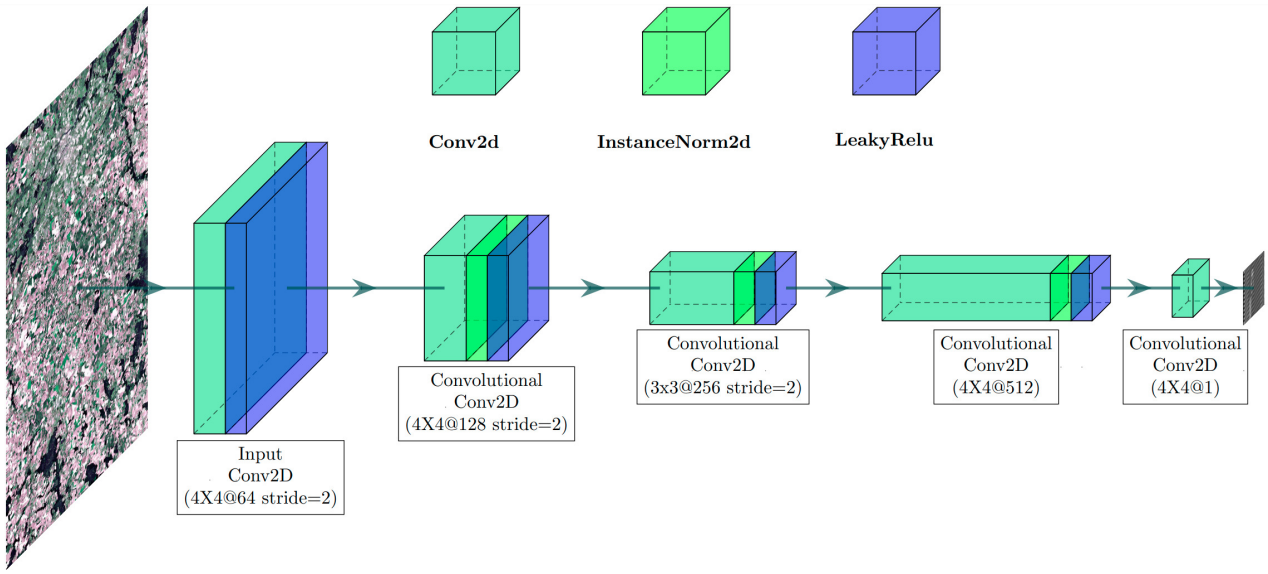


Figure 4.2: The image shows the architecture of the Discriminator.

4.1.3 Classifier

To perform the semantic segmentation, the generator architecture presented in *Subsection* 4.1.1 is employed. To implement the classification step, part of the generator weights are loaded. In *Figure* 4.1 is possible to see the network configuration used to perform the semantic segmentation task. The features in the *Input* block, in the *Downsampling* blocks, and in the *Residual Blocks*, learned during the pre-training/transcoding phase, are restored. Then, the two blocks called *Upsampling* and the block called *Output* are retrained. The last three layers are trained from scratch in a supervised fashion. This step is performed to adapt the features learned performing the transcoding task to the classification task. The greyed blocks are the pre-trained layers, the colored blocks are the layers retrained to perform the semantic segmentation task. To summarize, the semantic segmentation is performed using the generator architecture presented in *Subsection* 4.1.1 with the last layers retrained to perform the downstream task.
In this work, the semantic segmentation phase has not only been performed using a pre-trained network with the last layers retrained. In fact, has been implemented a so-called Baseline (BL) which performs semantic segmentation using the generator architecture presented in *Subsection* 4.1.1. The BL is trained using the generator architecture from scratch. Unlike the pre-trained version, the BL trains all layers from scratch. This means that both greyed and colored blocks are trained. The training is performed in a supervised fashion using Pol-SAR images and the semantic segmentation labels as for the pre-trained models.

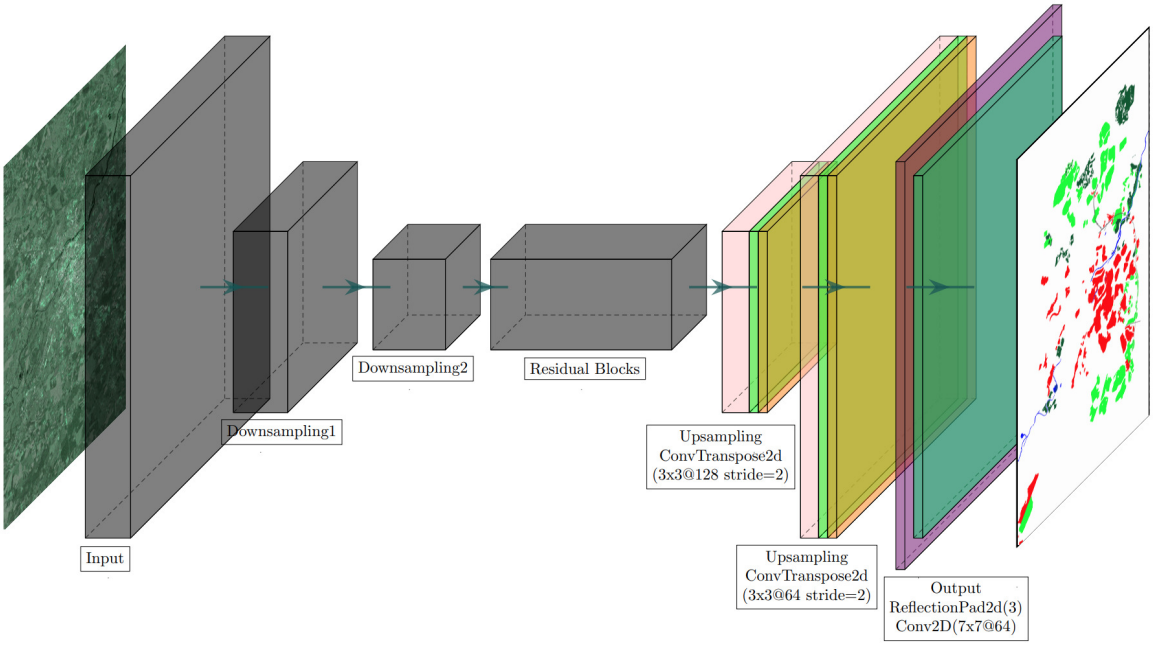


Figure 4.3: The image shows the generator architecture of *Figure* 4.1. The greyed blocks are fixed during the classification phase of pre-trained models. The remaining colored layers are the ones retrained during the fine-tuning phase.

4.1.4 Implementation Details

All the codes developed in this thesis are written in *Python*. To perform the machine learning algorithms, the *Pytorch* library has been used [Paszke *et al.* 2019]. All the blocks shown in *Figure* 4.1 and 4.2 are directly implemented in this library.

- *ReflectionPad2d (violet)*: pads the input tensor.
- *Conv2d (aqua)*: implements the convolution mathematical operation to two-dimensional inputs.
- *InstanceNorm2d (lime)*: implements the instance normalization proposed by [Ulyanov *et al.* 2017].
- *Relu (orange)*: applies the rectified linear unit function element-wise. This activation function returns zero for any input value below zero. For a value above zero, it returns its identity.
- *LeakyRelu (blue)*: is an updated version of the *ReLu*. Instead of returning zeros for values below zero, it returns the value multiplied by a coefficient.
- *Dropout (grey)*: randomly drops some values from the input tensor based on a probability threshold.
- *ConvTranspose2d (pink)*: is an upsampling block which applies a convolutional operation to the upsampled version of the input. It is a typical block used in the decoding phase.

4.2 Networks Training

Network training is the phase in which the algorithms learn to perform the task for which they have been designed. It is the most challenging task in ANNs developing. In fact optimization algorithms require a lot of time to be executed. In this *Section* are discussed the details of the training phases of the algorithms implemented. As previously discussed, a self-supervised features extraction method has been developed. The self-supervised task consists of transcoding images from the Pol-SAR to the optical domain. To perform this task have been developed three different features extractors. Then, to the self-supervised extracted features is applied a fine-tuning phase to classify the Pol-SAR data. The above mentioned steps consist in two distinct training phases, the former is the transcoding training phase and is explained for each implementation in *Subsections* 4.2.2 4.2.3 and 4.2.4. The latter is the fine-tuning training phase and is described in *Subsection* 4.2.5. In addition, to perform the semantic segmentation task has been trained a baseline to be used as a benchmark. The baseline training is presented in *Subsection* 4.2.1.

4.2.1 Baseline

The so-called BL is a supervised machine learning algorithm. In this setup, there is a generator network, the one in *Figure* 4.1, trained using the labeled data. The BL is trained using a *cross-entropy* loss and the *Adam* optimizer (both implemented in the *Pytorch* library). This network is used as a reference to show the capacity of the pre-trained architecture with respect to an equivalent supervised framework. In *Figure* 4.4 there is a scheme which shows the functioning of the BL, and here below is presented the loss function equation and the objective function equation for the optimal BL*.

$$\mathcal{L}_{CE}(BL) = \mathbb{E}_{x \sim p(x)}[-\log q(x)] \tag{4.1}$$

The loss function is characterized by the cross entropy between the softmax of the predictions and the labels. In the formula: $p(x)$ is the true labels distribution and $q(x)$ is the softmax probability approximated by the model.

$$BL^* = \arg\min_{BL} \mathcal{L}_{CE}(BL) \tag{4.2}$$

The BL parameters are optimized minimizing the L_{CE}.

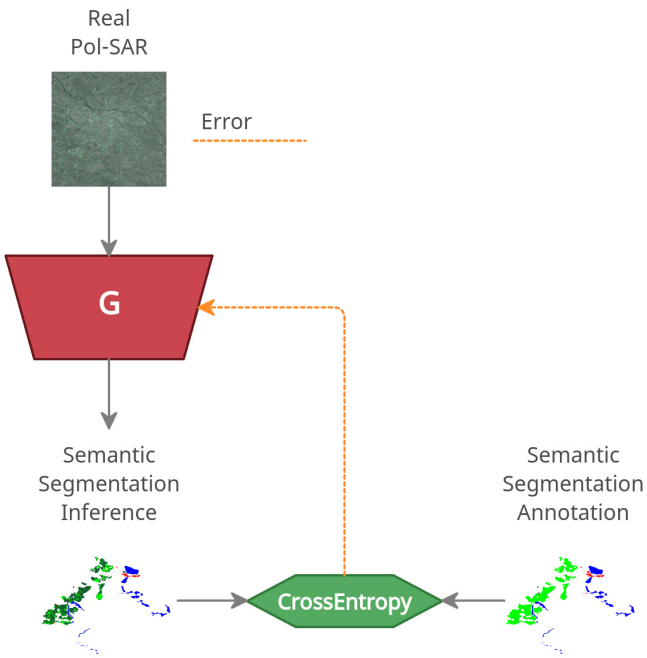


Figure 4.4: The image illustrates how the BL is trained to perform the semantic segmentation task. There is a Pol-SAR image in input to a generic generator G. The generator parameters are optimized based on the error between the generated map and the semantic segmentation annotation. The classification error is calculated using the cross entropy.

4.2.2 Regressor Transcoder

The Regressive Transcoder (RT) is the simplest transcoder implemented. The architecture is the one shown in *Figure* 4.1. The RT network learns to map input images x to output images y, $RT : x \rightarrow y$. The optimization process of RT is based on the $L1$ *norm* between the inferences of the $RT(x)$ and the real optical images y. This network is a fully supervised algorithm. The implementation is made using the *Adam* optimizer [Kingma & Ba 2015] and *L1Loss* function of the *Pytorch* library. In *Figure* 4.5 there is a scheme which shows the functioning of the RT, and here below are presented the loss function equation and the objective function equation for the optimal RT*.

$$\mathcal{L}_{L1}(RT) = \mathbb{E}_{x,y}\left[\|y - RT(x)\|_1\right] \tag{4.3}$$

In the equation above is described the loss function.

$$RT^* = \arg\min_{RT} \mathcal{L}_{L1}(RT) \tag{4.4}$$

The learning process tries to minimize the loss function updating the RT parameters.

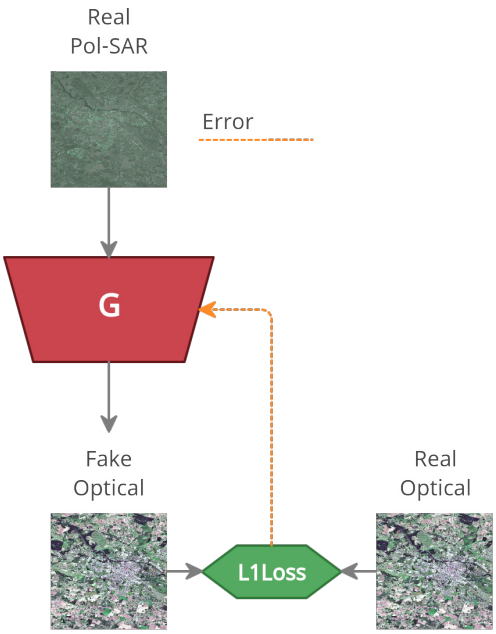


Figure 4.5: The image illustrates how the regressive transcoder is trained. There is a Pol-SAR image inputted to a generic generator G. The generator parameters are optimized with respect to the error between the generated optical image and the real optical image. The difference between the two images is calculated using the *L1Loss*.

4.2.3 Conditional Adversarial Transcoder

The Conditional Adversarial Transcoder (CAT) is the transcoder implemented using the conditional adversarial setup. This means that this network is trained using a GAN conditioned to the input, in this case Pol-SAR images. In this setup two networks are involved: a discriminator D implemented as in *Figure* 4.2, and a generator G implemented as in in *Figure* 4.1. The network G learns to map from input x to y, $G : x, z \rightarrow y$. The implementation and the training strategy is inspired to [Isola *et al.* 2017a]. The transcoder G is trained in contrast with the discriminator D, G tries to fool D and D is continuously trained to recognize G synthetic samples. Both networks are trained in a supervised fashion. The optimization is based on two metrics: the $L1$ distance between real and synthetic images, plus the mean square error of D to detect generated samples. The discriminator D is trained on the mean square error between the capacity to recognize, input conditioned, real and fake samples. The optimization process is performed using the *Adam* function of the *Pytorch* library. In *Figure* 4.6 there is a scheme which shows the functioning of the CAT, and here below are presented the loss function equations and the objective function equation for the optimal G^*.

$$\begin{aligned}\mathcal{L}_{cADV}(\mathcal{G},\mathcal{D}) = \mathbb{E}_{x,y}[\log \mathcal{D}(x,y)] \\ + \mathbb{E}_{x,z}[\log(1-\mathcal{D}(x,\mathcal{G}(x,z))]\end{aligned} \tag{4.5}$$

The conditioned adversarial loss is represented in the formulation above.

$$\mathcal{L}_{L1}(\mathcal{G}) = \mathbb{E}_{x,y,z}\left[\|y-\mathcal{G}(x,z)\|_1\right] \tag{4.6}$$

In the formulation above is shown the $L1$ loss term.

$$\mathcal{L}_{CAT}(\mathcal{G},\mathcal{D}) = \mathcal{L}_{cADV}(\mathcal{G},\mathcal{D}) + \mathcal{L}_{L1}(\mathcal{G}) \tag{4.7}$$

The final loss function is made by the summation of *Equations* 4.5 and 4.6.

$$\mathcal{G}^* = \arg\min_{\mathcal{G}}\max_{\mathcal{D}} \mathcal{L}_{CAT}(\mathcal{G}) \tag{4.8}$$

Here above the objective function for the optimal $\mathcal{G}^*$.

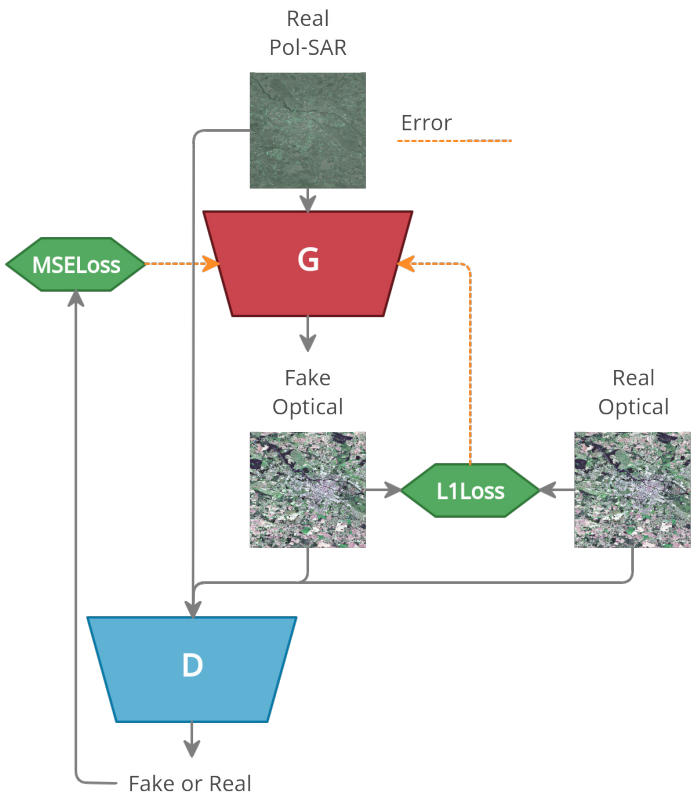


Figure 4.6: The scheme shows how the CAT is trained. The network G is fed with a Pol-SAR image. G parameters are updated based on the *L1* loss between synthetic and real optical images, plus the error of the discriminator D to recognize that the generated sample was actually fake. This training strategy, excluding the adversarial loss, is equal to the RT discussed in *Subsection* 4.2.2. The regression loss is a very good starting point to convert Pol-SAR images to optical ones. The main limitation of the *L1* loss is that the generated images are blurred. In fact, the regressive loss is very good at reproducing low frequencies, but cannot reproduce high frequencies with the same accuracy. Low frequencies contain low variational characteristics, instead high frequencies contain high variational information. As a result, the former well describes the image at a high level, the latter well describes sharp details. In *Figure* 7.3 this phenomena is very clear. This is the reason why the adversarial term is introduced. The adversarial term raises the pressure on the generator. The generator has to learn to reproduce high-frequency details to optimize the adversarial loss term. The adversarial term is an additional force that pushes the generator to improve itself.

4.2.4 Cycle Consistent Adversarial Transcoder

Cycle Consistent Adversarial Transcoder (Cycle-AT) is the implementation of the transcoder based on the cycle consistent adversarial loss. This training paradigm should support the ability to learn to transcode images using unpaired datasets. R and O are defined as two sets of radar and optical images, respectively. In this set up four networks are involved: a couple of generator-discriminators called G_R (generator of radar images) and D_R (discriminator of radar images). The second generator-discriminator couple is called G_O (generator of optical images) and D_O (discriminator of optical images). G_R has to learn to map $G_R : O \rightarrow R$, G_O has to learn to map $G_O : R \rightarrow O$. The two networks should be able to process samples and take them back to the original, hence given: $r \in R \rightarrow G_R(G_O(r)) \approx r$, and given $o \in O \rightarrow G_O(G_R(o)) \approx o$. The two transcoders are trained with a $L1$ *norm* distance between the reconstructed version of the images and the original ones. Besides, to each generator an adversarial loss term is applied. The optimization process is performed using the *Adam* function of the *Pytorch* library. In *Figure* 4.7 there is a scheme which shows the functioning of the Cycle-AT, and here below are presented the loss function equations and the objective function equation for the optimal ${G_R}^*, {G_O}^*$.

$$\mathcal{L}_{cyc_O}(\mathcal{G}_O) = \mathbb{E}_{o \in O}\left[\|\mathcal{G}_O(\mathcal{G}_\mathcal{R}(o)) - o\|_1\right] \tag{4.9}$$

$$\mathcal{L}_{cyc_R}(\mathcal{G}_\mathcal{R}) = \mathbb{E}_{r \in \mathcal{R}}\left[\|\mathcal{G}_\mathcal{R}(\mathcal{G}_O(r)) - r\|_1\right] \tag{4.10}$$

The two-cycle losses are represented by the $L1$ distance between the two reconstructed images and their original versions.

$$\begin{aligned}\mathcal{L}_{\mathrm{ADV_R}}(\mathcal{G}_\mathcal{R}, \mathcal{D}_\mathcal{R}) = \mathbb{E}_{r \in \mathcal{R}}\left[\log \mathcal{D}_\mathcal{R}(r)\right] \\ + \mathbb{E}_{o \in O}\left[\log\left(1 - \mathcal{D}_\mathcal{R}(\mathcal{G}_\mathcal{R}(o))\right)\right]\end{aligned} \tag{4.11}$$

$$\begin{aligned}\mathcal{L}_{\mathrm{ADV_O}}(\mathcal{G}_O, \mathcal{D}_O) = \mathbb{E}_{o \in O}\left[\log \mathcal{D}_O(o)\right] \\ + \mathbb{E}_{r \in \mathcal{R}}\left[\log\left(1 - \mathcal{D}_O(\mathcal{G}_O(r))\right)\right]\end{aligned} \tag{4.12}$$

