BRNO UNIVERSITY OF TECHNOLOGY
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

FACULTY OF INFORMATION TECHNOLOGY
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FUSION OF RADAR AND VISUAL DATA FOR REMOTE SENSING

FÚZE RADAROVÝCH A VIZUÁLNÍCH DAT PRO DÁLKOVÝ PRŮZKUM ZEMĚ

MASTER'S THESIS
DIPLOMOVÁ PRÁCE

AUTHOR **TOMÁŠ STRYCH**
AUTOR PRÁCE

SUPERVISOR **MARTIN KOLÁŘ, Ph.D.**
VEDOUCÍ PRÁCE

BRNO 2022

Brno University of Technology
Faculty of Information Technology

Department of Computer Graphics and Multimedia (DCGM) Academic year 2021/2022

Master's Thesis Specification

Student: **Strych Tomáš, Bc.**
Programme: Information Technology and Artificial Intelligence
Specialization: Machine Learning
Title: **Fusion of Radar and Visual Data for Remote Sensing**
Category: Image Processing
Assignment:

1. Prostudujte problematiku Style Transfer s použitím neuronových sítí.
2. Vyberte si současnou metodu využívající hluboké učení pro dálkový průzkum Země (s dostupnou open source implementací), analyzujte její podstatu a možnosti odhadnout vizuální modalitu z radarové.
3. Navrhněte úpravu metody, aby dokázala generovat spektrální data z radarových v různých rozlišeních.
4. Seznamte se s generátorem trénovacích dat a vyberte datové sady vhodné na experimenty (data může poskytnout spol. World from Space).
5. Pomocí vybrané metody vygenerujte vizuální DPZ data s použitím skutečných radarových dat. Kvantitativně vyhodnoťte kvalitu generovaných dat.
6. Porovnejte dosažené výsledky a diskutujte možnosti budoucího vývoje.
7. Vytvořte plakát prezentující vaši práci, její cíle a výsledky.

Recommended literature:

1. Wang, Puyang, and Vishal M. Patel. "Generating high quality visible images from SAR images using CNNs." 2018 IEEE Radar Conference (RadarConf18). IEEE, 2018.
2. Karras, Tero, Samuli Laine, and Timo Aila. "A style-based generator architecture for generative adversarial networks." Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2019.
3. Filgueiras, R.; Mantovani, E.C.; Althoff, D.; Fernandes Filho, E.I.; Cunha, F.F.d. Crop NDVI Monitoring Based on Sentinel 1. Remote Sens. 2019, 11, 1441. https://doi.org/10.3390/rs11121441
4. Mazza, Antonio, et al. "Estimating the ndvi from sar by convolutional neural networks." IGARSS 2018-2018 IEEE International Geoscience and Remote Sensing Symposium. IEEE, 2018.
5. Scarpa, Giuseppe, et al. "A CNN-based fusion method for feature extraction from sentinel data." Remote Sensing 10.2 (2018): 236.

Requirements for the semestral defence:

- Body zadání 1 až 3.

Detailed formal requirements can be found at https://www.fit.vut.cz/study/theses/

Supervisor: **Kolář Martin, Ph.D.**
Head of Department: Černocký Jan, doc. Dr. Ing.
Beginning of work: November 1, 2021
Submission deadline: May 18, 2022
Approval date: November 15, 2021

Abstract

The aim of this thesis is to generate optical imagery in case of its unavailability. Optical and radar data from the past are used to generate such images. The main application field of this thesis is agriculture, where countless vegetation indexes can be used. In this thesis, for simplicity, only NDVI is utilized. Four datasets were created, each for the first three seasons of the year and the fourth that connects them all. As a solution for an image to image translation, Pix2Pix-cGAN was chosen. The results show the differences in the use of these datasets, between the different amounts and types of used pictures, as well as the interval adjustments between pictures. Our research found that the network is capable of creating plausible imagery with valid numerical values, but struggles to correctly utilize the information about the radar difference, which is important in order to evaluate plant development mainly when the optical imagery is unavailable. This thesis and its results are unique due to the geographically diverse dataset across Europe and the focus on agriculture, regardless of crop type.