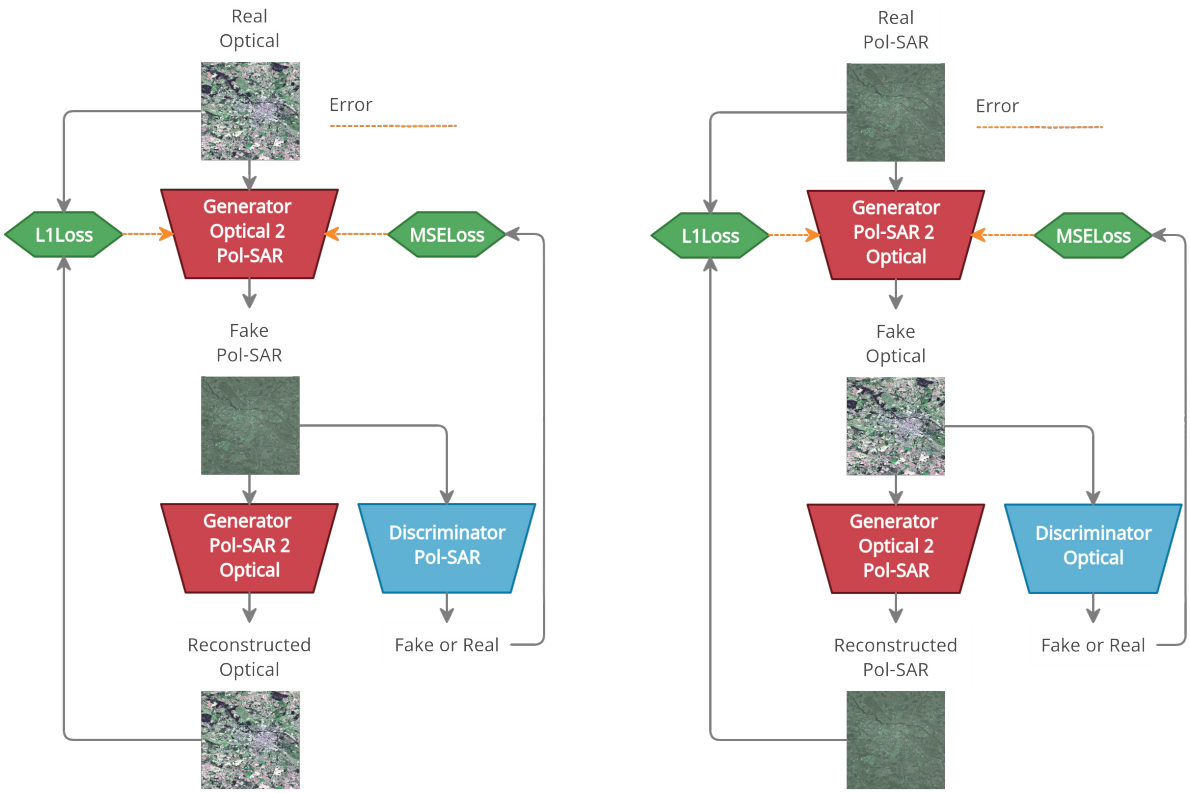
Here above the two adversarial terms are described. It is a standard adversarial loss as presented in *Section* 2.5.1.

$$\begin{aligned}\mathcal{L}(\mathcal{G}_\mathcal{R}, \mathcal{G}_O, \mathcal{D}_\mathcal{R}, \mathcal{D}_O) = \mathcal{L}_{\mathrm{ADV_O}}(\mathcal{G}_O, \mathcal{D}_O) \\ + \mathcal{L}_{\mathrm{ADV_R}}(\mathcal{G}_\mathcal{R}, \mathcal{D}_\mathcal{R}) \\ + \mathcal{L}_{cyc_O}(\mathcal{G}_O) \\ + \mathcal{L}_{cyc_R}(\mathcal{G}_\mathcal{R})\end{aligned} \tag{4.13}$$

The loss function is composed of the regressive cycle loss terms, which involves the cyclic loss in both the directions, plus an adversarial loss for each domain.

$$\mathcal{G}_\mathcal{R}^*, \mathcal{G}_O^* = \arg \min_{\mathcal{G}_\mathcal{R}, \mathcal{G}_O} \max_{\mathcal{D}_\mathcal{R}, \mathcal{D}_O} \mathcal{L}(\mathcal{G}_\mathcal{R}, \mathcal{G}_O, \mathcal{D}_\mathcal{R}, \mathcal{D}_O) \tag{4.14}$$

Finally the resulting optimal generators are optimized in an adversarial fashion.



a) Optical to Pol-SAR Generator *b) Pol-SAR to Optical Generator*

Figure 4.7: *Figure a)* and *Figure b)* show the framework to train the two Cycle-GAN generators. Each generator is trained using the loss of the reconstruction error and using the loss of a dedicated discriminator. In *Figure a)* the *Optical to Pol-SAR generator* transcodes the inputted optical image. The Pol-SAR generated image is: 1) reconverted to optical using the *Pol-SAR to Optical generator*, 2) given to the *Pol-SAR discriminator* to generate the fake or real label. The reconstructed image is used to optimize the generator based on the *L1* loss with respect to the original input. The discriminator output is used to calculate the adversarial loss to optimize the generator using the *MSE* loss. *Figure b)* shows the same process but for the *Pol-SAR to optical generator*. If we compare the Cycle-AT with CAT, beyond the complexity difference, conceptually the setup is very similar. The training process is composed of a regressive term, which should model the low frequencies of the image, and an adversarial term, which should model the high frequencies of the image. The difference lies in the double conversion applied to the input data. The double conversion is designed to empower the capacity to compare the generated sample directly with the input image. This strategy removes the constraint to have the real version of the transcoded input image. This feature introduces a marked complexity increasing. Firstly it doubles the framework parameters which augment the computational load. Secondly, the learning process is more complex due to the double transformation. The objective is not to transform an image from domain A to domain B. The objective of the Cycle-AT is to transform images from domain A to domain B to domain A. This double conversion is very challenging.

4.2.5 Fine-Tuning

All the networks described in the precedent *Sections*, once they reach their optimal, are used to perform semantic segmentation of Pol-SAR images. This operation is performed by loading a pre-trained generative network and retraining the last three convolutional layers, as introduced in *Subsection* 4.1.3. This operation is performed in a supervised framework exactly as it is explained in *Subsection* 4.2.1 for the BL. The network is trained using a *cross-entropy* loss and the *Adam* optimizer, both implemented in the *Pytorch* library.

4.2.6 Networks Complexity

All the implemented networks, BL, RT, CAT and Cycle-AT have different complexities. This is the direct consequence of the training strategy implemented. Basically, the complexity difference is related to the number of networks involved in the training process, and the consequent number of optimizers. On the other hand, during the evaluation phase, when the training is done, all the networks have the same number of parameters and the same complexity. If the network is used in evaluation mode to transcode images, the number of parameters corresponds to the number of parameters of the generator. If the network is used to evaluate the semantic segmentation performances, the networks have fewer parameters. As a fact the architecture, used as classifier, loses the last convolution operation, as it is possible to see comparing *Figure* 4.1 and in *Figure* 4.3.
Table 4.1 shows an estimated number of parameters for each ANN implemented. This *Table* shows the computational complexity. Is important to highlight that the complexity is related even to the task not only to the computational load. This means that the BL task has a complexity which is different with respect to the RT and the CAT tasks complexity, which in turn is different with respect to the Cycle-AT task complexity.

Network	Number of Generators	Number of Discriminators	Number of parameters [M]
BL	1	-	10.8
RT	1	-	11.4
CAT	1	1	14.2
Cycle-AT	2	2	28.3
Evaluation	1	-	10.8

Table 4.1: The table shows how many parameters has every network implemented in this work. The last line *Evaluation* shows the number of parameters of the generator once trained and used to perform semantic segmentation. The evaluation complexity is independently from the pre-trained model employed.

CHAPTER 5

Pipeline Bug Discovery

Contents

5.1 Bug Explanation 41
5.2 Wrongly Filtered Data Analysis 42
5.3 The Role of the Incorrect Data in this Work 43
5.4 The Effects on the Results 43

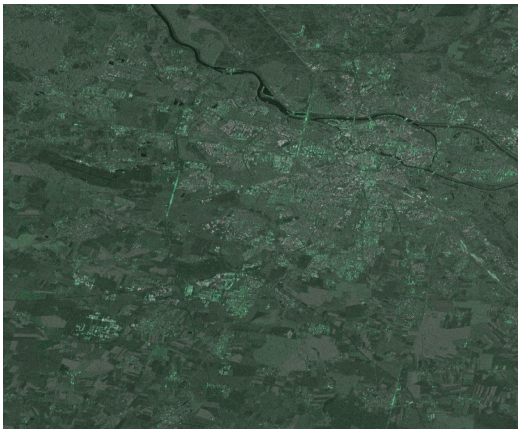
5.1 Bug Explanation

During the concluding phase of this work, when the results have been investigated deeply to precisely describe the outcomes, a problem has emerged. During the whole developing process the Pol-SAR images have been visualized using the vector v presented in *Equation* 3.1. To visualize the input data have been produced black and white images using one channel of the vector v. This simplified evaluation procedure has shadowed a procedural error. To explain where this error occurred, it is necessary to analyze the preprocessing steps. This phase is composed of these four operations:

- *Polarimetric raw data loading*: the scattering vector is loaded from the data. This vector is composed of two polarizations, each made of a complex value per each pixel.
- *Scattering matrix outer product*: for each pixel is computed the outer product between the scattering vector and its complex conjugate transpose. This gives a 2x2 Hermitian matrix.
- *Filtering phase*: once the Hermitian matrix has been calculated, each of its elements are stored in a channel of a multidimensional array. Then a two-dimensional box filter is applied, the filter averages the values in a local neighborhood of each channel of the resulting multidimensional array.
- *Feature vector v calculation*: the last step consists of extracting from the resulting matrix the features of the input vector v.

The problem concerns the *Filtering phase*. The filtering operation has been erroneously applied to three bands at the same time, instead of being applied to each band separately. During the filtering operation, each pixel has to be smoothed. The smoothing operation, in this work, has been applied using a box filter of size three. The correct application consists of transforming each pixel value in the mean of a window of size 3x3 centered over the interested pixel. An example of a correct filtering operation is shown in *Appendix* A.1.
The improper execution of this operation consists of applying the box filter to three dimensions. This means that the value of the filtered pixel is not the mean of a two-dimensional window of size 3x3, but it is applied over a three-dimensional window of size 3x3x3. The filtering operation, instead of combining the information belonging to the same channel, mixes the information between different channels. This type of calculation is incorrect, since it combines information of different nature in an unexpected way. In *Appendix* A.2 it is possible

to observe an example of a wrong filtering application. For simplicity let's introduce the acronyms Wrongly Filtered Pol-SAR Data (Wrongly-FD) and Correctly Filtered Pol-SAR Data (Correctly-FD).
This bug has been found trying to visualize the Pol-SAR input vector v in RGB format. In Figure 5.1 *a)* and *b)* is possible to see a comparison between the rendering of a colored image Wrongly-FD and Correctly-FD, respectively. The Wrongly-FD image is very similar to a black and white image. This effect is caused by the filtering operation which has averaged the values between channels. As a result, the values in the vector v are very similar.



a) Correctly filtered

b) Wrongly filtered

Figure 5.1: *Figure a)* and *Figure b)* show the coloured visualisation of vector v channels 0, 1, 2 correctly and wrongly filtered, respectively.

5.2 Wrongly Filtered Data Analysis

Analyzing the Wrongly-FD input image and the Correctly-FD input image, it is possible to observe that the informative content is not completely corrupted. Unlike, shapes and textures are quite well preserved. The smoothing operation of the wrongly applied filter combines: the channel itself, the previous one, and the consecutive one. Intuitively, this operation does not completely corrupt the information. What happens is that the original informative content of each Hermitian matrix, extracted from the outer product of the scattering matrix and its complex conjugate transpose, is deteriorated. As a proof, the image visualized in *Figure* 5.1 still contains a significative informative content.
The accidental combination of the elements of the Hermitian matrix, performed during the preprocessing phase, has no direct references in the literature. Anyway, there are plenty of methods to extract features from polarimetric synthetic aperture radar images. In particular, the so-called *Span* is one of the most similar. The *Span* is defined as S in the *Equation* 5.1, where $C = (c_{ij}), 1 \leq i, j \leq k$ is the covariance matrix. *Span* is computed summing the diagonal element of the covariance matrix [Hänsch & Hellwich 2018].

$$S = \sum_{i=1}^{k} c_{ii} \tag{5.1}$$

This is similar to what happens when the Wrongly-FD occurs. The difference lies firstly in the fact that the *Span* is not averaged, but it is a pure summation. Then the difference lies in the fact that in this work the summation has some additional terms, which are represented by the off-diagonal elements.

5.3 The Role of the Incorrect Data in this Work

After the information loss caused by the faulty preprocessing chain discovery it has not been possible to rerun all the experiments due to a matter of time. As a consequence, the results of this thesis are of two kinds Wrongly-FD and Correctly-FD, depending on which data has been used to perform the experiments.
Wrongly-FD have been used to perform all the preliminary tests and to find which is the most reasonable configuration to perform the final experiments. Besides, wrong data has been used to perform all the parameters investigations, shown in *Chapter* 6. Correctly-FD has been used to perform the final tests, which are presented in *Chapter* 7. Final tests represent the main insight of this work and define which is the real capacity of the framework implemented in this thesis. To summarize, the experiments performed can be divided in *preliminary* (Wrongly-FD), presented in *Chapter* 6, and *final* (Correctly-FD), presented in *Chapter* 7.

5.4 The Effects on the Results

[Hänsch & Hellwich 2018] have compared many Pol-SAR distance measures using several Pol-SAR features. In particular, between the others, they compare *Span* features with full polarimetric features. In the work of [Hänsch & Hellwich 2018], they perform the experiments using random forest algorithms on the semantic segmentation task. The results show that the accuracy difference related to the employed features is limited to a few percentage points. Hence, the outcomes of this work show how the performance of *Span* feature is comparable to full polarimetric information.
The above-mentioned statements support the good capacity shown by this framework to work with Wrongly-FD. Plus, the information contained in the Wrongly-FD is richer than the information contained in *Span*, as explained in earlier *Section* 5.2. As a matter of fact, the results between Wrongly-FD and Correctly-FD are similar. In *Table* 5.1 it is possible to see a comparison between: the semantic segmentation accuracy achieved using the four classifiers implemented trained with Wrongly-FD, and Correctly-FD data. The difference measured is not high, but it is significant since it shows that the information loss caused by the wrong processing was actually affecting the results. On the contrary, the limited difference in the accuracy achieved suggests that the framework implemented is resilient and can achieve good performances, even in the case of slightly corrupted input data. Finally, these results could motivate a deep investigation about the capacity of the implemented strategy, in scenarios in which data are corrupted, or noisy, or with lower resolution.

Network	Wrongly-FD	Correctly-FD	Improvement
BL	72%	73%	1% ⇑
RT	84%	87%	3% ⇑
CAT	87%	89%	2% ⇑
Cycle-AT	74%	76%	2% ⇑

Table 5.1: The table shows the semantic segmentation results using BL, the RT, the CAT and the Cycle-AT. For each implementation is proposed the top semantic segmentation OA using the Wrongly-FD and using Correctly-FD. The last column shows in green the improvement achieved using the correctly filtered data.

Chapter 6

Experiments and Networks Tuning

Contents

6.1 Transcoding Evaluation **45**
6.2 Semantic Segmentation Metric **45**
6.3 Computational Power and Execution Time **46**
6.4 Networks Tuning **47**
6.4.1 Generated Samples Bufferization 47
6.4.2 Reduced Discriminator Optimization Step 47
6.4.3 Channels Dropping 48
6.5 Network Experiments **49**
6.5.1 Patch Size 49
6.5.2 Generator Deepness 51
6.5.3 Discriminator Deepness 52

6.1 Transcoding Evaluation

In this thesis, the transcoding evaluation is not of paramount importance. What is of prominent importance is that the transcoding phase extracts meaningful features for the downstream task. In this work, the pure transcoding is evaluated straightforwardly by looking at the transcoding products. It is a very empirical method, but for this application scenario is more than enough. In fact, the goal of analyzing the transcoding products is to state if the synthetic generated images are meaningful and if the algorithms are going in the right direction. Hereafter, the best representative metric, to measure the capacity of the proxy task, is to apply the fine-tuning. Measuring the semantic segmentation accuracy is possible to investigate which transcoder extracts the best representative features for semantic segmentation of Pol-SAR images. This is the only real measure to check the transcoding features quality.

6.2 Semantic Segmentation Metric

The final task of this work is to generate a semantic map from radar images. In literature, there are many metrics to measure the accuracy between labels and predictions. In the work of [Ley *et al.* 2018], which is similar to this one, they use as classification metric the overall accuracy. The OA is calculated by dividing the number of correct predictions by the total number of predictions made. The metric formulation is shown in *Equation* 6.1.

$$Accuracy = \frac{Number\ of\ correct\ predictions}{Total\ Number\ of\ Predictions\ made} \tag{6.1}$$

6.3 Computational Power and Execution Time

A key parameter in developing machine learning algorithms is complexity, which is strictly related to the computational power. As mentioned in *Section* 2.3 more complex and deeper a network is, potentially, stronger its capacity to perform a task is. This is not always true, and in particular, the more complex a network is, the more data it requires. Besides the more complex and deeper the network is, the higher the computational time required to train the network is. As a consequence more powerful hardware is required.
Typically, DL algorithms are trained using dedicated hardware components. This class of components is called Graphic Processing Unit (GPU). GPU hardware is specifically designed to perform parallel mathematical operations, unlike from Central Processing Unit (CPU) hardware which is designed to perform sequential operations. ANNs are composed of thousands of parallel operations represented by neuron interconnections. The larger is the network, the larger is the number of operations required. As a result, the more dedicated memory is required, and the more time the training process takes. In this work has been employed an Nvidia RTX 2070 super GPU. This hardware provides 8GB of dedicated memory. The most complex algorithm implemented, the Cycle-AT, requires roughly 5GB of memory to be trained (using batch size of one and patch size of 128×128 pixels). *Table* 4.1 shows the complexity of the ANNs implemented.
Table 6.1 shows the time required to perform an optimization step using a batch size of one and a patch size of 128×128 pixels with each implemented architecture. The last line of the table shows the time required to generate the semantic segmentation map of a Pol-SAR patch of size 128×128 pixels.

Network	**Number of Parameters [M]**	**Optimization Time [ms]**
BL	10.8	~ 26
RT	11.4	~ 27
CAT	14.2	~ 28
Cycle-AT	28.3	~ 65
Semantic Map Inferring Time	10.8	~ 3

Table 6.1: The table shows the computational time to perform an optimization step using a patch size of 128×128 pixels and a batch size of 1 for each ANN implemented. The BL and RT, as expected, take roughly the same time to perform an optimization step. The CAT is slightly more complex, nevertheless, to perform an optimization step, it takes nearly the same time of the BL and RT. This is because as discussed in *Subsection* 6.4.2 the discriminator is trained every K optimization steps of the generator. The Cycle-AT takes double the time with respect to the other implementations. As a fact, it implements the transcoding task two times. One for the Pol-SAR to optical transcoding and one for the inverse process. The last line of the table shows the time required to generate the semantic segmentation map of a Pol-SAR patch of size 128×128 pixels. As is possible to see this operation is much faster with respect to the others. This is because in evaluation mode the algorithm does not perform the optimization steps. The network only infers the predictions for each pixel. This amount of time is independent of the type of feature extractor employed.

6.4 Networks Tuning

Architecture tuning can be very challenging for GANs. As mentioned in *Subsection* 2.5.2 there are a lot of possible failure modalities during GANs training. In this thesis, the training of all the networks has been mainly stable and has presented a convergence behavior. Anyway, not all the networks have reached the desired results. In particular, the Cycle-AT is the implementation that has shown most of the problems. The observed training difficulties could be addressed to some of the typical GANs failure modes. This point will be discussed, along with the final results achieved, in *Section* 7.2.2. Nevertheless, many experiments have been performed to try to find the optimal configuration for CAT and Cycle-AT. Both these two networks employ an adversarial loss. To control the capacity of the discriminator, with respect to the capacity of the generator, some strategies have been employed. All these strategies are presented in *Subsections* 6.4.1, 6.4.2 and 6.4.3. The tests to verify the functionality of these strategies have been performed using a preliminary setup with a reduced amount of data. Then it is important to highlight that the data employed to perform all the preliminary tests and the tuning experiments are Wrongly-FD as explained in *Section* 5.3. The resulting implementation is not the absolute optimal solution but is the most reasonable found.

6.4.1 Generated Samples Bufferization

Generated samples bufferization is a strategy that stores all the last N generated samples in a buffer. This buffer works as a First In First Out (FIFO) list. This list of generated samples is used to train the discriminator. The discriminator is trained to recognize fake and real samples. Applying this strategy, synthetic samples given as input to the discriminator are with a fifty percent probability: the last generated samples, or a batch of randomly drawn samples from the buffer. In other words, the discriminator is trained on the most optimized version of the synthetic samples or on random samples between the last N generated. This solution has been applied to the Cycle-AT implementation since has been shown to perform better with respect to the other strategies presented below.

6.4.2 Reduced Discriminator Optimization Step

A typical training strategy in GANs is to train to its optimal the Discriminator and the Generator, one after the other. In this work, to control the capacity of the discriminator to recognize the generated samples, the discriminator is trained every K generator optimization steps. In this way, the generator is trained longer with respect to the discriminator. As a result, the generator should be stronger than the discriminator. Improving itself, the generator, should be able to fool the discriminator. Whether the discriminator does not recognize synthetic samples created using the generator, the discriminator prediction error will be high. As a consequence, the discriminator parameters are updated to improve its capacities. Finally, a stronger discriminator increases the pressure on the generator which, in turn, should improve its abilities. If the balance between these two forces is not controlled they can be stuck in a non-optimal state. This strategy has been applied only to the CAT implementation, since the other two strategies investigated have not shown significant improvements.

6.4.3 Channels Dropping

To control the capacity of the discriminator with respect to the generator a dropping strategy has been implemented. In practice, the dropping consists of altering the discriminator input tensor. The input is modified using two degrees of randomness. The first consists of applying the dropping only if a randomly generated number (from a normal distribution between 0 and 1) is above a specific threshold. The second decides how many and which channels to drop. The decision is made by randomly choosing from a minimum of one to a maximum of three channels. Then are generated the corresponding random numbers to decide which channels to drop. If the first randomly generated number is above a specific threshold, every single value of the randomly chosen channels will be stubbed to zero.
The dropping force is controlled using the actual capacity of the discriminator to recognize the generated samples. If the discriminator is strong, hence the confidence that a fake sample is actually fake is high, the channel dropping is enforced. On the contrary, if the discriminator is weak the channel dropping is decreased. The channel dropping force is controlled by changing the probability threshold and the maximum number of channels to drop. This strategy has not been applied since was quite complex and did not show significant improvements.

6.5 Network Experiments

The networks used for the transcoding task have been investigated to understand their sensibility to some specific parameters. The *Subsections* 6.5.1, 6.5.2 and 6.5.3 explain which and how parameters have been investigated. Each parameter investigated is applied to all the transcoding networks implemented plus the baseline. To test each parameter the evaluation is performed running a full dataset test for around twenty epochs. Training for twenty epochs does not reach the absolute optimal. Anyway, has been observed to be a reasonable number of epochs to find out which is the potential of the setups implemented. Even in this case, it is of paramount importance, to highlight that the data employed to perform these sets of tests are Wrongly-FD, as explained in *Section* 5.3. To measure the transcoding results quality the reference metric is the semantic segmentation OA presented in *Section* 6.2. All the numerical results reported in the next *Subsections* (6.5.1, 6.5.2 and 6.5.3) are performed using the test tile (see *Section* 3.3). The accuracy numerical results reports the OA achieved after using the transcoder as features extractors for the semantic segmentation task. The calculations are made training the networks with the entire training set and evaluating the results with the whole test set. This means that if a classification algorithm reaches 80% of accuracy around 1 million of pixels in the test set are correctly classified. This, because as shown in *Section* 3.3 the test set is made of around 1.2 million of annotated pixels.

6.5.1 Patch Size

The first parameter analyzed is the patch size. The patch sizes investigated are two:

- *256×256 pixels.*
- *128×128 pixels.*

In *Figure* 6.1 is possible to see the difference between a sample of size 256×256 pixels, versus a sample of 128×128 pixels. In *Table* 3.2 is shown the number of patches with respect to the two above-mentioned sizes.
The reasons to analyze the patch size are related to the fact that using larger patches is possible to analyze larger areas. This could help to recognize structures that are not present in smaller patches. This is not absolutely true, since the image content is not only related to the number of pixels but even on the pixels resolution.
[Zhu *et al.* 2017a] have first proposed the framework implemented in this thesis, they used the Cycle-GAN using patch sizes between 128×128 and 256×256 pixels. They state to get better performances using an architecture with nine residual blocks in the backbone with a patch size of 256×256 pixels. Instead, using a patch size of 128×128 pixels, they state to get better performances with six residual blocks in the backbone.
For these reasons, the effect of the patch dimension has been investigated.
Table 6.2 shows the semantic segmentation accuracy performances using patches of size 128×128 pixels, versus patches of 256×256 pixels. The above-mentioned *Table* 6.2 contains the results of the BL, the RT, the CAT and the Cycle-AT. The table shows very clearly that the ANNs trained with a patch size of 128×128 pixels outperforms the networks trained with a patch size of 256×256 pixels. One possible reason for this result is related to the number of patches: 128×128 patches are four times the 256×256 patches (see *Table* 3.2). Besides, larger patches are more challenging to train since the informative content is higher and the complexity of the image increases with size. During the very first deploying phase some tests were performed on very small patches (32x32 pixels) and the results were bad because very small patches does not allow to recognize some fundamental characteristic of the target classes. For example, urban areas are not well defined due to the resolution of the image, recognizing an urban area using only 32x32 pixels can be not trivial. In particular patches are randomly sampled form the original tile, hence only a fraction of the pixels could represent urban making the classification very complex. In conclusion, according to *Table* 6.2 all the implementations present

a significant degradation of the accuracy when the patch size is enlarged. This is a strong indication that the most reasonable patch size to execute the final tests is 128×128 pixels.

a) 128×128 pixels

b) 256×256 pixels

Figure 6.1: *Figure a)* shows a patch with 128×128 pixels and *Figure b)* shows a patch with 256×256 pixels. The images are presented with the same dimension but a different number of pixels, as a fact the *Figure b)* has higher resolution.

Network	Patch Size	Overall Accuracy Test Set
BL	128	73%
BL	256	71% - 2% ⇓
RT	128	84%
RT	256	79% - 5% ⇓
CAT	128	87%
CAT	256	81% - 6% ⇓
Cycle-AT	128	76%
Cycle-AT	256	75% - 1% ⇓

Table 6.2: The table shows the semantic segmentation results using BL, the RT, the CAT and the Cycle-AT. The first line of each implementation shows the result using a patch size of 128×128 pixels. The second line of each implementation shows the result using a patch size of 256×256 pixels. In red is presented the variation of the accuracy changing the patch size. The arrow indicates the decreasing direction.

6.5.2 Generator Deepness

Another investigation performed during the testing phase has been devoted to network deepness. The tests were performed changing the deepness of the generator which is presented in *Subsection* 4.1.1. The generator architecture deepness has been manipulated changing the number of residual blocks placed in the backbone of the network. As a matter of fact, in this work [Reyes *et al.* 2019] state that more residual blocks they employ better resolution they achieve. On the other hand, in the original Cycle-GAN, [Zhu *et al.* 2017a] state that they got better results using a shallower architecture (six residual blocks) if the patch size is 128×128 pixels. Instead if the patch size is larger they use a deeper architecture (nine residual blocks). In this thesis, the standard patch size is 128×128 pixels as discussed in *Subsection* 6.5.1. The generator architecture has been tested using both nine and six residual blocks in the backbone.
In *Table* 6.3 are presented the results of BL, the RT, the CAT and the Cycle-AT trained using six or nine residual blocks. As reported in *Table* 6.3, the variation of the deepness of the generator architecture does not involve a critical performance change. As a result, this architectural alteration has not been considered during the final test phase. Even if the six residual blocks configuration reduces the deepness of the generator slightly accelerating both the training and the evaluation phases. In fact, a shallower algorithm has a reduced amount of parameters. Despite this advantage, the final implementations have been configured with nine residual blocks for the final test phase presented in *Chapter* 7. This decision is related to the fact that the classification results using six residual blocks were slightly worse. For what regards computational reduced complexity, instead, the difference was negligible.

Network	Number of Residual Blocks	Overall Accuracy Test Set
BL	9	73%
BL	6	74% - 1% ⇑
RT	9	84%
RT	6	84% - 0% ~
CAT	9	87%
CAT	6	85% - 2% ⇓
Cycle-AT	9	76%
Cycle-AT	6	75% - 1% ⇓

Table 6.3: The table shows the semantic segmentation results using BL, the RT, the CAT and the Cycle-AT. The first line of each implementation shows the OA accuracy using nine residual blocks in the backbone of the generator (the architecture details are discussed in *Subsection* 4.1.1). The second line of each implementation shows the result using six residual blocks in the backbone of the generator. The colored numbers describe the variation between the two versions of each classification method. In red, the accuracy values decreased using six residual blocks instead of nine, in orange when there is no variation (only small fluctuations), and in green if the accuracy has increased. The arrows indicate the increasing/decreasing directions. The tilde represents no significant changes.

6.5.3 Discriminator Deepness

The discriminator deepness is another investigation that has been performed. The discriminator is composed of five convolution operations, three used to downsample the image, one in input and one in output. The detailed architecture is shown in *Subsection* 4.1.2. The discriminators play an important role in the min-max game of GANs training. They have always shown to be strong in recognizing real and fake samples: the discriminator architecture has been analyzed changing the number of downsampling layers. The rationale behind this idea is to weaken the discriminator. The consequence should be a better balance between generation and discrimination.
The discriminator deepness is a parameter that is not involved in all the transcoders implemented. The need of a discriminator depends on the functioning of the training paradigm implemented. CAT and Cycle-AT are the two implementations that rely on a discriminator, one for the former and two for the latter. This parameter evaluation has been performed testing both the CAT and Cycle-AT discriminators. The transcoders have been investigated using three downsampling convolutional layers, and two downsampling convolutional layers.
The alteration of the discriminator deepness has not been considered during the final test phase since the results are not significant. In fact, as is possible to see in *Table* 6.4, the networks classification capabilities do not improve using a shallower discriminator.

Network	**Number of Convolutions in the Discriminators**	**Overall Accuracy Test Set**
CAT	3	87%
CAT	2	87% - 0% ~
Cycle-AT	3	76%
Cycle-AT	2	74% - 2% ⇓

Table 6.4: The table shows the semantic segmentation results using CAT and the Cycle-AT. The first line of each implementation shows the OA accuracy using a discriminator architecture with three downsampling convolutional layers (the discriminator architecture details are discussed in *Subsection* 4.1.2). The second line of each implementation shows the result using two downsampling convolutional layers in the discriminator. The colored numbers describe the variation between the two versions of each classification method. In red, the accuracy value decreased using two downsampling layers instead of three, and in orange when there is no variation (only small fluctuations). The arrow indicates the decreasing direction. The tilde represents no significant changes.

CHAPTER 7

Results and Discussions

Contents

7.1 Final Configuration **53**
7.2 Transcoding Results **54**
7.2.1 Transcoders Comparison 56
7.2.2 Cycle Consistent Adversarial Transcoder Analysis 58
7.3 Semantic Segmentation Results **59**
7.3.1 Semantic Segmentation Accuracy Analysis 60
7.3.2 Accuracy Results Statistical Validation 62
7.3.3 Labeled Samples Sensitivity Analysis 63
7.3.4 Classification Maps Analysis 66

7.1 Final Configuration

In this *Chapter* are presented all the final tests executed. As mentioned, in *Section* 5.3, the final experiments are performed using Correctly-FD. To perform the trials reported in the next *Sections*, the two original tiles have been reprocessed using the correct procedure and cut into patches to run experiments using real Pol-SAR data. The experiments are done using the tile from the area of Poznan, Poland to train the algorithms, and the tile coming from the area of Wroclaw, Poland to test the capacity of the algorithms. The numerical accuracy results presented will refer to OA metric, which is presented in *Section* 6.2. The calculation of the OA is performed inferring the classification value for each pixel in the test set.
The final transcoders configuration is based on the generator presented in *Subsection* 4.1.1, exactly as it is, with nine residual blocks. The discriminator architecture is based on the one presented in *Subsection* 4.1.2, exactly as it is, with three downsampling convolutional layers. The patch size employed is 128×128 pixels.
All the above-mentioned configurations are valid for all the final tests presented in this *Chapter*. *Table* 7.1 shows the final optimal configuration for the framework implemented. The choice of these hyperparameters is discussed in *Section* 6.5.
The classification step is performed with the architectures configured as described in *Section* 4.1.3.

Configuration	Patch Size	N of Residual Blocks	Downsampling Conv Layers	Pol-SAR Data
Final	128×128	9	3	Correctly-FD

Table 7.1: The table shows the configuration for all the final tests performed presented in *Chapter* 7. The choice of these hyperparameters is discussed in *Subsection* 6.5.

7.2 Transcoding Results

In this *Section* the framework transcoding capacity is presented. Firstly, is presented an example of a full test and train input Pol-SAR tile transcoded to the optical domain. In *Figures* 7.1 and 7.2, there are the best transcoding results achieved using the CAT trained for 100 epochs. The network that has created these images is the same which is able to reach the highest accuracy level during the semantic segmentation phase. The results of semantic segmentation are presented in the *Section* 7.3. In *Appendix* B is shown the original input Pol-SAR images, and the original optical images. *Appendixes* B.1, B.4, B.2, and B.5 show train and test original Pol-SAR images and train and test original optical images, respectively. Besides, in *Appendix* C.1, C.5, C.2 and C.6 are shown the input Pol-SAR images (both train and test tile) transcoded using the best RT and the best Cycle-AT networks implemented, respectevely.
Comparing the original optical images with the ones transcoded using the RT and the CAT, it is possible to see that, at a high level, the train tiles are truly similar to the original ones. The Cycle-AT, instead, preserves the general appearance but has not the quality shown by the other two transcoded products.
Concerning the test tiles, as expected, the results are worse, also at a macroscopic level. Anyway, the images produced using the RT and the CAT are reasonable and they are similar to the original optical tile. Instead, for what regards the Cycle-AT, the synthetic optical test tile is not similar to the real optical test tile. In *SubSection* 7.2.1 the transcoded products are deeply discussed.
Independently from the esthetic and the appearance of the transcoded products, the objective of this work is not to transcode images. The core objective is to learn representative features and verify their capacity when employed to perform the semantic segmentation task. Hence, the main goal is that the transcoders create reasonable synthetic versions of the input Pol-SAR images, in order to confirm the sanity of the framework. The aim is to extract features that allow to achieve good performances when employed to perform Pol-SAR images semantic segmentation.

Figure 7.1: The image shows the synthetic optical train tile generated using the optimal CAT.

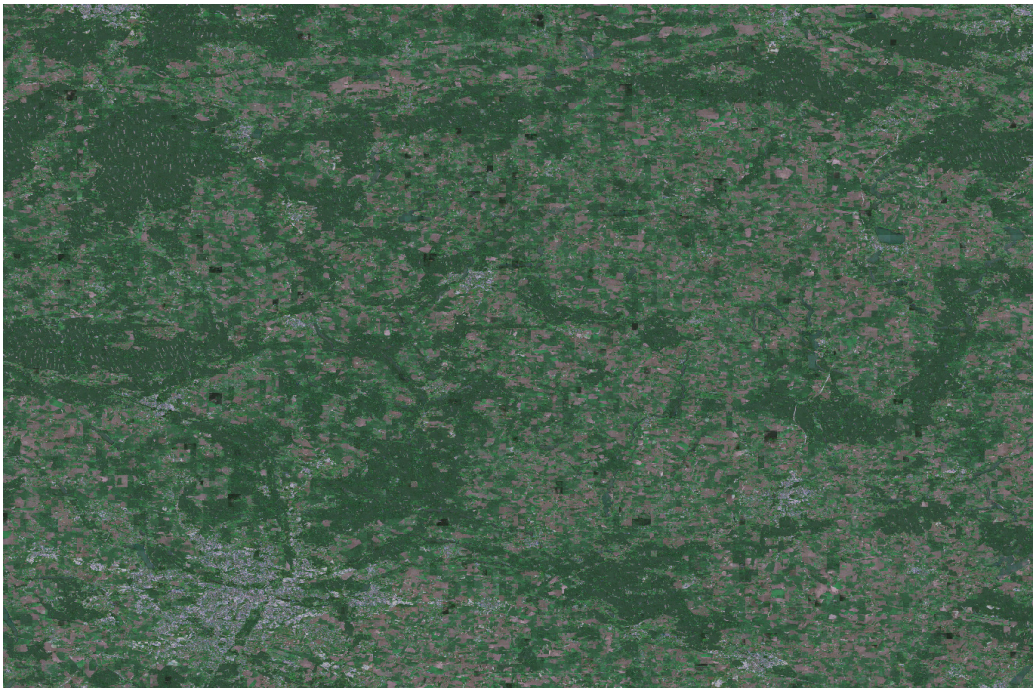


Figure 7.2: The image shows the synthetic optical test tile generated using the optimal CAT.

7.2.1 Transcoders Comparison

In *Figure* 7.3 is shown the comparison between the input Pol-SAR data, the Pol-SAR input image transcoded using: the RT, the CAT and the Cycle-AT, respectively, and the actual optical corresponding image. For each of the image types reported are shown three samples belonging to the test tile. The first sample represents a mixed area with forests, fields, and urban. The second one represents a rural area with fields. The last one represents an urban area with a lake and a river. The samples shown in *Figure* 7.3 have been chosen since they represent almost every target class in the semantic segmentation dataset (*Section* 3.2). The semantic segmentation target classes summarize almost every element present in the tile.
Analysing *Figure* 7.3 is possible to see that the images transcoded using the RT are blurred but are coherent. Every part is transcoded recalling the real optical image. The main elements are preserved, for example in the first image the city is not well-rendered but is recognizable.
The images transcoded using the CAT are macroscopically similar to the ones transcoded with the RT. They show the main macro elements. Unlike the RT transcoded images, the resolution of CAT transcoded products is much higher. For instance, the third image, which is mainly composed of urban areas, clearly highlights the good resolution achieved using the CAT.
Focusing the attention on the images in the first line of *Figure* 7.3, it is possible to see that the forest textures, reconstructed using the CAT, is very sharp. Nevertheless, there are also some defects. In the central image, it is possible to observe in the left top corner an artifact. This artifact could be reconnected to mode-collapse, since it is not the sole occurrence in the whole image [Durall *et al.* 2020]. Anyway, these aberrations are isolated and most of the transcoded patches are well rendered.
The CAT and the RT present a common defect, the fields are not clearly rendered. In particular, fields are often more similar to forests. For example, in the first line of the *Figure* 7.3 this phenomena is very clear. At the top of the pictures, some fields are transcoded as forests. This is proof of the difficulties in distinguishing between forests and fields. As a matter of fact, as explained in *Subsection* 7.3.4 the target class *Fields* is one of the most challenging. More precisely the target class *Fields* is often assigned to the class *Forests*, as is possible to see in the confusion matrices in *Appendix* D.
The images transcoded using the Cycle-AT are clearly not at the level of the ones transcoded using the RT and the CAT. As a fact, even the classification results, using this pre-trained model, do not reach the capacity of the other pre-trained networks, as deeply discussed in *Subsection* 7.3. The Cycle-AT developed in this thesis is not able to perform high-quality transcoding. But it is a fact that it implements the most complex training paradigm investigated in this work. Nevertheless, *Section* 7.3 shows that the accuracy, using the feature extracted by the Cycle-AT, performs better than BL classifier. Besides, the Cycle-AT pre-trained features have shown to be very resilient to label scarcity. This indicates that, even if the transcoding products are not perfect, the features extracted are meaningful.

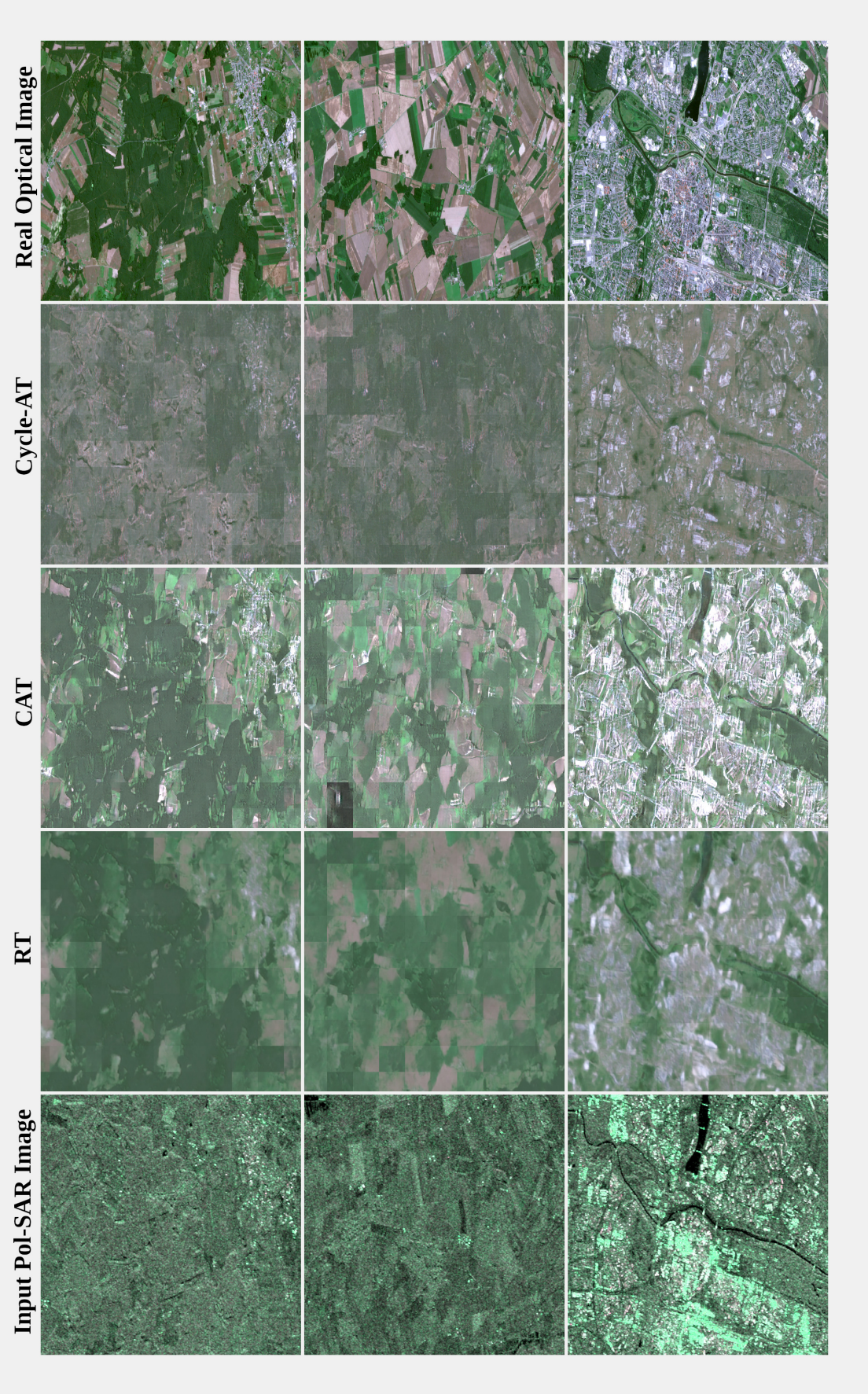


Figure 7.3: The figure shows three different samples from the test tile, each sample is shown in a different row. Each column represents: 1) the Pol-SAR input features vector (the first three features of the vector are rendered as RGB channels), 2) the column 1) transcoded version using the RT, 3) the column 1) transcoded version using the CAT, 4) the column 1) transcoded version using the Cycle-AT, 5) the real optical version of the input 1).