Abstrakt

Cieľom práce je vygenerovať satelitný optický snímok v prípade jeho nedostupnosti. Takýto snímok je vygenerovaný z aktuálneho radarového snímku a za pomoci radarových a optických snímkov z minulosti. Zameranie práce cieli na poľnohospodárstvo, kde sa na analýzu dát používajú rôzne vegetačné indexy. Pre zjednodušenie problematiky je práca zameraná len na optický snímok zobrazujúci NDVI. Boli vytvorené 4 dátové sady, pre prvé tri ročné obdobia a štvrtý, ktorý ich spája. Pre riešenie problému preloženia obrázku z jedného na druhý bol použitý model Pix2Pix-cGAN. Výsledky práce zobrazujú rozdiely pri použití dátových sád, rozličného množstva a typu použitých snímkov, tak ako aj intervalu medzi snímkami. Daným výskumom bolo zistené, že sieť je schopná vytvárať reálne uveriteľné obrázky s validnými numerickými hodnotami, avšak má problém správne využiť informáciu o radarovej zmene, ktorá je dôležitá pre ohodnotenie vývoja rastlín práve v prípade nedostupnosti optického snímku. Táto práca a jej výsledky sú jedinečné vďaka naprieč Európou geograficky rozmanitej dátovej sade a vďaka zameraniu na agrikultúru, a to bez ohľadu na typ plodín.

Keywords

GAN, Image to image transfer, Pix2Pix, Remote sensing, Earth observation, agriculture, dataset creation, radar imagery, optical imagery, NDVI, Sentinel 1, Sentinel 2, satellite imagery

Klíčová slova

GAN, predklad obrázka na obrázok, Pix2Pix, ďiaľkový priezkum zeme, agrikultúra, vytvorenie dátovej sady, radarový snímok, optický snímok, NDVI, Sentinel 1, Sentinel 2, satellitný snímok

Reference

STRYCH, Tomáš. *Fusion of Radar and Visual Data for Remote Sensing.* Brno, 2022. Master's thesis. Brno University of Technology, Faculty of Information Technology. Supervisor Martin Kolář, Ph.D.

Rozšířený abstrakt

Táto práca sa zaoberá vytvorením optického snímku v prípade, že je nedostupný. Dôvodom nedostupnosti optického snímku sú najmä oblaky. Zvyčajne sa na vyriešenie tohto problému používajú dva prístupy.

Prvý prístup je vhodný v prípade, ak je zamračená len časť obrázku. Vtedy sa snažíme vyhladiť optický snímok na miestach kde sú oblaky. Aby sa oblasť odhadla správne, používa sa optický obrázok z minulosti. V prípade, že je obrázok celý zakrytý oblakmi musí byť odvodený z predchádzajúceho optického snímku.

Druhý prístup, ktorý používa radarové snímky, je vhodný práve v takomto prípade, pretože radarový snímok je dostupný za akéhokoľvek počasia. Druhý prístup sa snaží transformovať radarový obrázok na optický.

Zámerom práce je spojiť obidva prístupy. Na vytvorenie aktuálneho optického snímku používame optický snímok z minulosti súčasne s aktuálnym radarovým snímkom.

K ďalšiemu vylepšeniu siete by malo dopomôcť pridanie radarového snímku z minulosti. Sieť má potom možnosť využiť ďalšiu informáciu, ktorá sa skrýva v porovnaní rozdielu medzi radarovým snímkom z minulosti a aktuálnym radarovým snimkom. Zmena radaru medzi meraniami by mala zlepšiť odhad zmeny pre optický snímok.

Radarové aj optické snímky sú poskytované rôznymi satelitmi. Zvyčajne sa preto líši deň pozorovania. V tejto práci je tento faktor odstránený tým, že merania medzi radarovým a optickým snímkom sú od seba vzdialené najviac o jeden deň. Ďalším dôležitým faktorom je, že sa nesnažíme vytvárať optický snímok, ale priamo snímok zobrazujúci jeden z vegetačných indexov nazývaný normalizovaný vegetačný index – NDVI.