7.2.2 Cycle Consistent Adversarial Transcoder Analysis

GANs training is equivalent to the search of Nash equilibrium in a high-dimensional, highly non-convex optimization space. As already discussed in *Subsection* 2.5.2, the main problems that it is possible to encounter during GANs optimization are: *instabilities*, *mode collapse*, *difficulties in evaluation*, *catastrophic forgetting*, *non-convergence*, *vanishing gradient*.
In the framework implemented it is difficult to state which failure mode to ascribe to the results. The cause can be a very strong mode-collapse. In *Appendixes* C.5 and C.6 are shown the Pol-SAR train and test tiles transcoded to optical using the Cycle-AT. As it is possible to see, the test tile transcoding result seems to be made of a limited set of possibilities in each image position. To mitigate the weaknesses of Cycle-AT have been made a lot of trials. For example, it has been tried to implement the Wasserstein Generative Adversarial Network (WGAN) loss [Arjovsky *et al.* 2017]. The WGANs are an updated version of the GANs. The authors propose a modified loss function that is correlated with the quality of the generated images, instead of being based only on real or fake labels. The *Wasserstein* loss can be helpful to overcome the mode-collapse and the vanishing gradient issues. This experiment has shown no improvements in the Cycle-AT performances.
Another attempt has been done trying both to enforce and to weaken the discriminators employed to train the Cycle-AT. In *Section* 6.4 and *Subsection* 6.5.3 have been extensively explained the modalities that have been employed to try to control the optimization of the Cycle-AT. As discussed in the above-mentioned *Sections*, despite all the strategies employed to improve the Cycle-AT, the performances shown by the RT and CAT have not been achieved.
A possible reason to explain the lower level of performances of the cycle consistent adversarial training strategy, with respect to the RT and CAT, could be related to the unbalanced complexity of the transcoding task. Transcoding images from the Pol-SAR domain to the optical domain, with respect to transcoding images from the optical domain to the Pol-SAR domain, is not balanced as the task implemented in the original Cycle-GAN proposed by [Zhu *et al.* 2017a]. Unlike this work, the framework of [Zhu *et al.* 2017a] translates optical images changing colors, textures or other image properties. Hence the source and the destination domains have different characteristics but are both composed of optical images. In the Cycle-GAN implementation of [Reyes *et al.* 2019], the authors perform a slightly different task with respect to this work. They aim at correctly visualizing images and they do not reconstruct the polarimetric information. The work of [Saha *et al.* 2020] instead seems to be successful in applying a very similar task, but the paper does not show images or specific details about the transcoding. The paper is devoted to the change detection task and transcoding is used only as pretext task without going into the details.
Another possible reason, to explain the difficulties in transcoding images using the Cycle-AT, could be ascribed to the intrinsic complexity of the task. In fact, the complexity of transcoding images from domain A to domain B is much simpler than transcoding images from domain A to domain B and back to domain A. The former is the objective of the RT and the CAT, the latter is the case of the Cycle-AT. As a matter of fact, these three implementations are deployed using the same pipeline, the only difference concerns the training strategies. It is possible that using these data, these setups, these strategies, these architectures, the results achieved using the RT and the CAT are not achievable using the Cycle-AT.
Finally, the consequence of the above-mentioned suppositions, based on the experiments performed, suggest that the only solution to achieve better performances, using the Cycle-AT, could be a profound change. For instance, changing the data processing, enlarging the dataset, or deeply altering the network architectures.

7.3 Semantic Segmentation Results

In this *Section* are presented the final semantic segmentation results. The final objective is to use the networks pre-trained to transcode radar to optical images, to perform Pol-SAR semantic segmentation. In *Table* 7.2 is summarized the final capacity of the four implementations proposed. The values reported in the *Table* represent the OA of the classifiers implemented, trained for 30 epochs, using all the labeled patches available. These values are calculated performing a *statistical* validation, as explained in *Subsection* 7.3.2. Basically, the results are achieved by performing ten times the training process and measuring the accuracy performances on the whole test set. The results reported in the *Table* 7.2 are the average values achieved over the ten experiments. The last column of the table reports the standard deviation of the results over the ten runs performed.
In *Figure* 7.4 there is a plot which shows the OA of the four concepts implemented (BL, RT, CAT and Cycle-AT). The plot shows the accuracy calculated on the whole test set at each classifier training epoch. The chart reports one of the ten experiments performed during the *statistical* validation phase. The results shown in this *Chapter* are achieved using the framework configured as described in *Table* 7.1 and input Pol-SAR data Correctly-FD. The classifier, powered by the features extracted using the CAT, reaches a capacity of around 89% of correctly classified pixels with respect to the total number of classified pixels. The conditional adversarial transcoder achieves the best results. The results accomplished on the train set are not considered, since are not significant.
In *Figure* 7.4 it is interesting to notice that the accuracy of the CAT, the RT and of the Cycle-AT are optimal already after very few epochs. This is due to the very shallow architecture of the classification network. But it is even a confirmation that the pre-trained features are easily adaptable to the classification domain and well representative for the downstream task.

Network	**Pre-Trained**	**Overall Accuracy Test Set Over 10 Runs**	
		Average	**Standard Deviation**
BL	NO	73%	± 0.94%
RT	YES	87%	± 0.19%
CAT	YES	89%	± 0.18%
Cycle-AT	YES	76%	± 1.42%

Table 7.2: The table shows the final accuracy levels reached using the four classifiers implemented. The accuracy values reported are the average value achieved performing ten times the training process. The last column shows the standard deviation of the values calculated over the ten repetitions performed.

7.3.1 Semantic Segmentation Accuracy Analysis

Here below follows a bullet point list in which are described the accuracy performances, of each classifier implemented, referring to *Figure* 7.4:

- *The Baseline*: is the one who performs worse and the only model that is not pre-trained. It reaches a maximum accuracy of around 73%. The learning curve has a fundamental difference with respect to the others: it learns slower. The baseline is implemented using the generator shown in *Subsection* 4.1.1 which, unlike the other methods, is not pre-trained. As a result, the semantic segmentation training process has to train all the layers. Anyway, once it reaches its optimal the results are neither comparable to the RT or the CAT capacity. The results are even worse than the Cycle-AT classification capacity.

- *The Regressive Transcoder*: is the second best implementation, it reaches around 87% of OA. The results are very similar to the CAT. The RT final accuracy is around two percentage points below the one reached using the pre-trained features extracted from the CAT. The process to learn the pre-trained features, using the regressive transcoder, is one of the simplest, comparable to the complexity of the BL. In *Table* 4.1 is shown each training strategy complexity. The RT is simpler compared to the CAT, in fact, after 20 epochs it reaches its optimal capacity. After the same number of epochs, the classification accuracy of the CAT is slightly worse. The results shown in *Figure* 7.4 are obtained training the RT for 20 epochs and the CAT for 100 epochs.

- *The Conditional Adversarial Transcoder*: is the top-performing implementation, it reaches almost 89% of overall accuracy. As shown in *Figure* 7.4, the CAT produces the best transcoding product with the highest resolution. Nevertheless, the RT, even if the transcoding results are less sharp, achieves very competitive classification results.

- *The Cycle Consistent Adversarial Transcoder*: reaches 76% of accuracy. As already mentioned in the previous *Subsection* 7.2.1, the Cycle-AT does not transcode images with the quality of the RT or CAT. Cycle-AT generates images with some artifacts and aberrations. In fact, the classification results are not competitive with respect to the other two pre-trained models. The Cycle-AT accuracy is more than ten percentage points lower than the RT and the CAT. However, the capacity of the classifier which uses the pre-trained features of the Cycle-AT performs better than the BL. The Cycle-AT pre-trained features used to perform the semantic segmentation achieve an OA that is three points higher with respect to the baseline classification accuracy.

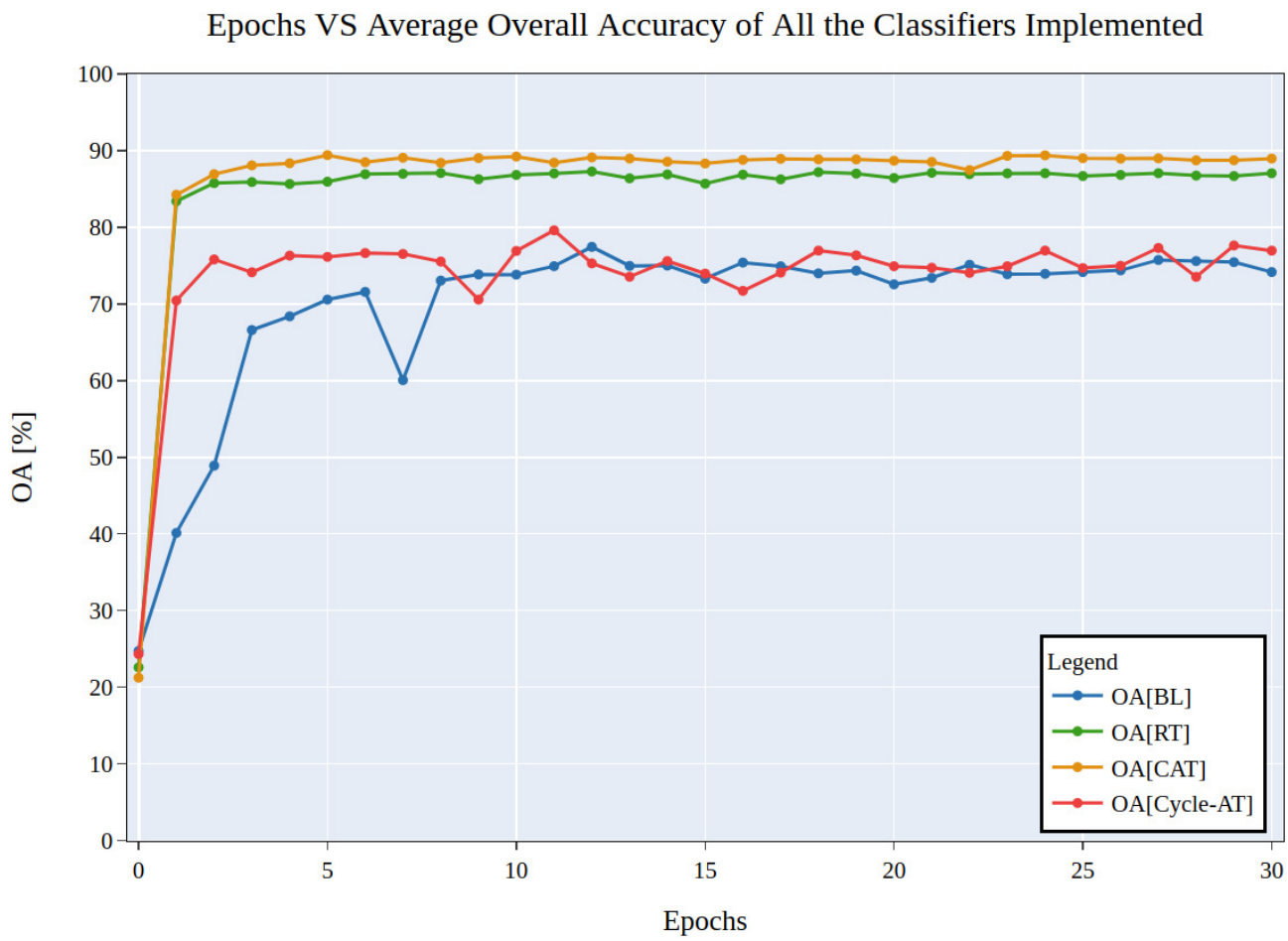


Figure 7.4: The image represents the OA of the four classification networks implemented after each training epoch. In the legend: [BL] refers to the results achieved using the BL, [RT] refers to the results achieved using the RT with the last layers retrained, [CAT] refers to the results achieved using the CAT with the last layers retrained, [Cycle-AT] refers to the results achieved using the Cycle-AT with the last layers retrained. The detailed process of fine-tuning is discussed in *Section* 4.1.3.

7.3.2 Accuracy Results Statistical Validation

The accuracy results shown in this thesis have been statistically validated. The validation performed is similar to Monte Carlo Cross Validation (MCCV). The MCCV randomly samples a part of the dataset and uses it as train set, while the remaining part is used as test set. This process is repeated N times, at each iteration the train and test sets are randomly sampled to generate different combinations. The result is the average of all the measures performed [Kuhn 2013].
Unlike MCCV, in this thesis, there is not a unique dataset to be divided into test and train sets. In this work, as discussed in *Section* 3.3, there are two separate train and test sets. As a result, the statistical validation is performed randomly sampling only the subsets of the train set. The testing set is always fully employed.
The statistical validation of the results is required to validate the sensitivity analysis shown in *Subsection* 7.3.3. The sensitivity is performed training the classification algorithms using the subsets of the training dataset shown in *Table* 7.3. The validation process is necessary because the downsampling process can produce wrong or imprecise results, in particular when the subset is very small. In fact, whether a subset is not composed with all of the classes in the dataset, or the classes are strongly unbalanced, the classification algorithm has no chance to learn to recognize never-seen classes.
The statistical validation is performed by training the classifiers using every subset N times. For each training, a new subset is randomly sampled. Each classifier implemented (the BL, the RT, the CAT and the Cycle-AT), is trained using the same randomly sampled subsets. Each of the N tests performs differently based on how the dataset is sampled. To measure the accuracy N times allows to correctly estimate the real capacity of the algorithms. The outcomes of the ten measures are the average and the standard deviation of the accuracy values. These two statistical indexes well describe the capacity of the classifiers. In theory, to perfectly describe the statistical average and standard deviation of the results, all the possible combinations of patches should be tested, but this is not feasible.
For completeness, also the full dataset results have been performed calculating the accuracy over ten measures. In the training procedure, there are random components that can affect the final results. For example, in this framework, the ANNs weights are randomly initialized. In the generator, there are dropout layers that introduce randomness when searching the final optimal of the network. These uncertainty sources can cause fluctuations. These little fluctuations are smaller than the ones that can cause training a classifier with a strongly unbalanced subset.

Percentage of Labeled Samples	Number of Samples
100%	2730
90%	2457
80%	2184
70%	1911
60%	1638
50%	1365
40%	1092
30%	819
20%	546
10%	273
5%	136

Table 7.3: The table shows the subsets used to perform the sensitivity analysis presented in *Subsection* 7.3.3. The first column shows the name of the subsets, which also represents the percentage of total available annotation employed to train the classifiers. The second column shows the corresponding number of training patches with size 128×128 pixels.

7.3.3 Labeled Samples Sensitivity Analysis

Self-supervised learning is a technique that empowers the capacity to leverage unlabeled data in scenarios where few labeled and many unlabeled data are available. In fact, self-supervised learning allows to automatically generate labels without any human support. This strategy autonomously supervises itself. In this thesis, this concept is applied to the Pol-SAR to optical image transcoding. Training an algorithm to translate radar images into optical ones allows to learn useful features for radar image classification.
In *Section* 7.3 has been shown the framework semantic segmentation capacity using all the available labeled patches. The self-supervised learning is thought to improve the results when labeled data are scarce and the annotation procurement is a costly operation. For these reasons, this *Section* shows the analysis concerning the sensitivity of the classifiers implemented with respect to the number of labeled samples available. The classifiers are trained using ten subsets of the training dataset. In *Table* 7.3 is shown the number of training patches of each subset tested. As it is possible to see, the subsets are composed using a number of samples that span from 5% to 100% of the total available data samples. This set of experimentations aims to investigate how the semantic segmentation OA varies changing the quantity of labeled data available.
The *Figure* 7.5 plots the accuracy results of each classifier implemented, versus the subsets of labeled samples employed to perform the classifiers training phases. In this plot are shown all the results achieved using the subsets shown in *Table* 7.3.
As mentioned in *Subsection* 7.3.2, the accuracies are represented by means of two statistical indexes: the average and the standard deviation. These parameters are calculated performing ten measures of each investigated classifier OA trained with each annotation subset. Every measure is characterized by a different random combination of the patches in the subset used to train the algorithm. The four classifiers implemented are all trained using the same ten combinations of each subset presented in *Table* 7.3.
In *Figure* 7.5, for each investigated percentage, the bar heights represent the average value, the grey lines rep-

resent the standard deviation, both calculated over ten runs.
Analyzing the *Figure* 7.5, it appears that the pre-trained models, varying the availability of labeled samples, have a logarithmic trend which tends asymptotically to their maximum capacities. In fact, pre-trained models are essentially not affected by label data scarcity over the 30% of data availability. Instead, under 30% the algorithms are barely affected by data lacking. This is a sign of very strong resilience to the quantity of labeled data. In particular, with only the 5% of labeled data, the CAT transcoder fine-tuned reaches an accuracy higher than 81%. All three pre-trained models lose only roughly seven percentage accuracy points when the amount of annotated data is reduced to the 5% of the total. Instead, for what concerns the baseline, it is possible to see a trend that seems to linearly decrease under the 80% of data availability. Besides, the baseline, when trained with less than the 40% of total labeled data, shows performances under the 50% of OA.
In *Figure* 7.5 the grey lines, that show the accuracy value dispersion, are mainly not significant. Only the BL and the Cycle-AT present slightly unstable results, in particular when trained with the 20% and the 90% subset, respectively.
These results can be explained by the fact that these two networks exhibit a training process which is less stable than the RT and CAT, as it is possible to notice in *Figure* 7.4. The baseline instabilities are related to the reduced dimension of the batch size used during the learning process. The GPU employed in this work does not support the implemented models with large batch sizes. The problem is that the GPU does not have enough dedicated memory. Regarding the Cycle-AT, the instabilities could be caused by the quality of the extracted features that are not stable as the ones of the RT and the CAT. As shown in *Section* 7.2.2, the transcoding performances of the Cycle-AT are not perfect.
Finally, the results achieved using the smaller subsets are less stable. This is true for all the experiments, in fact, it is a general behavior in the plot. This, because as explained in *Subsection* 7.3.2, it is more likely to find non optimal combinations of samples in small subsets.

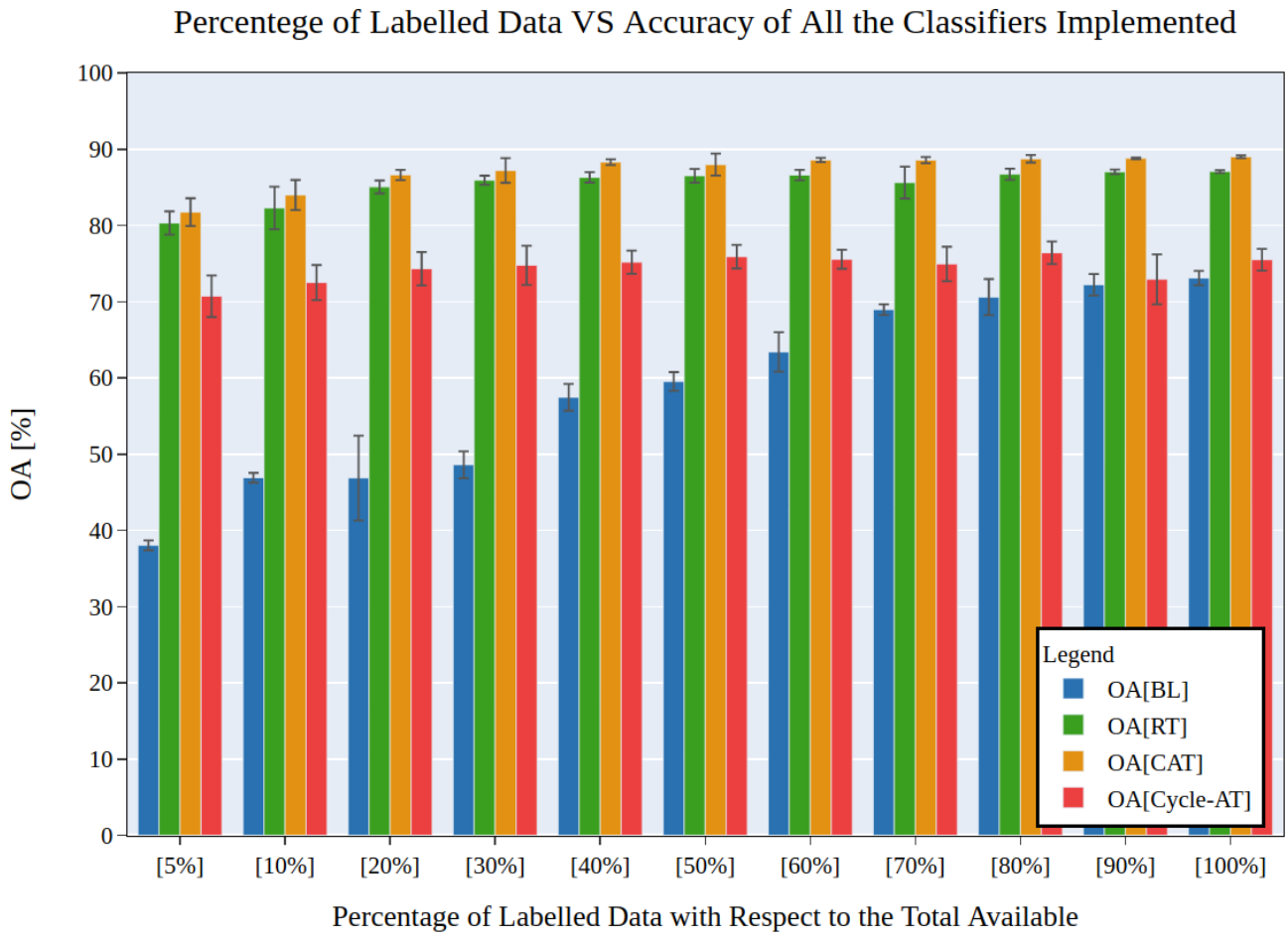


Figure 7.5: The image represents the OA accuracy of the four classification networks implemented. For each implementation is plotted the top-performing accuracy achieved using a different amount of labeled data. Each group of bars represents the accuracy achieved with each implementation using the corresponding percentage of labeled data shown on the x-axis. The bar heights represent the average value, the grey lines represent the standard deviation achieved performing ten measures for each bar represented. Every measure differs for the combination of samples in the training subset. For each percentage investigation are used the same ten dataset combinations. In the legend: [BL] refers to the results achieved using the BL, [RT] refers to the results achieved using the RT with the last layers retrained, [CAT] refers to the results achieved using the CAT with the last layers retrained, [Cycle-AT] refers to the results achieved using the Cycle-AT with the last layers retrained.

7.3.4 Classification Maps Analysis

In *Figures* 7.6, 7.7, and 7.8 are depicted, from the first to the last row, three samples of the semantic segmentation test tile map. The samples are generated using the BL classifier and the RT, the CAT and the Cycle-AT pre-trained models with the last layers retrained to perform the classification task. Instead, from the left to the right column the image contains:

- *Input Pol-SAR Image*: is the input sample form where the maps have been generated.
- *5%*: represents a map generated using the classifiers trained with the 5% of total available annotated data.
- *50%*: represents a map generated using the classifiers trained with the 50% of total available annotated data.
- *100%*: represents a map generated using the classifiers trained with the totality of available annotated data.
- *Labeled Image*: represents the reference label map.

The resulting maps presented in *Figures* 7.6, 7.7, and 7.8, as expected, are aligned with the numerical accuracy presented previously in this *Section*. To support the images interpretation, in *Appendix* D, for each transcoder implemented, is shown its Confusion Matrix (CM). The CMs shown in *Appendix* D refer to a single run of each transcoder. The run to which the CMs are referred is performed employing all the available annotations in the train set, exactly as the transcoding maps *100%* in *Figures* 7.6, 7.7, and 7.8.
Analyzing the CMs shown in *Appendix* D makes possible to understand how the classifiers work with respect to the target classes. The target classes are: *Forests*, *Streets*, *Fields*, *Urban*, and *Water*, as presented in *Section* 3.2. On the diagonal of the confusion matrices is shown the recall accuracy of each target class. The recall accuracy of a specific target class is calculated by dividing the number of correctly classified pixels of that class by the number of correctly classified pixels of that class plus the number of wrongly classified pixels of that class. In other terms, is the ratio between: the *True Positive*, and the summation of the *True positive* and the *False Negative*. The formulation of the recall accuracy is shown in *Equation* 7.1.

$$Recall\ Accuracy = \frac{True\ Positive}{True\ Positive + False\ Negative} \tag{7.1}$$

The last row and the last line of the matrices report the total number of classified pixels for each class shown on the x and y-axis. The values on the x-axis represent the number of predictions assigned to each class. The values on the y-axis represent the number of annotations belonging to each class.
The higher is the color intensity on the diagonal, the better is the capacity for the corresponding class. Instead, the higher is the intensity outside the diagonal, the higher is the confusion between the two interested classes.
Focusing on *Figures* 7.6, 7.7, and 7.8, it is evident that the CAT and the RT are the two best implementations and that are very resilient to the scarcity of labeled data. Unlike, the model pre-trained using the Cycle-AT performs worse at an absolute level, but is even more resilient to annotation lack. At this visual level, it is difficult to notice the differences between the Cycle-AT and the BL. But when the annotated data availability decreases, the Cycle-AT remains stable, instead the BL results are compromised. The CMs confirm these results: in general all the matrices present very accurate results for the classes *Urban*, *Water*, and *Forests*. In fact, all the implementations achieve results over 90% of recall accuracy for these classes. *Figure* 7.8 is a proof of the framework capacity to recognize these three classes.
The most problematic targets are *Fields* and *Highways*. Regarding the target class *Fields*, the CAT and the RT achieve around 80% of recall accuracy. Instead, the Cycle-AT and the BL achieve a recall accuracy lower

than the 40%. This behaviour is clearly noticeable in *Figure* 7.6. Most of the wrongly classified pixels of the target class *Fields* are misclassified, since they are assigned to the target class *Forests*. These two classes have common characteristics and are not trivial to be distinguished. In fact, as discussed in *Section* 7.2 the synthetic fields in the test image are often reconstructed as forests. The baseline, which is trained in a fully supervised fashion, presents the same problem. This suggests that distinguishing between *Forests* and *Fields* is not a trivial task, independently from the features extraction technique employed.
The target class *Highways* reports the worst results. In fact, the RT, which is the strongest classifier for this class, reaches a recall accuracy of 43%. The other classifiers do not exceed the 30 % of recall accuracy. The reason for these results can be ascribed to the fact that this class is unbalanced with respect to the others. As is it possible to see in *Section* 3.2 and in *Table* 3.1, the labeled pixels belonging to the class *Highways* are only 27 thousands. The other classes are made of an average of roughly 500 thousand pixels. It is possible that with more samples the algorithms would be able to correctly recognize this class. It is interesting to notice, in the CMs in *Appendix* D, that most of the pixels of the target class *Highways* are classified as *Fields*. This is a little bit counterintuitive, since what is expected is that the most similar class is *Urban*. This is an additional suggestion that, to better recognize pixels belonging to this class, is required a deeper investigation with a balanced dataset. *Figure* 7.7 shows clearly the difficulties of the classifiers in recognizing highways.
In conclusion, all the samples from the classification map (*Figures* 7.6, 7.7, and 7.8) show undoubtedly that the features extracted using all the transcoding strategies implemented make the algorithms resilient to annotation scarcity.

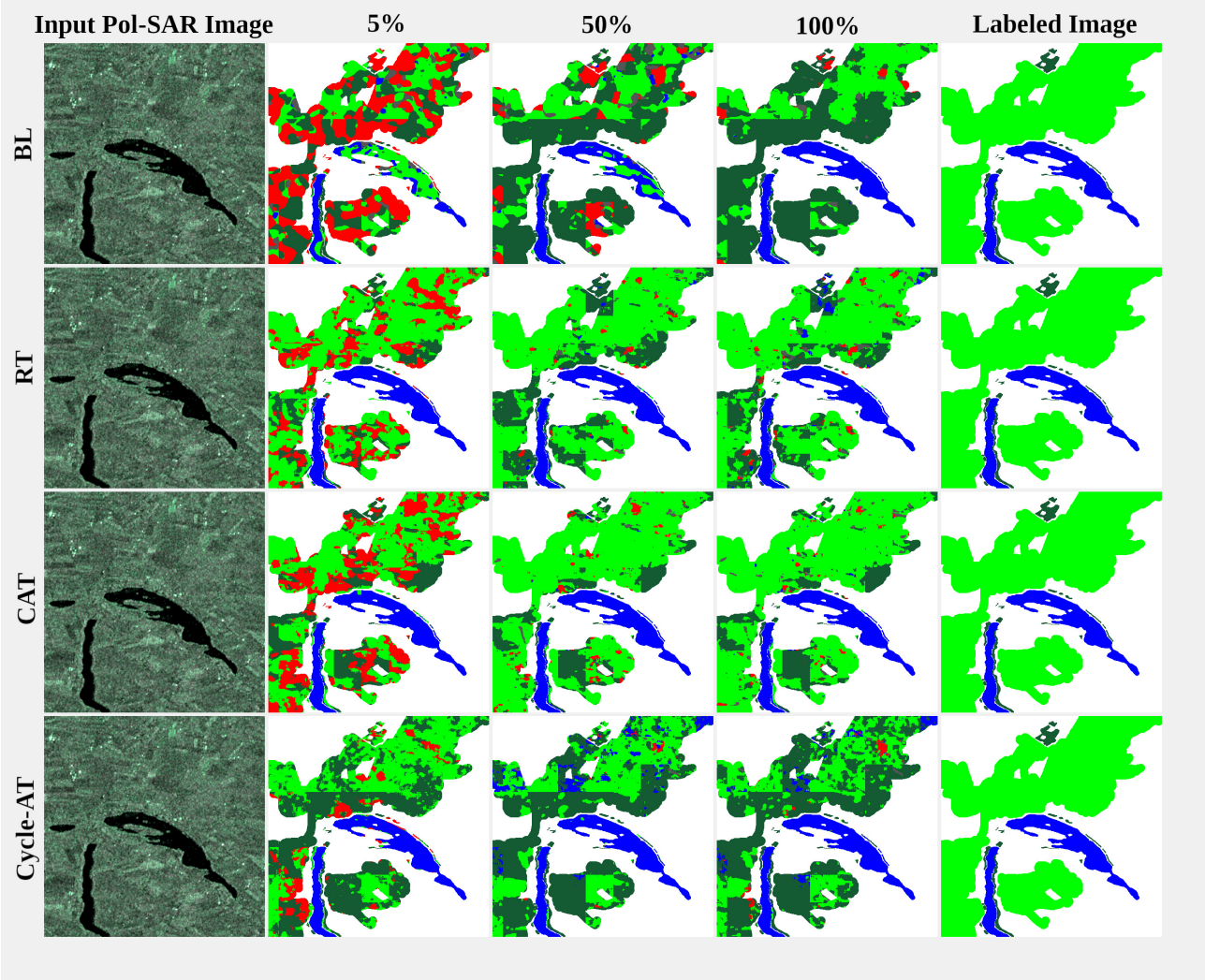


Figure 7.6: The image shows a sample from the classification map generated by the four classifiers implemented. Each row represents a different classifier. The first column represents the input Pol-SAR image. The three central columns represent the sample maps generated using every classifier trained with 5%, 50%, and 100% of all available annotated data. The last column shows the corresponding label map. This image contains classified areas which belong to *Fields*, *Water* and *Forests*. In the maps are shown: in dark green areas that represent *Forests*, in grey areas that represent *Streets*, in light green areas that represent *Fields*, in red areas that represent *Urban*, in blue areas that represent *Water*.

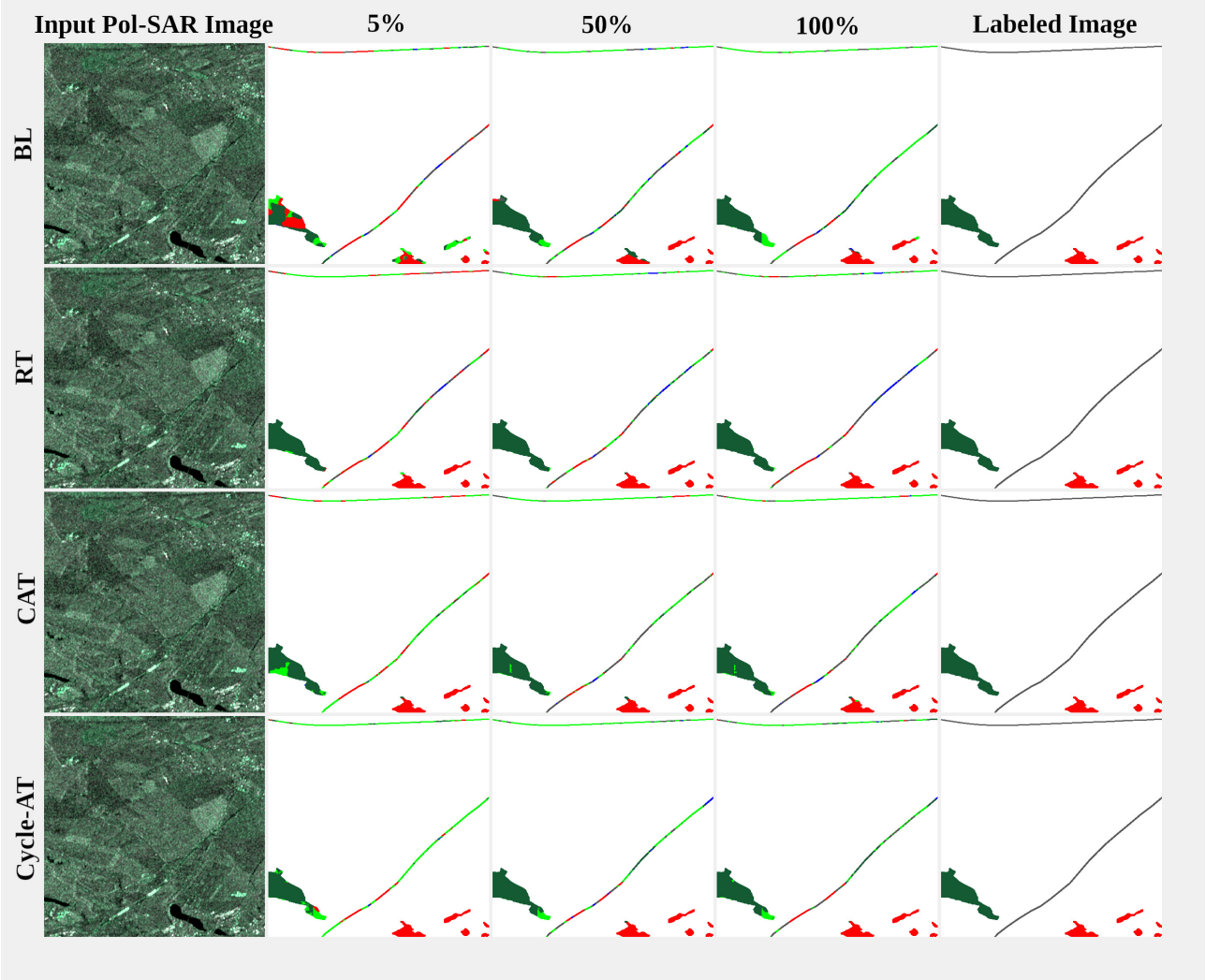


Figure 7.7: The image shows a sample from the classification map generated by the four classifiers implemented. Each row represents a different classifier. The first column represents the input Pol-SAR image. The three central columns represent the sample maps generated using every classifier trained with 5%, 50%, and 100% of all available annotated data. The last column shows the corresponding label map. This image contains classified areas which belong to *Forests*, *Highway* and *Urban*. In the maps are shown: in dark green areas that represent *Forests*, in grey areas that represent *Streets*, in light green areas that represent *Fields*, in red areas that represent *Urban*, in blue areas that represent *Water*.

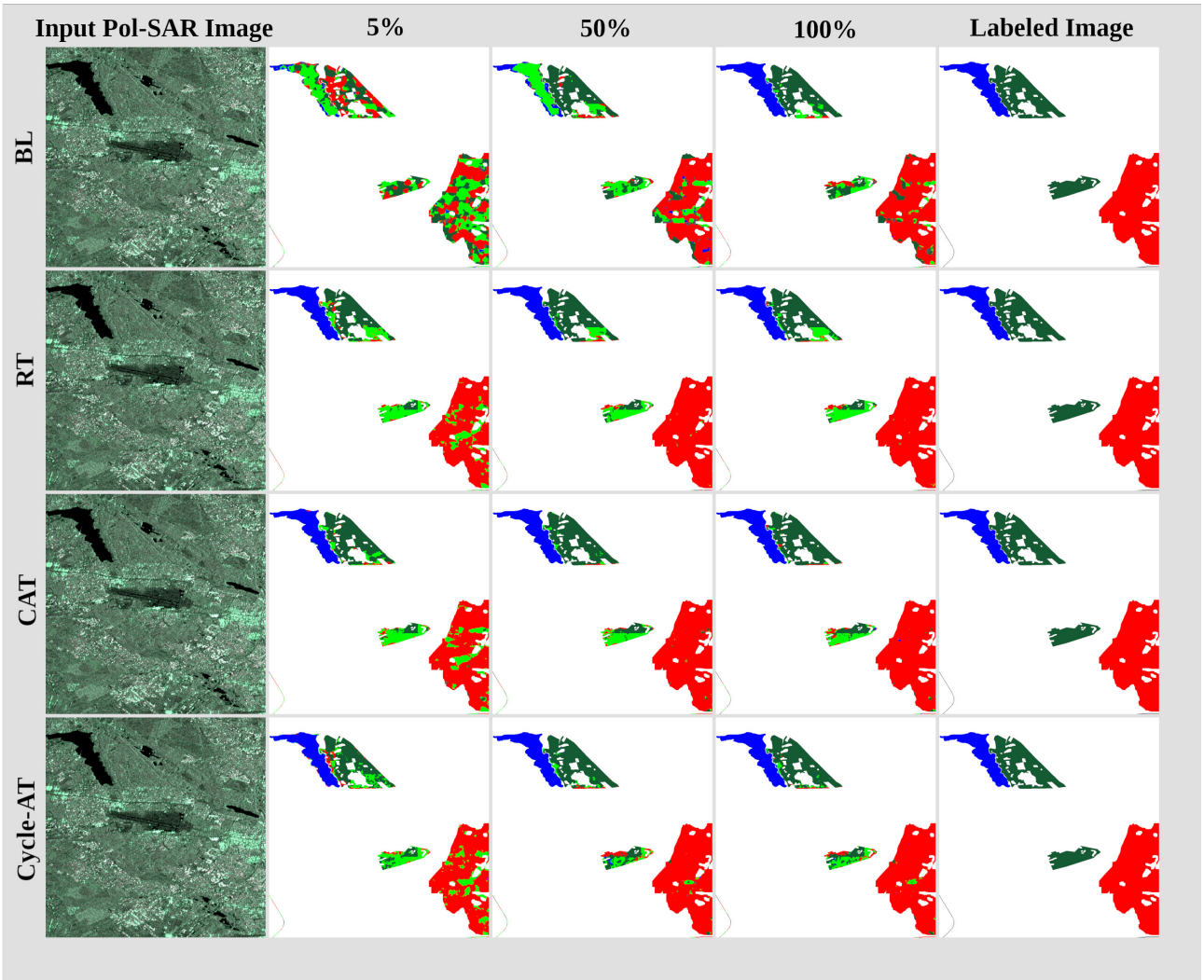


Figure 7.8: The image shows a sample from the classification map generated by the four classifiers implemented. Each row represents a different classifier. The first column represents the input Pol-SAR image. The three central columns represent the sample maps generated using every classifier trained with 5%, 50%, and 100% of all available annotated data. The last column shows the corresponding label map. This image contains classified areas which belong to *Forests*, *Highway*, *Water* and *Urban*. In the maps are shown: in dark green areas that represent *Forests*, in grey areas that represent *Streets*, in light green areas that represent *Fields*, in red areas that represent *Urban*, in blue areas that represent *Water*.

CHAPTER 8

Conclusions

Contents

8.1 Final Considerations 71
8.2 Future Works 72

8.1 Final Considerations

The final objective of this thesis is to implement a self-supervised deep learning framework. The aim is to investigate the capacity of the Pol-SAR to optical image transcoding to extract strong representative features for the semantic segmentation task of Pol-SAR images. This work is focused on the capacity of the self-supervision paradigm to leverage unlabeled data to deploy reliable classifiers, in particular when annotations are scarce and the procurement requires a significant effort. To understand the quality of the features extracted, using the self-supervision, has been trained the same ANN employing three different loss functions: a regressive loss, a conditional adversarial loss, and a cycle consistent adversarial loss. Then, each of these methods has been fine-tuned to perform the semantic segmentation task. Besides, has been developed a benchmarking ANN, exactly equal to the others, trained in a fully supervised fashion. The baseline has been designed to directly infer semantic segmentation maps of Pol-SAR images.
The features extraction capabilities have been qualitatively investigated by looking at the transcoding products. The best synthetic optical images have been generated using the CAT, which produces very sharp images, even if with some artifacts and aberrations. The RT generates good samples very similar to the ones produced using the CAT, but much more blurred. The Cycle-AT has some transcoding problems and visually does not create very realistic synthetic images, anyway it creates meaningful transcoding products.
Then, the features extraction capabilities have been investigated in terms of semantic segmentation accuracy. This framework achieves a top classification OA of 89%. This result has been achieved by fine-tuning the last layers of the CAT using all the available annotated data. The same architecture trained in a fully supervised fashion has achieved an overall accuracy of 73%. This is a very strong indication that the transcoding employed as proxy task extracts effective features for the classification of Pol-SAR images.
The RT extracted features, when employed to perform the semantic segmentation task, achieve a surprisingly 87% of overall accuracy. This result is only two percentage points minus the features extracted by the CAT. Considering that the RT is faster, simpler, and more stable with respect to the CAT, this is a very interesting insight. The classifier based on the features extracted using the Cycle-AT achieves 76% of overall accuracy.
Finally, has been investigated the resilience of the features extracted using the proxy task to the data annotation lacking. This work proves that the features learned, using all the pre-trained models implemented, guarantee a potent adaptability to label scarcity. The pre-trained models, using only the 5% of the total available annotations, lose only six points of OA. The RT and the CAT achieve more than 80% of overall accuracy, which is higher compared to the BL capacity when trained using all the available data.
The Cycle-AT does not achieve the performances of the other two concepts implemented. The reasons for these results can be ascribed to the nature of the training algorithm, which is much more complex. In fact, it

involves a double transformation: from the radar domain to the optical domain and vice-versa. Moreover, the two transformations are not balanced: the one from Pol-SAR to optical images is more complex than the one from optical to Pol-SAR images.
In conclusion, this thesis demonstrates that transcoding is a very effective features extraction technique for Pol-SAR images semantic segmentation in scenarios where very few labels are available. In addition, this work proves that simpler does not necessarily mean worse and the RT is the actual demonstration of this.

8.2 Future Works

In this thesis have been deeply exploited many aspects of the strategies implemented, nevertheless, there are still many unknowns to investigate. First of all, the next experiment to perform employing this framework is the verification of the Cycle-AT using unpaired datasets. As a matter of fact, the Cycle-AT already supports uncoupled Pol-SAR and optical images transcoding. Theoretically, using data as similar as possible to the one employed in this work, with the only difference of being uncoupled, could be possible to achieve the same results presented in this thesis. This confirmation would bring a big advantage: it would enable the possibility to train the Cycle-AT with much more data. Actually, the paired images would be no more a requirement. In this way, it could be possible to employ more data to improve the quality of the features extracted using this transcoder. And, more important, with a restrained data procurement effort. These argumentations strongly motivate the possibility to continue to investigate the topic presented in this thesis.
Another interesting investigation, that would be of paramount importance to verify the real potential of the implemented feature extractors, regards the semantic segmentation task. In fact, to verify the real strength and the generalization capacity of the features extracted, it would be gripping to check the capacity of the classifiers in two unexplored scenarios. The former concerns the use of images coming from different climatic areas and seasons. The latter concerns the verification of the discriminative capacity when employed to train datasets with more classes, less diverse, such as urban density levels.
If the above-presented investigations showed similar capacity to the ones achieved in this work, it would be possible to confirm that the feature extractors implemented are valid almost in any condition. This would make this method a very strong asset for the automatic analysis of Pol-SAR data.

Appendices

Appendix A

Scattering Matrix Filtering

Contents

A.1 Correct Filter Usage 75

A.2 Wrong Filter Usage 76

A.1 Correct Filter Usage

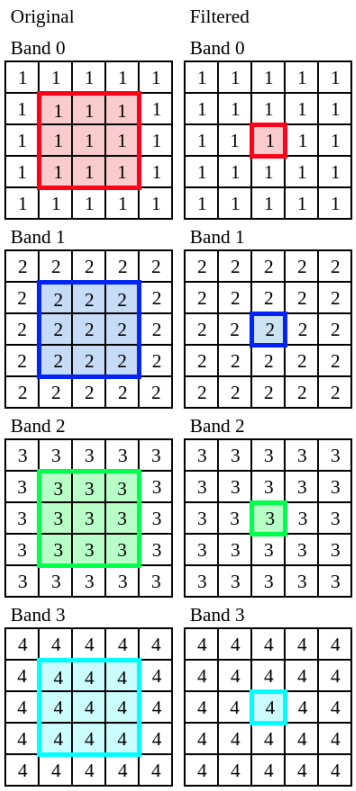


Figure A.1: The figure shows an example of the correct application of the box filter. In the left column are shown the results of the outer product between the scattering matrix and its complex conjugate transpose. The ensemble of the pixels, in the same position of each band, represents the resulting Hermitian matrices. On the right is shown the result of the filtering operation to a sample pixel. The box filter is represented by the colored masks that are two-dimensional and are applied separately to each band.