V prvej časti práce som sa venoval štúdiu vzdialeného pozorovania a hľadal som možné riešenia v oblasti hlbokého učenia. V rámci radarového obrázku som sa venoval hlavne základným pojmom a faktorom, ktoré ovplyvňujú radarový snímok. Pri optickom snímku som sa sústredil na multispektrálny obrázok a NDVI. Nakoniec som sa zaoberal satelitmi, ktoré tieto dáta poskytujú. Vhodným riešením pre generovanie obrázkov je generatívna sieť – GAN, respektíve jej rôzne varianty. Pre riešenie som vybral sieť Pix2Pix – cGAN, ktorej účel je preklad obrázku na iný obrázok.

Návrh práce prezentuje postupné pridávanie vstupov do siete Pix2Pix. Model 0 prekladá aktuálny radarový obrázok na aktuálny optický obrázok. Model 1 používa na vstupe aktuálny radarový obrázok a starý optický snímok. Oproti modelu 1 model 2 navyše používa ešte starý radarový snímok. Model 3 má rovnakú architektúru ako model 2, avšak bude používať dvojnásobný rozostup medzi meraniami. Nakoniec som vytvoril model, ktorý som nazval opravený model 3 s malými zmenami v architektúre. Cieľom je vygenerovať NDVI obrázok, ktorý sa čo najviac podobá k aktuálnemu NDVI obrázku.

Pred trénovaním som vytvoril 4 dátové sady. Prvé tri dátové sady zodpovedajú prvým trom ročným obdobiam a štvrtá ich spája. V spojenej dátovej sade je 33501 párov obrázkov. Tieto obrázky majú veľkosť 256x256 pixelov a obsahujú maximálne 5% oblačnosti, alebo inak nevhodných pixelov. Jeden pixel zodpovedá oblasti 10x10 metrov.

Modely som numericky vyhodnocoval na základe strednej absolútnej chyby a absolútnej relatívnej chyby. K trénovaniu som použil spojenú dátovú sadu a dátovú sadu zameranú na jar. Model 1 poskytol najlepšie numerické výsledky. Model 3 sa najviac približuje k realite, kde často krát po dobu niekoľkých meraní nemáme dostupný optický snímok z minulosti. Model 3 je najvhodnejší pre ďalšiu prácu aj z ďalšieho dôvodu. Dlhší interval medzi meraniami odzrkadľuje väčšie rozdiely vo vegetácii a táto informácia je želaná vedomosť, ktorú od siete očakávame. Opravený model 3 nepriniesol zlepšenie výsledkov oproti základnému modelu 3. Model 2 mal na rozdiel od modelu 1 dostupnosť rozdielu medzi predchádzajúcim a aktuálnym radarovým

snímkom. Avšak model 2 nepreukázal numerické zlepšenie. Všetky modely okrem modelu 0, vytvárajú obrázky podobné realite. Hlavným problémom siete je slabé zachytenie väčšej zmeny vegetácie medzi predchádzajúcim a aktuálnym snímkom.

Počas experimentov sa mi podarilo zistiť, že odchýlka generatívnej siete je dostatočne malá a nepoškodzuje tak správnosť vygenerovaných obrázkov. Upravená stratová funkcia, ktorej cieľom bolo podporiť učenie dát s väčším rozdielom ako je odchýlka dát, priniesla len malé zmeny v distribúcii generovaných dát, ale nepriniesla žiadne vizuálne zmeny.

V závere som zhodnotil povzbudivé výsledky natrénovaných modelov a experimentov a popísal niekoľko návrhov na zlepšenie výsledkov aktuálnej práce v budúcnosti.

Fusion of Radar and Visual Data for Remote Sensing

Declaration

I hereby declare that this Master's thesis was prepared as an original work by the author under the supervision of Martin Kolář, Ph.D. The supplementary information was provided by Roman Bohovič, Ph. and by Ing. Mikuláš Muroň. I have listed all the literary sources, publications and other sources, which were used during the preparation of this thesis.