A.2 Wrong Filter Usage

Figure A.2: The figure shows an example of wrong application of the box filter. Each couple of *Original* and *Filtered* columns shows the filtering operation applied to the central pixel of a different band. The pixels interested by the filtering operation are highlighted with different colours. In the left column are always shown the results of the outer product between the scattering matrix and its complex conjugate transpose. The ensemble of the pixels, in the same position of each band, represents the resulting Hermitian matrices. On the right is shown the result of the filtering operation to a sample pixel. Unlike in *Appendix* A *Section* A.1, in this figure it is possible to see that the filter (the colored masks) is three-dimensional. The filter is applied not only to a band but to three at the same time, hence it is a box filter with size 3x3x3. This figure shows the filtered value of the same four pixels in *Appendix* A *Section* A.1

Appendix B

Original Data Tiles

Contents

B.1 Pol-SAR Train Tile 78
B.2 Optical Train Tile 78
B.3 Label Map Train Tile 79
B.4 Pol-SAR Test Tile 79
B.5 Optical Test Tile 80
B.6 Label Map Test Tile 80

B.1 Pol-SAR Train Tile

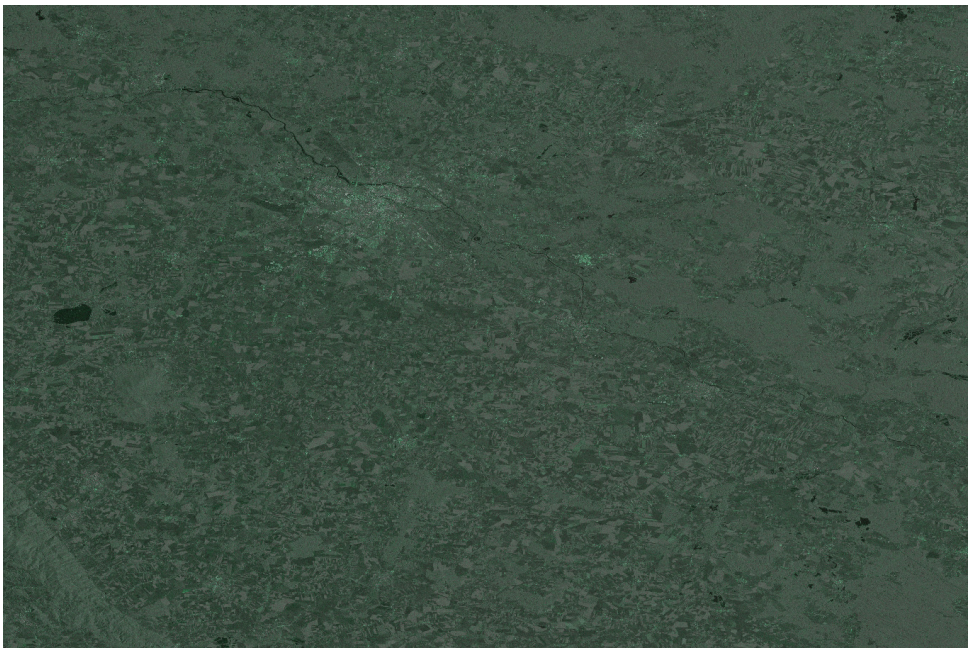


Figure B.1: The image is the original Pol-SAR train tile.

B.2 Optical Train Tile

Figure B.2: The image is the original optical train tile.

B.3 Label Map Train Tile

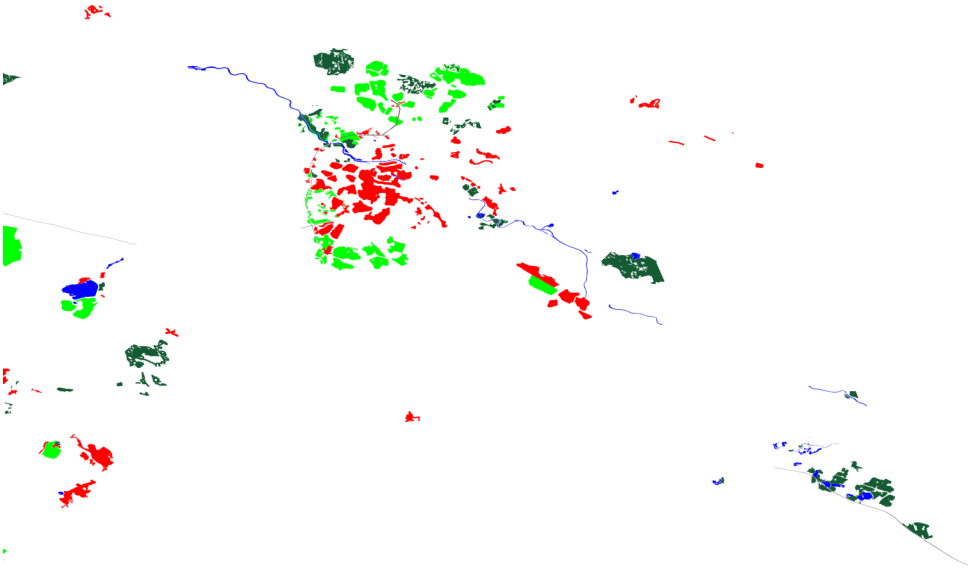


Figure B.3: The image is the label map of the train tile.

B.4 Pol-SAR Test Tile

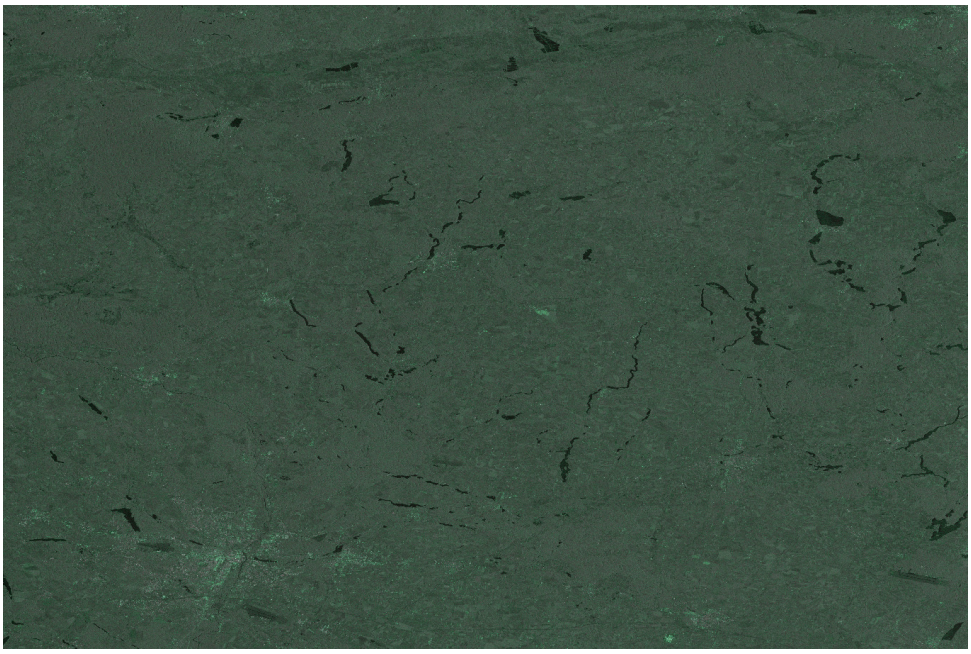


Figure B.4: The image is the original Pol-SAR test tile.

B.5 Optical Test Tile

Figure B.5: The image is the original optical test tile.

B.6 Label Map Test Tile

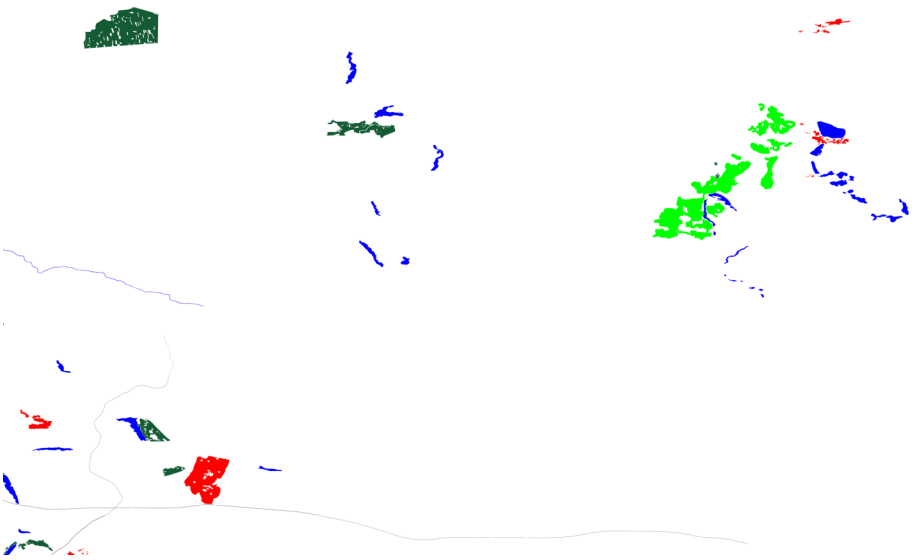


Figure B.6: The image is the label map of the test tile.

Appendix C

Full Transcoded Images

Contents

C.1 Train Tile Transcoding Result - Regressive Transcoder 82
C.2 Test Tile Transcoding Result - Regressive Transcoder 82
C.3 Train Tile Transcoding Result - Conditional Adversarial Transcoder 83
C.4 Test Tile Transcoding Result - Conditional Adversarial Transcoder 83
C.5 Train Tile Transcoding Result - Cycle Consistent Adversarial Transcoder 84
C.6 Test Tile Transcoding Result - Cycle Consistent Adversarial Transcoder 84

C.1 Train Tile Transcoding Result - Regressive Transcoder

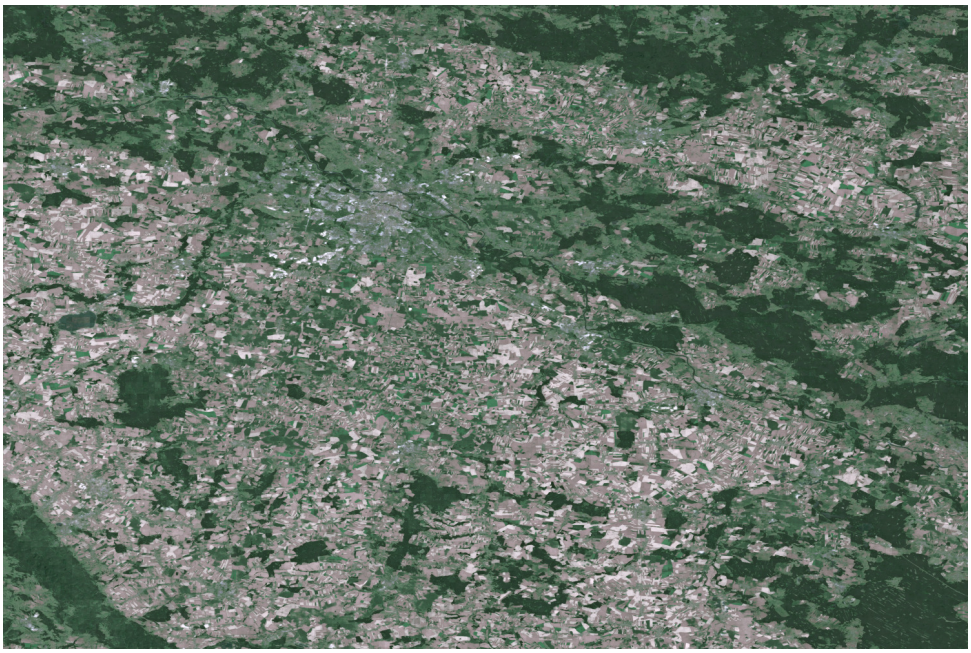


Figure C.1: The image is the synthetic optical train tile generated using the optimal RT.

C.2 Test Tile Transcoding Result - Regressive Transcoder

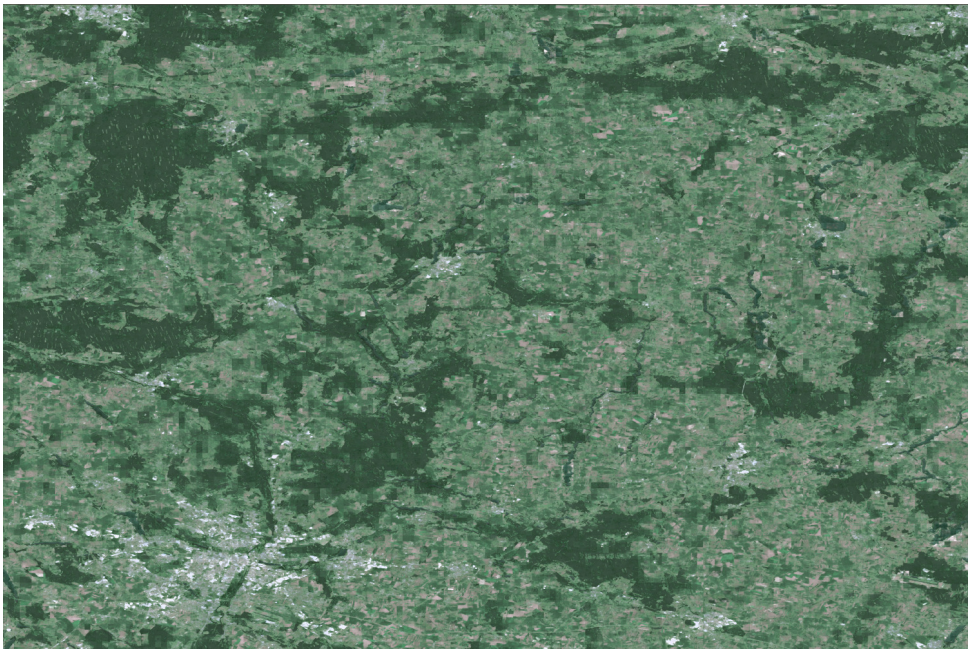


Figure C.2: The image is the synthetic optical test tile generated using the optimal RT.

C.3 Train Tile Transcoding Result - Conditional Adversarial Transcoder

Figure C.3: The image is the synthetic optical train tile generated using the optimal CAT.

C.4 Test Tile Transcoding Result - Conditional Adversarial Transcoder

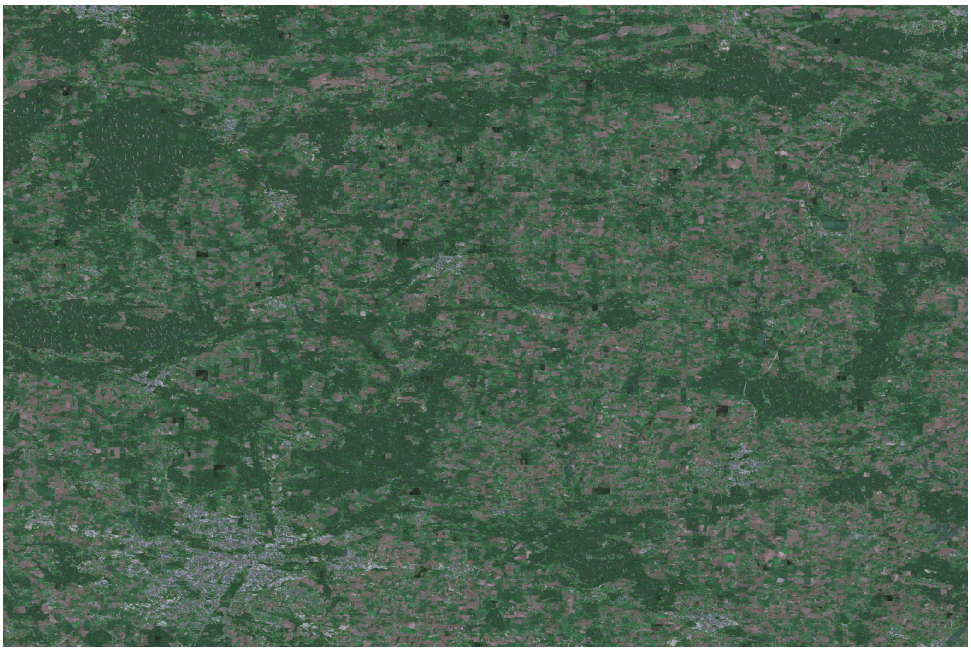


Figure C.4: The image is the synthetic optical test tile generated using the optimal CAT.

C.5 Train Tile Transcoding Result - Cycle Consistent Adversarial Transcoder

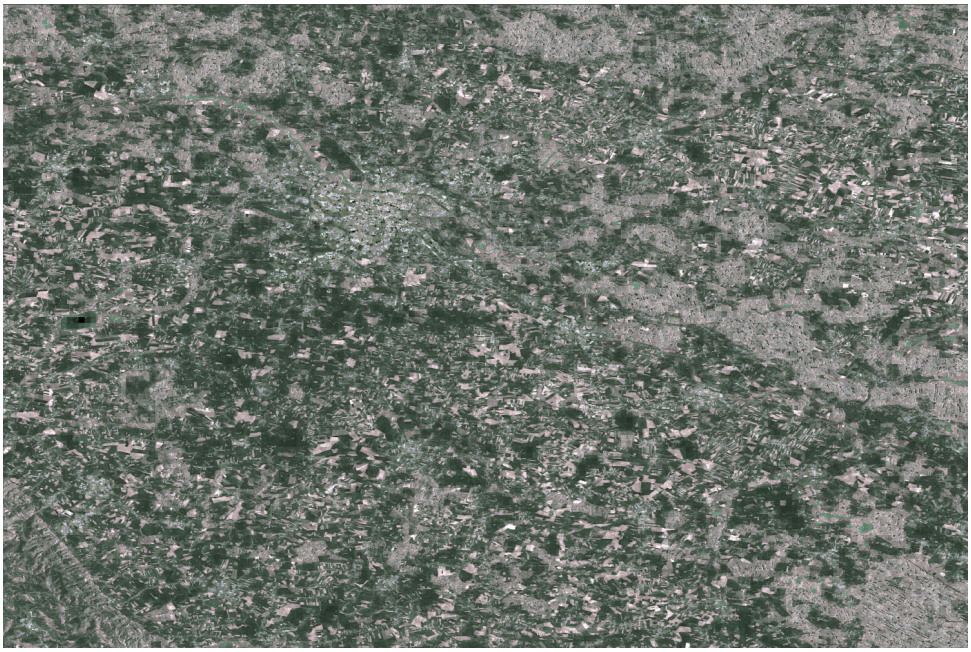


Figure C.5: The image is the synthetic optical train tile generated using the optimal Cycle-AT.

C.6 Test Tile Transcoding Result - Cycle Consistent Adversarial Transcoder

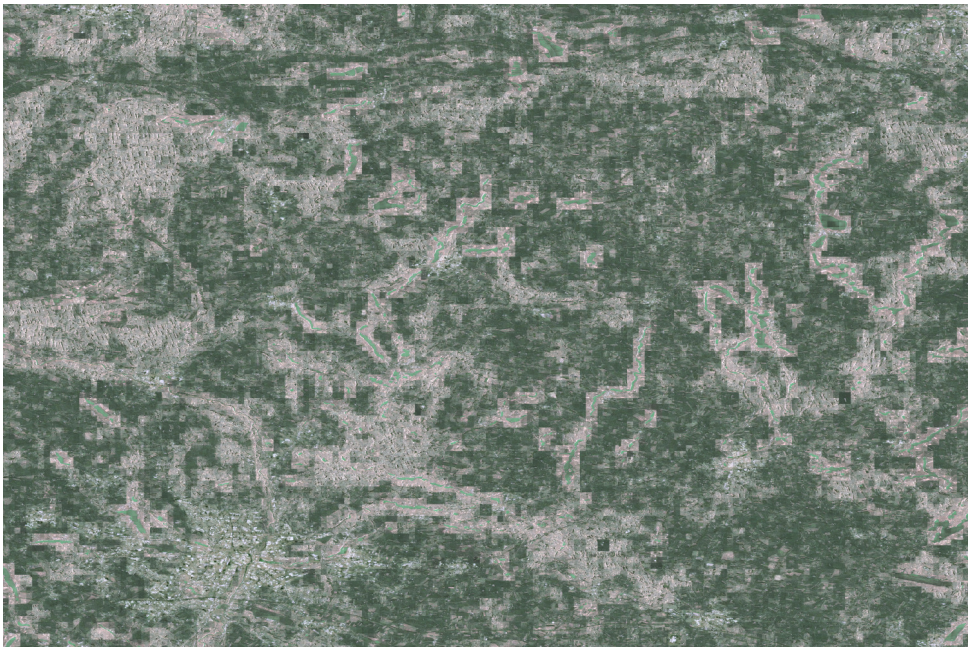


Figure C.6: The image is the synthetic optical test tile generated using the optimal Cycle-AT.

Appendix D

Confusion Matrices

Contents

D.1 Confusion Matrix - Baseline 86
D.2 Confusion Matrix - Regressive Transcoder 87
D.3 Confusion Matrix - Conditional Adversarial Transcoder 88
D.4 Confusion Matrix - Cycle Consistent Adversarial Transcoder 89

D.1 Confusion Matrix - Baseline

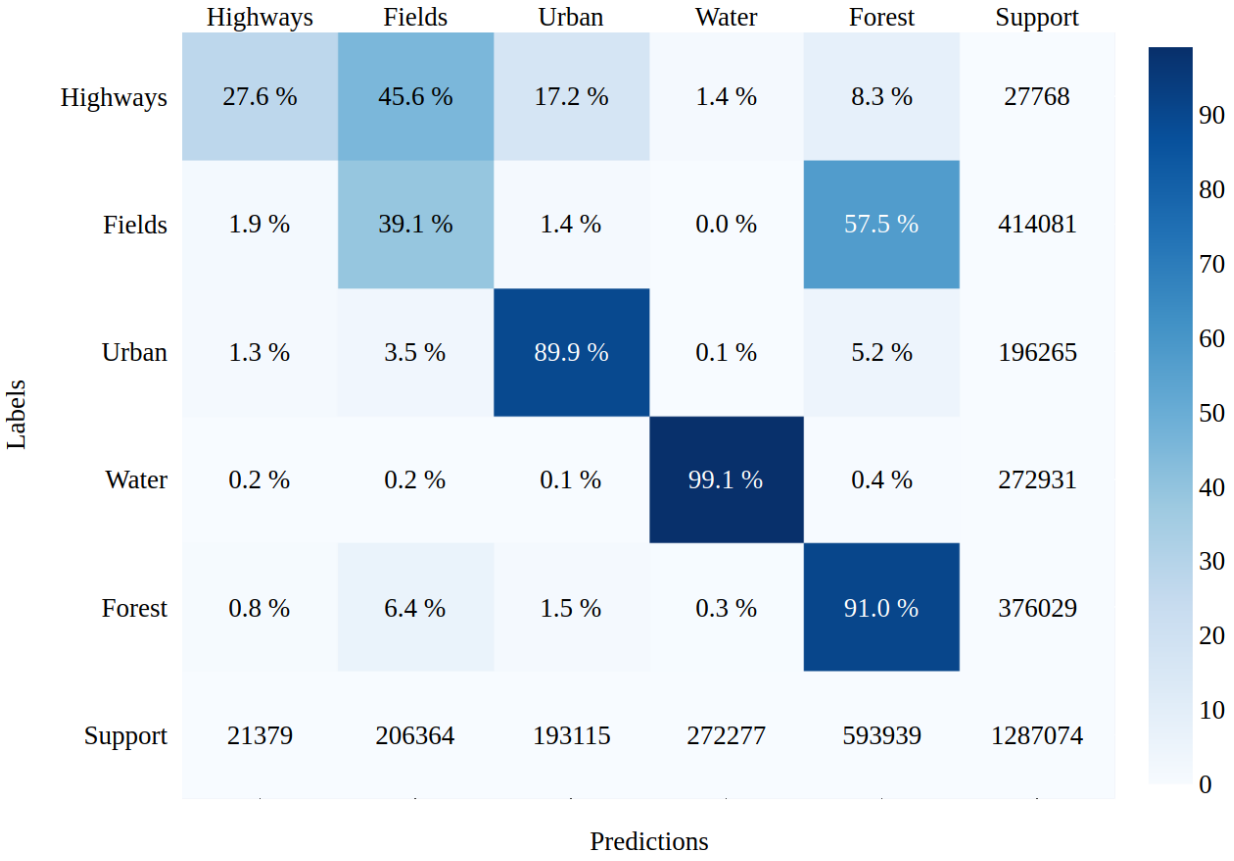


Figure D.1: The figure shows the confusion matrix of the classification map of the test tile, created using the BL classifier trained with all the annotations available. On the diagonal are reported the recall accuracies for each target class. The target classes are: *Forests*, *Streets*, *Fields*, *Urban*, and *Water*, as presented in *Section* 3.2. The x-axis represents the predictions for each target class. The y-axis represents the label for each target class. The higher is the intensity of the color on the diagonal, the better is the capacity of the algorithm. On the contrary, the higher is the color outside the diagonal, the higher is the classification error. The last line and the last row show the occurrences of the corresponding class both for labels and predictions.

D.2 Confusion Matrix - Regressive Transcoder

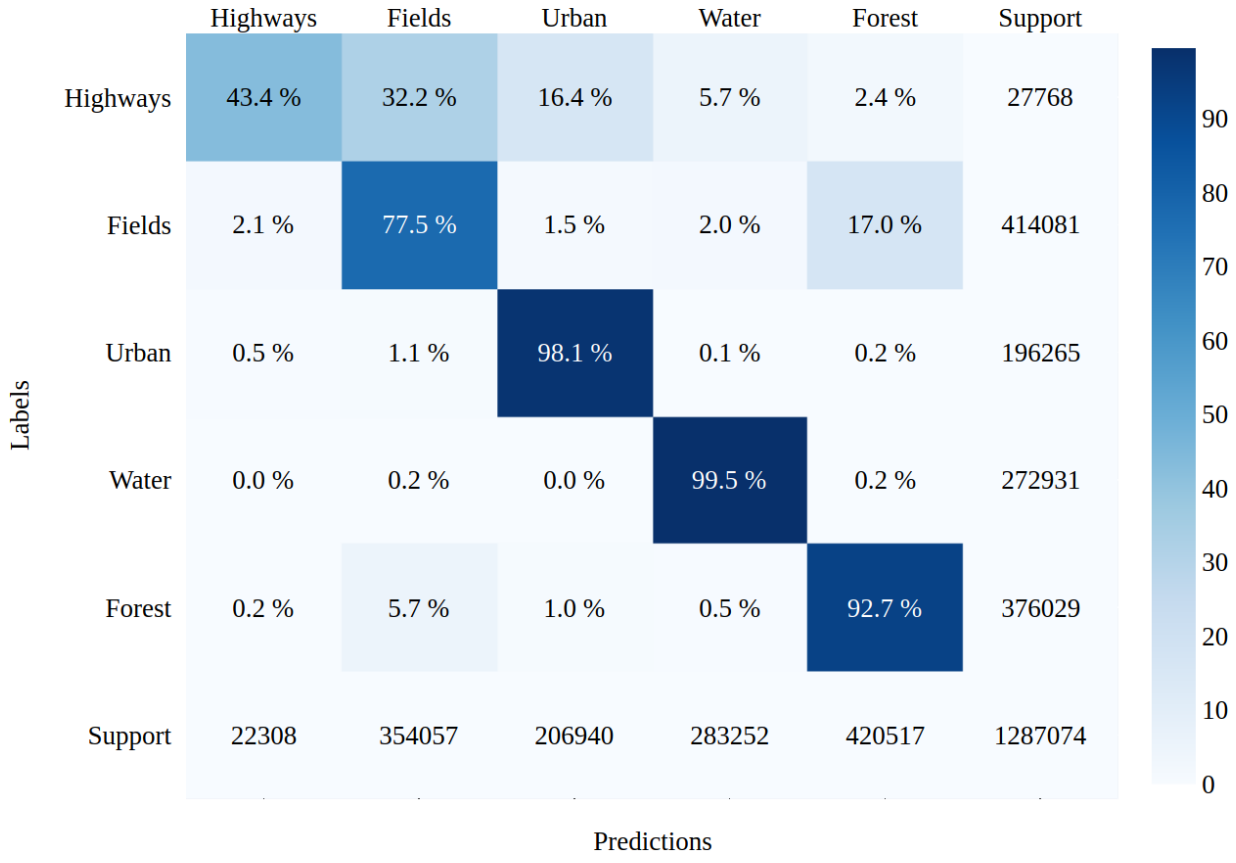


Figure D.2: The figure shows the confusion matrix of the classification map of the test tile, created using the RT with the last layers retrained with all the annotations available. On the diagonal are reported the recall accuracies for each target class. The target classes are: *Forests*, *Streets*, *Fields*, *Urban*, and *Water*, as presented in *Section* 3.2. The x-axis represents the predictions for each target class. The y-axis represents the label for each target class. The higher is the intensity of the color on the diagonal, the better is the capacity of the algorithm. On the contrary, the higher is the color outside the diagonal, the higher is the classification error. The last line and the last row show the occurrences of the corresponding class both for labels and predictions.

D.3 Confusion Matrix - Conditional Adversarial Transcoder

Labels \ Predictions	Highways	Fields	Urban	Water	Forest	Support
Highways	29.5 %	46.5 %	15.8 %	5.4 %	2.9 %	27768
Fields	1.8 %	80.0 %	1.3 %	0.4 %	16.6 %	414081
Urban	0.5 %	1.5 %	96.9 %	0.2 %	0.8 %	196265
Water	0.0 %	0.2 %	0.1 %	99.4 %	0.2 %	272931
Forest	0.1 %	3.1 %	0.8 %	0.3 %	95.7 %	376029
Support	17041	359137	203099	276425	431372	1287074

Figure D.3: The figure shows the confusion matrix of the classification map of the test tile, created using the CAT with the last layers retrained with all the annotations available. On the diagonal are reported the recall accuracies for each target class. The target classes are: *Forests*, *Streets*, *Fields*, *Urban*, and *Water*, as presented in *Section* 3.2. The x-axis represents the predictions for each target class. The y-axis represents the label for each target class. The higher is the intensity of the color on the diagonal the better is the capacity of the algorithm. On the contrary, the higher is the color outside the diagonal, the higher is the classification error. The last line and the last row show the occurrences of the corresponding class both for labels and predictions.

D.4 Confusion Matrix - Cycle Consistent Adversarial Transcoder

Labels \ Predictions	Highways	Fields	Urban	Water	Forest	Support
Highways	30.4 %	32.2 %	22.1 %	4.5 %	10.8 %	27768
Fields	0.9 %	37.8 %	0.9 %	4.3 %	56.2 %	414081
Urban	0.3 %	1.7 %	96.1 %	0.1 %	1.8 %	196265
Water	0.4 %	0.5 %	0.3 %	98.2 %	0.6 %	272931
Forest	0.3 %	4.0 %	1.0 %	1.8 %	93.0 %	376029
Support	15354	184755	202838	293911	590216	1287074

Figure D.4: The figure shows the confusion matrix of the classification map of the test tile, created using the Cycle-AT with the last layers retrained with all the annotations available. On the diagonal are reported the recall accuracies for each target class. The target classes are: *Forests*, *Streets*, *Fields*, *Urban*, and *Water*, as presented in *Section* 3.2. The x-axis represents the predictions for each target class. The y-axis represents the label for each target class. The higher is the intensity of the color on the diagonal the better is the capacity of the algorithm. On the contrary, the higher is the color outside the diagonal higher is the classification error. The last line and the last row show the occurrences of the corresponding class both for labels and predictions.

List of Figures

1.1 The left half of the image is a passive RGB image, the right half is an active radar image. . . . 2

2.1 A general scheme on the radars functioning [Bruzzone 2019]. 6
2.2 The image shows a satellite which rotates around the earth. In blue the areas interested by its sensor [Bruzzone 2017]. 7
2.3 Each position of the radar antenna (green squares) contributes to the quality of the single point *"P"*. Normally, the information belonging to the point *"P"* would be only acquired by the antenna in the central position. Applying the synthetic aperture is possible to achieve the resolution as if the antenna would be as long as from *x1* to *x2*. Actually, the synthetic aperture combines the echoes coming from the point *"P"* at each position of the antenna.[Flores-Anderson *et al.* 2019]. 8
2.4 Point A, B, C are equidistant on the target surface but due to the surface morphology they appear not equidistant to the sensor [Bruzzone 2017]. 9
2.5 The area between point B and C is not illuminated by the sensor [Bruzzone 2017]. 9
2.6 Point A results farther than point B in the SAR image [Bruzzone 2017]. 9
2.7 The image shows the detail of how a convolutional layer is composed. 14
2.8 The scheme shows the main elements of a GAN. z is the input noise to the generator. G is the generator network. G is fed with z and it outputs x'. x' is the synthetic/generated image and it is inputted to the discriminator. D is the discriminator which is fed with x and x' and it outputs if samples are real or fake. Based on its decisions the network is optimized. 15
2.9 The scheme shows the main elements of a cGAN. $x1$ is the input sample from one data distribution (1) given to the generator. G is the generator network fed with $x1$ that outputs $x2'$. $x2'$ is the synthetic/generated image that should belong to the second data distribution (2) and it is inputted to the discriminator. D is the discriminator which is fed with $x1$, $x2$ and $x2'$ and it outputs if samples are real or fake. Based on its decisions the network is optimized. 18
2.10 The figure shows the main elements for training the Cycle-GANs. $x1$ is the input sample from data distribution (1) given to the generator $G1$. There are two generators $G1$ and $G2$. $G1$ converts samples belonging to data distribution (1) to data distribution (2). $G2$ converts samples belonging to data distribution (2) to data distribution (1). $x2'$ is a synthetic sample that is given to $G2$. $G2$ converts $x2'$ back to domain (1). $G2$ outputs $x1''$, which is the reconstructed sample used to generate the cycle loss comparing it with the input $x1$. $D2$ is a discriminator trained to recognize if samples coming from data distribution (2) are real or fake. Based on the $D2$ output $G2$ parameters are optimized. This is the so-called adversarial loss. This image represents only the process related to the training of $G1$, but it is necessary to perform the same operations for $G2$. $G2$ requires a cyclic loss between $x2$ and $x2''$. Besides an additional discriminator, called $D1$, used to recognize if samples coming from (2) are real or fake. 19

3.1 *Figure a)* and *Figure b)* show a sample from the training data tiles. 24
3.2 The image shows a sample from the label map train tile. Dark Green areas represent *Forests*, grey areas represent *Streets*, light green areas represent *Fields*, red areas represent *Urban*, blue areas represent *Water*. 26

4.1 The image shows the architecture of the generator. 29

4.2 The image shows the architecture of the Discriminator. 30

4.3 The image shows the generator architecture of *Figure* 4.1. The greyed blocks are fixed during the classification phase of pre-trained models. The remaining colored layers are the ones retrained during the fine-tuning phase. 31

4.4 The image illustrates how the BL is trained to perform the semantic segmentation task. There is a Pol-SAR image in input to a generic generator *G*. The generator parameters are optimized based on the error between the generated map and the semantic segmentation annotation. The classification error is calculated using the cross entropy. 33

4.5 The image illustrates how the regressive transcoder is trained. There is a Pol-SAR image inputted to a generic generator *G*. The generator parameters are optimized with respect to the error between the generated optical image and the real optical image. The difference between the two images is calculated using the *L1Loss*. 34

4.6 The scheme shows how the CAT is trained. The network *G* is fed with a Pol-SAR image. *G* parameters are updated based on the *L1* loss between synthetic and real optical images, plus the error of the discriminator *D* to recognize that the generated sample was actually fake. This training strategy, excluding the adversarial loss, is equal to the RT discussed in *Subsection* 4.2.2. The regression loss is a very good starting point to convert Pol-SAR images to optical ones. The main limitation of the *L1* loss is that the generated images are blurred. In fact, the regressive loss is very good at reproducing low frequencies, but cannot reproduce high frequencies with the same accuracy. Low frequencies contain low variational characteristics, instead high frequencies contain high variational information. As a result, the former well describes the image at a high level, the latter well describes sharp details. In *Figure* 7.3 this phenomena is very clear. This is the reason why the adversarial term is introduced. The adversarial term raises the pressure on the generator. The generator has to learn to reproduce high-frequency details to optimize the adversarial loss term. The adversarial term is an additional force that pushes the generator to improve itself. 36

4.7 *Figure a)* and *Figure b)* show the framework to train the two Cycle-GAN generators. Each generator is trained using the loss of the reconstruction error and using the loss of a dedicated discriminator. In *Figure a)* the *Optical to Pol-SAR generator* transcodes the inputted optical image. The Pol-SAR generated image is: 1) reconverted to optical using the *Pol-SAR to Optical generator*, 2) given to the *Pol-SAR discriminator* to generate the fake or real label. The reconstructed image is used to optimize the generator based on the *L1* loss with respect to the original input. The discriminator output is used to calculate the adversarial loss to optimize the generator using the *MSE* loss. *Figure b)* shows the same process but for the *Pol-SAR to optical generator*. If we compare the Cycle-AT with CAT, beyond the complexity difference, conceptually the setup is very similar. The training process is composed of a regressive term, which should model the low frequencies of the image, and an adversarial term, which should model the high frequencies of the image. The difference lies in the double conversion applied to the input data. The double conversion is designed to empower the capacity to compare the generated sample directly with the input image. This strategy removes the constraint to have the real version of the transcoded input image. This feature introduces a marked complexity increasing. Firstly it doubles the framework parameters which augment the computational load. Secondly, the learning process is more complex due to the double transformation. The objective is not to transform an image from domain *A* to domain *B*. The objective of the Cycle-AT is to transform images from domain *A* to domain *B* to domain *A*. This double conversion is very challenging. . 38

5.1 *Figure a)* and *Figure b)* show the coloured visualisation of vector v channels 0, 1, 2 correctly and wrongly filtered, respectively. 42

6.1 *Figure a)* shows a patch with 128×128 pixels and *Figure b)* shows a patch with 256×256 pixels. The images are presented with the same dimension but a different number of pixels, as a fact the *Figure b)* has higher resolution. 50

7.1 The image shows the synthetic optical train tile generated using the optimal CAT. 55

7.2 The image shows the synthetic optical test tile generated using the optimal CAT. 55

7.3 The figure shows three different samples from the test tile, each sample is shown in a different row. Each column represents: 1) the Pol-SAR input features vector (the first three features of the vector are rendered as RGB channels), 2) the column 1) transcoded version using the RT, 3) the column 1) transcoded version using the CAT, 4) the column 1) transcoded version using the Cycle-AT, 5) the real optical version of the input 1). 57

7.4 The image represents the OA of the four classification networks implemented after each training epoch. In the legend: [BL] refers to the results achieved using the BL, [RT] refers to the results achieved using the RT with the last layers retrained, [CAT] refers to the results achieved using the CAT with the last layers retrained, [Cycle-AT] refers to the results achieved using the Cycle-AT with the last layers retrained. The detailed process of fine-tuning is discussed in *Section* 4.1.3. 61

7.5 The image represents the OA accuracy of the four classification networks implemented. For each implementation is plotted the top-performing accuracy achieved using a different amount of labeled data. Each group of bars represents the accuracy achieved with each implementation using the corresponding percentage of labeled data shown on the x-axis. The bar heights represent the average value, the grey lines represent the standard deviation achieved performing ten measures for each bar represented. Every measure differs for the combination of samples in the training subset. For each percentage investigation are used the same ten dataset combinations. In the legend: [BL] refers to the results achieved using the BL, [RT] refers to the results achieved using the RT with the last layers retrained, [CAT] refers to the results achieved using the CAT with the last layers retrained, [Cycle-AT] refers to the results achieved using the Cycle-AT with the last layers retrained. 65

7.6 The image shows a sample from the classification map generated by the four classifiers implemented. Each row represents a different classifier. The first column represents the input Pol-SAR image. The three central columns represent the sample maps generated using every classifier trained with 5%, 50%, and 100% of all available annotated data. The last column shows the corresponding label map. This image contains classified areas which belong to *Fields*, *Water* and *Forests*. In the maps are shown: in dark green areas that represent *Forests*, in grey areas that represent *Streets*, in light green areas that represent *Fields*, in red areas that represent *Urban*, in blue areas that represent *Water*. 68

7.7 The image shows a sample from the classification map generated by the four classifiers implemented. Each row represents a different classifier. The first column represents the input Pol-SAR image. The three central columns represent the sample maps generated using every classifier trained with 5%, 50%, and 100% of all available annotated data. The last column shows the corresponding label map. This image contains classified areas which belong to *Forests*, *Highway* and *Urban*. In the maps are shown: in dark green areas that represent *Forests*, in grey areas that represent *Streets*, in light green areas that represent *Fields*, in red areas that represent *Urban*, in blue areas that represent *Water*. 69

7.8 The image shows a sample from the classification map generated by the four classifiers implemented. Each row represents a different classifier. The first column represents the input Pol-SAR image. The three central columns represent the sample maps generated using every classifier trained with 5%, 50%, and 100% of all available annotated data. The last column shows the corresponding label map. This image contains classified areas which belong to *Forests*, *Highway*, *Water* and *Urban*. In the maps are shown: in dark green areas that represent *Forests*, in grey areas that represent *Streets*, in light green areas that represent *Fields*, in red areas that represent *Urban*, in blue areas that represent *Water*. 70

A.1 The figure shows an example of the correct application of the box filter. In the left column are shown the results of the outer product between the scattering matrix and its complex conjugate transpose. The ensemble of the pixels, in the same position of each band, represents the resulting Hermitian matrices. On the right is shown the result of the filtering operation to a sample pixel. The box filter is represented by the colored masks that are two-dimensional and are applied separately to each band. 75

A.2 The figure shows an example of wrong application of the box filter. Each couple of *Original* and *Filtered* columns shows the filtering operation applied to the central pixel of a different band. The pixels interested by the filtering operation are highlighted with different colours. In the left column are always shown the results of the outer product between the scattering matrix and its complex conjugate transpose. The ensemble of the pixels, in the same position of each band, represents the resulting Hermitian matrices. On the right is shown the result of the filtering operation to a sample pixel. Unlike in *Appendix* A *Section* A.1, in this figure it is possible to see that the filter (the colored masks) is three-dimensional. The filter is applied not only to a band but to three at the same time, hence it is a box filter with size 3x3x3. This figure shows the filtered value of the same four pixels in *Appendix* A *Section* A.1 76

B.1 The image is the original Pol-SAR train tile. 78
B.2 The image is the original optical train tile. 78
B.3 The image is the label map of the train tile. 79
B.4 The image is the original Pol-SAR test tile. 79
B.5 The image is the original optical test tile. 80
B.6 The image is the label map of the test tile. 80

C.1 The image is the synthetic optical train tile generated using the optimal RT. 82
C.2 The image is the synthetic optical test tile generated using the optimal RT. 82
C.3 The image is the synthetic optical train tile generated using the optimal CAT. 83
C.4 The image is the synthetic optical test tile generated using the optimal CAT. 83
C.5 The image is the synthetic optical train tile generated using the optimal Cycle-AT. 84
C.6 The image is the synthetic optical test tile generated using the optimal Cycle-AT. 84

D.1 The figure shows the confusion matrix of the classification map of the test tile, created using the BL classifier trained with all the annotations available. On the diagonal are reported the recall accuracies for each target class. The target classes are: *Forests*, *Streets*, *Fields*, *Urban*, and *Water*, as presented in *Section* 3.2. The x-axis represents the predictions for each target class. The y-axis represents the label for each target class. The higher is the intensity of the color on the diagonal, the better is the capacity of the algorithm. On the contrary, the higher is the color outside the diagonal, the higher is the classification error. The last line and the last row show the occurrences of the corresponding class both for labels and predictions. 86