.....
Tomáš Strych
May 16, 2022

Acknowledgements

I would like to thank my supervisor Martin Kolář, Ph.D. for his advice and insights. Also, to an external company World From Space for providing data and additional consultations and discussions about the topic. Computational resources were supplied by the project "e-Infrastruktura CZ" (e-INFRA CZ LM2018140) supported by the Ministry of Education, Youth and Sports of the Czech Republic. I want to thank Anna Mária Pisoňová and Zuzana Strychová for proofreading the most critical parts of the thesis. In the end, I would like to thank all of my friends and family members for their support.

Contents

Chapter 1

Introduction

Multispectral optical images taken by satellites are used for many purposes. One such is agriculture. Thanks to vegetation indices, which are calculated from multispectral optical images, it is possible to measure the healthiness of plants. Farmers, foresters or ecology activists can benefit from such information. Although, optical images are frequently unavailable, mainly because of clouds.

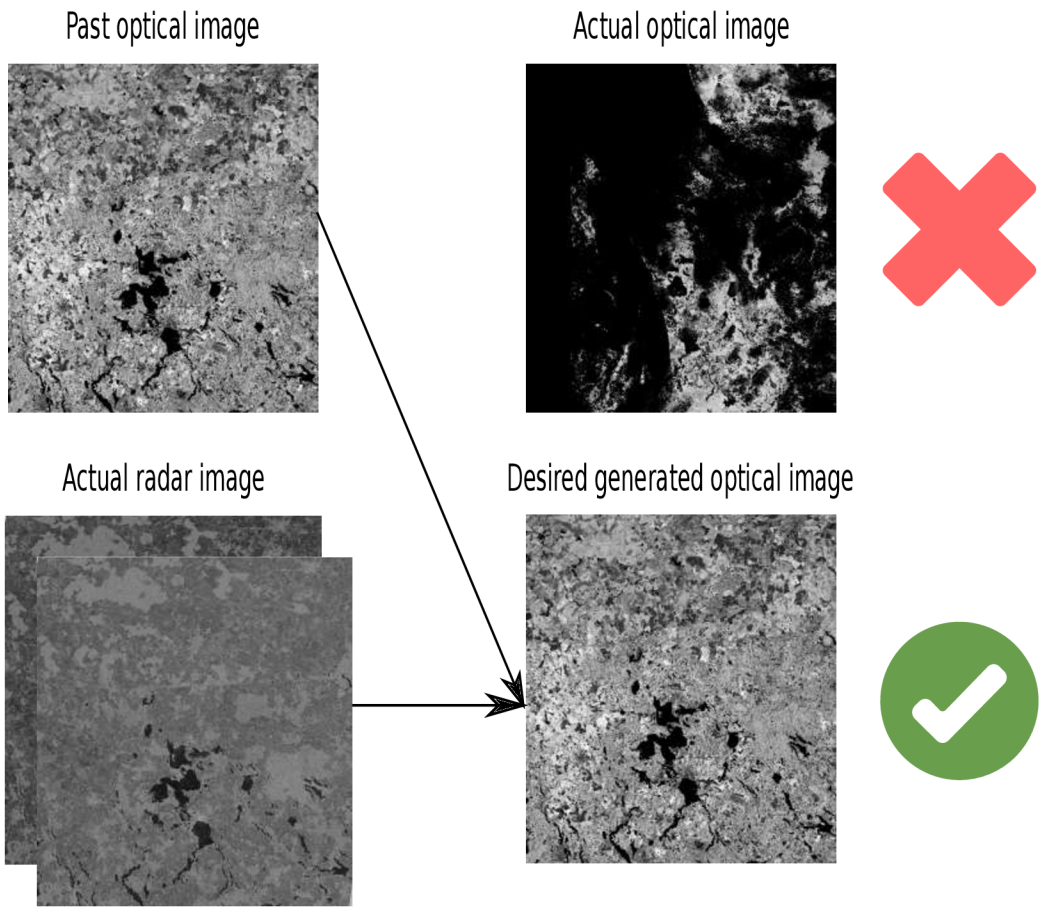

Figure 1.1: Motivation. Actual cloud full image versus the desired image composed from the past optical and the actual radar image.