D.2 The figure shows the confusion matrix of the classification map of the test tile, created using the RT with the last layers retrained with all the annotations available. On the diagonal are reported the recall accuracies for each target class. The target classes are: *Forests*, *Streets*, *Fields*, *Urban*, and *Water*, as presented in *Section* 3.2. The x-axis represents the predictions for each target class. The y-axis represents the label for each target class. The higher is the intensity of the color on the diagonal, the better is the capacity of the algorithm. On the contrary, the higher is the color outside the diagonal, the higher is the classification error. The last line and the last row show the occurrences of the corresponding class both for labels and predictions. . 87

D.3 The figure shows the confusion matrix of the classification map of the test tile, created using the CAT with the last layers retrained with all the annotations available. On the diagonal are reported the recall accuracies for each target class. The target classes are: *Forests*, *Streets*, *Fields*, *Urban*, and *Water*, as presented in *Section* 3.2. The x-axis represents the predictions for each target class. The y-axis represents the label for each target class. The higher is the intensity of the color on the diagonal the better is the capacity of the algorithm. On the contrary, the higher is the color outside the diagonal, the higher is the classification error. The last line and the last row show the occurrences of the corresponding class both for labels and predictions. 88

D.4 The figure shows the confusion matrix of the classification map of the test tile, created using the Cycle-AT with the last layers retrained with all the annotations available. On the diagonal are reported the recall accuracies for each target class. The target classes are: *Forests*, *Streets*, *Fields*, *Urban*, and *Water*, as presented in *Section* 3.2. The x-axis represents the predictions for each target class. The y-axis represents the label for each target class. The higher is the intensity of the color on the diagonal the better is the capacity of the algorithm. On the contrary, the higher is the color outside the diagonal higher is the classification error. The last line and the last row show the occurrences of the corresponding class both for labels and predictions. . 89

List of Tables

3.1 The table shows the number of labeled pixels for each class in the dataset. The top column shows one next to the other the number of classified pixels in the train and test sets. In the last row the total number of labelled pixels is presented. 25

3.2 The table shows the number of samples for every dataset configuration. 26

4.1 The table shows how many parameters has every network implemented in this work. The last line *Evaluation* shows the number of parameters of the generator once trained and used to perform semantic segmentation. The evaluation complexity is independently from the pre-trained model employed. 39

5.1 The table shows the semantic segmentation results using BL, the RT, the CAT and the Cycle-AT. For each implementation is proposed the top semantic segmentation OA using the Wrongly-FD and using Correctly-FD. The last column shows in green the improvement achieved using the correctly filtered data. 43

6.1 The table shows the computational time to perform an optimization step using a patch size of 128×128 pixels and a batch size of 1 for each ANN implemented. The BL and RT, as expected, take roughly the same time to perform an optimization step. The CAT is slightly more complex, nevertheless, to perform an optimization step, it takes nearly the same time of the BL and RT. This is because as discussed in *Subsection* 6.4.2 the discriminator is trained every K optimization steps of the generator. The Cycle-AT takes double the time with respect to the other implementations. As a fact, it implements the transcoding task two times. One for the Pol-SAR to optical transcoding and one for the inverse process. The last line of the table shows the time required to generate the semantic segmentation map of a Pol-SAR patch of size 128×128 pixels. As is possible to see this operation is much faster with respect to the others. This is because in evaluation mode the algorithm does not perform the optimization steps. The network only infers the predictions for each pixel. This amount of time is independent of the type of feature extractor employed. 46

6.2 The table shows the semantic segmentation results using BL, the RT, the CAT and the Cycle-AT. The first line of each implementation shows the result using a patch size of 128×128 pixels. The second line of each implementation shows the result using a patch size of 256×256 pixels. In red is presented the variation of the accuracy changing the patch size. The arrow indicates the decreasing direction. 50

6.3 The table shows the semantic segmentation results using BL, the RT, the CAT and the Cycle-AT. The first line of each implementation shows the OA accuracy using nine residual blocks in the backbone of the generator (the architecture details are discussed in *Subsection* 4.1.1). The second line of each implementation shows the result using six residual blocks in the backbone of the generator. The colored numbers describe the variation between the two versions of each classification method. In red, the accuracy values decreased using six residual blocks instead of nine, in orange when there is no variation (only small fluctuations), and in green if the accuracy has increased. The arrows indicate the increasing/decreasing directions. The tilde represents no significant changes. 51

6.4 The table shows the semantic segmentation results using CAT and the Cycle-AT. The first line of each implementation shows the OA accuracy using a discriminator architecture with three downsampling convolutional layers (the discriminator architecture details are discussed in *Subsection* 4.1.2). The second line of each implementation shows the result using two downsampling convolutional layers in the discriminator. The colored numbers describe the variation between the two versions of each classification method. In red, the accuracy value decreased using two downsampling layers instead of three, and in orange when there is no variation (only small fluctuations). The arrow indicates the decreasing direction. The tilde represents no significant changes. 52

7.1 The table shows the configuration for all the final tests performed presented in *Chapter* 7. The choice of these hyperparameters is discussed in *Subsection* 6.5. 53

7.2 The table shows the final accuracy levels reached using the four classifiers implemented. The accuracy values reported are the average value achieved performing ten times the training process. The last column shows the standard deviation of the values calculated over the ten repetitions performed. 59

7.3 The table shows the subsets used to perform the sensitivity analysis presented in *Subsection* 7.3.3. The first column shows the name of the subsets, which also represents the percentage of total available annotation employed to train the classifiers. The second column shows the corresponding number of training patches with size 128×128 pixels. 63

List of Acronyms

SEN1-2 Sentinel-1, Sentinel-2 1

ESA European Space Agency 1

NASA National Aeronautics and Space Administration 1

RGB Red, Green, Blue 2

Pol-SAR Polarimetric Synthetic Aperture Radar 2

GAN Generative Adversarial Network 3

Cycle-GAN Cycle Consistent Generative Adversarial Network 3

SAR Synthetic Aperture Radar 8

DLR German Aerospace Center 8

TSX TerraSAR-X 8

ANN Artificial Neural Network 10

ML Machine Learning 10

DL Deep Learning 10

CNN Convolutional Neural Network 12

cGAN Conditional Generative Adversarial Network 17

MLP Multilayer Perceptrons 19

RF Random Forest 19

DANN Deep Artificial Neural Network 20

ResNet Residual Network 21

SEN-1 Sentinel-1 23

SEN-2 Sentinel-2 23

SNAP Sentinel Application Platform 23

NIR Near-Infrared 23

DEM Digital Elevation Model 23

BL Baseline 31

RT Regressive Transcoder 34

CAT Conditional Adversarial Transcoder 35

Cycle-AT Cycle Consistent Adversarial Transcoder 37

Wrongly-FD Wrongly Filtered Pol-SAR Data 42

Correctly-FD Correctly Filtered Pol-SAR Data 42

OA Overall Accuracy

FIFO First In First Out 47

GPU Graphic Processing Unit 46

CPU Central Processing Unit 46

WGAN Wasserstein Generative Adversarial Network 58

MCCV Monte Carlo Cross Validation 62

CM Confusion Matrix 66

Bibliography

[Arjovsky *et al.* 2017] Martin Arjovsky, Soumith Chintala and Léon Bottou. *Wasserstein Generative Adversarial Networks*. In Doina Precup and Yee Whye Teh, editors, Proceedings of the 34th International Conference on Machine Learning, volume 70 of *Proceedings of Machine Learning Research*, pages 214–223, International Convention Centre, Sydney, Australia, 06–11 Aug 2017. PMLR. (Cited on page 58.)

[Blum & Mitchell 1998] A. Blum and T. Mitchell. *Combining labeled and unlabeled data with co-training*. In COLT' 98, 1998. (Cited on page 11.)

[Bruzzone *et al.* 2004] L. Bruzzone, M. Marconcini, U. Wegmüller and A. Wiesmann. *An advanced system for the automatic classification of multitemporal SAR images*. IEEE Transactions on Geoscience and Remote Sensing, vol. 42, pages 1321–1334, 2004. (Cited on page 19.)

[Bruzzone 2017] Lorenzo Bruzzone. *Acquisition of Remote Sensing Images*. Course of Introduction to Remote Sensing, jan 2017. (Cited on pages 7, 9 and 91.)

[Bruzzone 2019] Lorenzo Bruzzone. *Introduction to Active Remote Sensing Systems*. Course of Radar and Radiolocalization, jan 2019. (Cited on pages 6 and 91.)

[Caron *et al.* 2018] M. Caron, P. Bojanowski, Armand Joulin and M. Douze. *Deep Clustering for Unsupervised Learning of Visual Features*. In ECCV, 2018. (Cited on page 12.)

[Cattoi 2021] Alessandro Cattoi. *Self-Supervised Learning for Semantic Segmentation of Pol-SAR Images Via SAR-to-Optical Transcoding*, feb 2021. Accessed on 27/02/2021. (Cited on page 27.)

[Durall *et al.* 2020] Ricard Durall, Avraam Chatzimichailidis, Peter Labus and Janis Keuper. *Combating Mode Collapse in GAN training: An Empirical Analysis using Hessian Eigenvalues*, 2020. (Cited on page 56.)

[esa.int a] esa.int. *Sentinel-1*. `https://sentinel.esa.int/web/sentinel/missions/sentinel-1`. Accessed on 25/01/2021. (Cited on pages 1 and 23.)

[esa.int b] esa.int. *Sentinel-2*. `https://sentinel.esa.int/web/sentinel/missions/sentinel-2`. Accessed on 02/02/2021. (Cited on pages 1 and 23.)

[Flores-Anderson *et al.* 2019] A. Flores-Anderson, K. E. Herndon, R. B. Thapa and E.Cherrington. The synthetic aperture radar (sar) handbook: Comprehensive methodologies for forest monitoring and biomass estimation. SERVIR Global Science Coordination Ofce National Space Science and Technology Center, 320 Sparkman Drive, Huntsville, AL 35805, 2019. (Cited on pages 7, 8, 9 and 91.)

[Gidaris *et al.* 2018] Spyros Gidaris, Praveer Singh and Nikos Komodakis. *Unsupervised Representation Learning by Predicting Image Rotations*. ArXiv, vol. abs/1803.07728, 2018. (Cited on page 12.)

[Goodfellow *et al.* 2014] Ian J. Goodfellow, Jean Pouget-Abadie, M. Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron C. Courville and Yoshua Bengio. *Generative Adversarial Nets*. In NIPS, 2014. (Cited on pages 12, 15, 16 and 17.)

[Goodfellow *et al.* 2016] Ian Goodfellow, Yoshua Bengio and Aaron Courville. Deep learning. MIT Press, 2016. `http://www.deeplearningbook.org`. (Cited on pages 10, 11 and 20.)

[Hänsch & Hellwich 2010] R. Hänsch and O. Hellwich. *Random Forests for building detection in polarimetric SAR data*. 2010 IEEE International Geoscience and Remote Sensing Symposium, pages 460–463, 2010. (Cited on page 19.)

[Hänsch & Hellwich 2017] R. Hänsch and O. Hellwich. *Skipping the real world: Classification of PolSAR images without explicit feature extraction*. Isprs Journal of Photogrammetry and Remote Sensing, vol. 140, pages 122–132, 2017. (Cited on pages 2, 8 and 20.)

[Hänsch & Hellwich 2018] R. Hänsch and O. Hellwich. *A Comparative Evaluation of Polarimetric Distance Measures within the Random Forest Framework for the Classification of Polsar Images*. IGARSS 2018 - 2018 IEEE International Geoscience and Remote Sensing Symposium, pages 8440–8443, 2018. (Cited on pages 42 and 43.)

[Hänsch 2010] R. Hänsch. *Complex-Valued Multi-Layer Perceptrons -An Application to Polarimetric SAR Data*. Photogrammetric Engineering and Remote Sensing, vol. 76, pages 1081–1088, 2010. (Cited on page 20.)

[He *et al.* 2016] Kaiming He, X. Zhang, Shaoqing Ren and Jian Sun. *Deep Residual Learning for Image Recognition*. 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 770–778, 2016. (Cited on pages 21, 27 and 28.)

[Hjelm *et al.* 2019] R. Devon Hjelm, A. Fedorov, Samuel Lavoie-Marchildon, Karan Grewal, Adam Trischler and Yoshua Bengio. *Learning deep representations by mutual information estimation and maximization*. ArXiv, vol. abs/1808.06670, 2019. (Cited on page 20.)

[Hua *et al.* 2019] Wenqiang Hua, S. Wang, Wen Xie, Yanhe Guo and Xiaomin Jin. *Dual-Channel Convolutional Neural Network for Polarimetric SAR Images Classification*. IGARSS 2019 - 2019 IEEE International Geoscience and Remote Sensing Symposium, pages 3201–3204, 2019. (Cited on pages 2 and 20.)

[Isola *et al.* 2017a] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou and Alexei A. Efros. *Image-to-Image Translation with Conditional Adversarial Networks*. 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 5967–5976, 2017. (Cited on pages 17, 18, 27, 30 and 35.)

[Isola *et al.* 2017b] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou and Alexei A Efros. *Image-to-Image Translation with Conditional Adversarial Networks GitHub repository*, mar 2017. Accessed on 27/01/2021. (Cited on page 27.)

[Jenni & Favaro 2018] S. Jenni and P. Favaro. *Self-Supervised Feature Learning by Learning to Spot Artifacts*. 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 2733–2742, 2018. (Cited on page 20.)

[Jing & Tian 2020] Longlong Jing and Yingli Tian. *Self-supervised Visual Feature Learning with Deep Neural Networks: A Survey*. IEEE transactions on pattern analysis and machine intelligence, 2020. (Cited on pages 2 and 12.)

[Johnson *et al.* 2016] J. Johnson, Alexandre Alahi and Li Fei-Fei. *Perceptual Losses for Real-Time Style Transfer and Super-Resolution*. In ECCV, 2016. (Cited on pages 27 and 28.)

[Khaleefah *et al.* 2020] Shihab Hamad Khaleefah, S. A. Mostafa, A. Mustapha, N. Samsudin, M. Nasrudin and Abdullah Baz. *A survey on local binary pattern and gabor filter as texture descriptors of smart*

profiling systems. Indonesian Journal of Electrical Engineering and Computer Science, vol. 20, page 1379, 2020. (Cited on page 12.)

[Kingma & Ba 2015] Diederik P. Kingma and Jimmy Ba. *Adam: A Method for Stochastic Optimization*. CoRR, vol. abs/1412.6980, 2015. (Cited on page 34.)

[Kuhn 2013] Kjell Kuhn Max; Johnson. Applied predictive modeling. Springer New York, 2013. (Cited on page 62.)

[LeCun *et al.* 2010] Yann LeCun, Corinna Cortes and CJ Burges. *MNIST handwritten digit database*. ATT Labs [Online]. Available: http://yann.lecun.com/exdb/mnist, vol. 2, 2010. (Cited on page 10.)

[Lee 2013] Dong-Hyun Lee. *Pseudo-Label : The Simple and Efficient Semi-Supervised Learning Method for Deep Neural Networks*. ICML 2013 Workshop : Challenges in Representation Learning (WREPL), 07 2013. (Cited on page 11.)

[Ley *et al.* 2018] A. Ley, O. Dhondt, S. Valade, R. Haensch and O. Hellwich. *Exploiting GAN-Based SAR to Optical Image Transcoding for Improved Classification via Deep Learning*. In EUSAR 2018; 12th European Conference on Synthetic Aperture Radar, pages 1–6, 2018. (Cited on pages 17, 21, 23, 27 and 45.)

[Li *et al.* 2016] Y. Li, Manohar Paluri, James M. Rehg and Piotr Dollár. *Unsupervised Learning of Edges*. 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 1619–1627, 2016. (Cited on page 12.)

[Mahajan *et al.* 2018] D. Mahajan, Ross B. Girshick, Vignesh Ramanathan, Kaiming He, Manohar Paluri, Y. Li, Ashwin Bharambe and L. V. D. Maaten. *Exploring the Limits of Weakly Supervised Pretraining*. ArXiv, vol. abs/1805.00932, 2018. (Cited on page 11.)

[Merkle *et al.* 2017] N. Merkle, W. Luo, S. Auer, R. Müller and R. Urtasun. *Exploiting Deep Matching and SAR Data for the Geo-Localization Accuracy Improvement of Optical Satellite Images*. Remote. Sens., vol. 9, page 586, 2017. (Cited on pages 2, 7 and 17.)

[Mirza & Osindero 2014] Mehdi Mirza and Simon Osindero. *Conditional Generative Adversarial Nets*. CoRR, vol. abs/1411.1784, 2014. (Cited on page 17.)

[Miyato *et al.* 2019] Takeru Miyato, S. Maeda, Masanori Koyama and S. Ishii. *Virtual Adversarial Training: A Regularization Method for Supervised and Semi-Supervised Learning*. IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 41, pages 1979–1993, 2019. (Cited on page 11.)

[Moser *et al.* 2006] G. Moser, J. Zerubia and S. Serpico. *SAR amplitude probability density function estimation based on a generalized Gaussian model*. IEEE Transactions on Image Processing, vol. 15, pages 1429–1442, 2006. (Cited on page 19.)

[Moser *et al.* 2010] G. Moser, V. Krylov, S. Serpico and J. Zerubia. *High resolution SAR-image classification by Markov random fields and finite mixtures*. In Electronic Imaging, 2010. (Cited on page 19.)

[nasa.gov] nasa.gov. *Landsat-8*. https://landsat.gsfc.nasa.gov/landsat-8/landsat-8-overview. Accessed on 25/01/2021. (Cited on page 1.)

[Noroozi & Favaro 2016] M. Noroozi and P. Favaro. *Unsupervised Learning of Visual Representations by Solving Jigsaw Puzzles*. In ECCV, 2016. (Cited on page 12.)

[Paszke *et al.* 2019] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai and Soumith Chintala. *PyTorch: An Imperative Style, High-Performance Deep Learning Library*. In Advances in Neural Information Processing Systems 32, pages 8024–8035. Curran Associates, Inc., 2019. (Cited on page 32.)

[Pathak *et al.* 2015] Deepak Pathak, Philipp Krähenbühl and Trevor Darrell. *Constrained Convolutional Neural Networks for Weakly Supervised Segmentation*. 2015 IEEE International Conference on Computer Vision (ICCV), pages 1796–1804, 2015. (Cited on page 11.)

[Pathak *et al.* 2016] Deepak Pathak, Philipp Krähenbühl, J. Donahue, Trevor Darrell and Alexei A. Efros. *Context Encoders: Feature Learning by Inpainting*. 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 2536–2544, 2016. (Cited on page 12.)

[Pathak *et al.* 2017] Deepak Pathak, Ross B. Girshick, Piotr Dollár, Trevor Darrell and Bharath Hariharan. *Learning Features by Watching Objects Move*. 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 6024–6033, 2017. (Cited on page 12.)

[Raghu & Schmidt 2020] M. Raghu and E. Schmidt. *A Survey of Deep Learning for Scientific Discovery*. ArXiv, vol. abs/2003.11755, 2020. (Cited on pages 3, 11, 12 and 20.)

[Ren & Lee 2018] Zhongzheng Ren and Y. Lee. *Cross-Domain Self-Supervised Multi-task Feature Learning Using Synthetic Imagery*. 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 762–771, 2018. (Cited on page 12.)

[Reyes *et al.* 2019] M. F. Reyes, S. Auer, N. Merkle, Corentin Henry and M. Schmitt. *SAR-to-Optical Image Translation Based on Conditional Generative Adversarial Networks - Optimization, Opportunities and Limits*. Remote. Sens., vol. 11, page 2067, 2019. (Cited on pages 22, 51 and 58.)

[Saha *et al.* 2020] Sudipan Saha, F. Bovolo and L. Bruzzone. *Building Change Detection in VHR SAR Images via Unsupervised Deep Transcoding*. IEEE Transactions on Geoscience and Remote Sensing, pages 1–13, 2020. (Cited on pages 21 and 58.)

[Salimans *et al.* 2016] Tim Salimans, Ian J. Goodfellow, W. Zaremba, Vicki Cheung, A. Radford and Xi Chen. *Improved Techniques for Training GANs*. In NIPS, 2016. (Cited on page 15.)

[Singh *et al.* 2018] Suriya Singh, Anil Batra, Guan Pang, L. Torresani, Saikat Basu, Manohar Paluri and C. Jawahar. *Self-Supervised Feature Learning for Semantic Segmentation of Overhead Imagery*. In BMVC, 2018. (Cited on page 20.)

[Skolnik 2008] M.I. Skolnik. Radar handbook, third edition. Electronics electrical engineering. McGraw-Hill Education, 2008. (Cited on pages 6 and 7.)

[Ulyanov *et al.* 2017] Dmitry Ulyanov, Andrea Vedaldi and Victor Lempitsky. *Instance Normalization: The Missing Ingredient for Fast Stylization*, 2017. (Cited on page 32.)

[Wang *et al.* 2019] L. Wang, X. Xu, Y. Yu, R. Yang, Rong Gui, Zhaozhuo Xu and F. Pu. *SAR-to-Optical Image Translation Using Supervised Cycle-Consistent Adversarial Networks*. IEEE Access, vol. 7, pages 129136–129149, 2019. (Cited on pages 2 and 22.)

[Wood 2020] Therese Wood. *Visualizing All of Earth's Satellites: Who Owns Our Orbit?* https://www.visualcapitalist.com/visualizing-all-of-earths-satellites/, 2020. Accessed on 27/01/2021. (Cited on page 1.)

[Zhang *et al.* 2016] Richard Zhang, Phillip Isola and Alexei A. Efros. *Colorful Image Colorization*. In ECCV, 2016. (Cited on page 12.)

[Zhu *et al.* 2017a] Jun-Yan Zhu, T. Park, Phillip Isola and Alexei A. Efros. *Unpaired Image-to-Image Translation Using Cycle-Consistent Adversarial Networks*. 2017 IEEE International Conference on Computer Vision (ICCV), pages 2242–2251, 2017. (Cited on pages 3, 18, 27, 30, 49, 51 and 58.)

[Zhu *et al.* 2017b] X. Zhu, D. Tuia, Lichao Mou, Gui-Song Xia, L. Zhang, Feng Xu and F. Fraundorfer. *Deep Learning in Remote Sensing: A Comprehensive Review and List of Resources*. IEEE Geoscience and Remote Sensing Magazine, vol. 5, pages 8–36, 2017. (Cited on page 12.)