There are two common ways to obtain the desired image shown in figure 1.1, and neural networks are often used for both approaches. The first approach is identifying the clouds and creating an approximate solution without them, often used when the clouds are only in part of the picture. If clouds cover the whole image, the answer must be derived from the last cloudless image. The second approach translates the radar image into an optical image. The radar is always available. This thesis fuses both approaches. It uses the actual radar image with the past optical image to generate. Additional beneficial information that should result in a more reasonable estimation is the difference between the past radar and the actual radar image.

Radar and optical images are provided by different satellites. Therefore the observation day for the same area usually varies. This factor is no longer an issue in this thesis because the maximum observation day difference between the radar and optical image is set to one day. Another factor that reduces the complexity of the problem is the usage of just one vegetation index called NDVI, so we aim to train a network to master the direct change between the NDVI and the radar picture.

This thesis uses a conditional generative adversarial network to experiment with the amount of actual and past data provided to the network. The goal of the thesis and the goal of the conditional generative adversarial network is to generate an NDVI image as similar as possible to the real NDVI image.

To start with, in chapter 2, a reader can find a basic understanding of remote sensing, radar and optical imagery, vegetation NDVI, and satellite programs that provide data. Next, chapter 3 leads us through deep learning, where the main topic is GAN evolution, finishing with the possible approaches for the proposal. The data preprocessing and dataset creation are defined in chapter 4. Next, the model proposal with implementation and training details is explained in chapter 5. The results of trained models with additional details are described in detail in chapter 6. In addition, chapter 7 experiments with the results to get even more insights. Lastly, chapter 8 concludes the results and experiments, proposes future research, and summarizes the outcomes of the thesis.

Chapter 2

Remote sensing

In this thesis, agriculture is the object of remote sensing. The purpose of remote-sensing in agriculture is to determine the plant's condition based on the colours of leaves or the overall appearance of plants. [37]

To understand remote sensing, here is the definition. *„Remote sensing has been variously defined, but basically, it is the art or science of telling something about an object without touching it. (Fischer et al., 1976, p. 34.)“* [19]

Remote sensing is done by Earth observation – EO. Earth observation can be done remotely or in situ. Our focus is mainly on remote EO satellites. It can also be called satellite-based remote sensing.

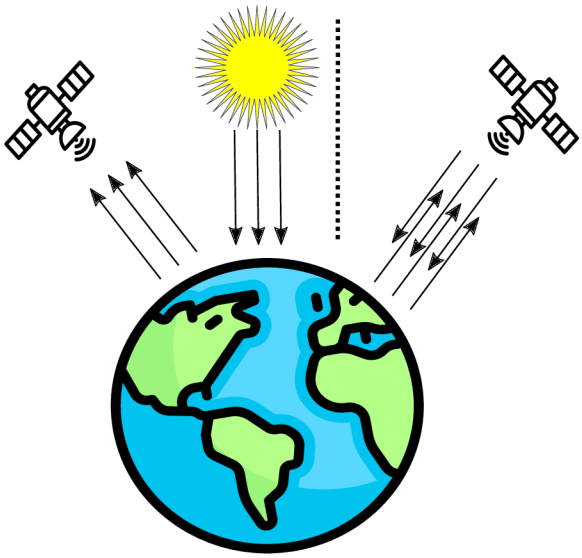

Figure 2.1: The difference between passive (left) and active sensor (right side of the figure), where passive sensor relies on external energy source (e.g. the Sun), whereas active ones have their energy source.

In this thesis, we will use both of the principles from figure 2.1. Sentinel 1 is an active sensor with Synthetic aperture radar, and Sentinel 2 is a passive sensor optically monitoring the Earth. But firstly, we aim to understand the radar in section 2.1 and Synthetic aperture radar in section 2.2. The latter, the optical part, guides us through the introduction to the observable optical

world, via a specific vegetation index, in section 2.4. In the end, in section 2.5, the data provider of both radar and optical imagery is introduced.

2.1 RADAR - Radio Detection and Ranging

This paragraph is based on [42] if not mentioned otherwise.

Radar transmits electromagnetic energy and detects an echo of a targeted object. Information about the target comes from the echo. Received echo is also called backscatter. More about backscatter is in the next paragraph.

Radar can detect the range, distance and also angular location of the target. Additionally, it is possible to predict the future position of the moving target. Thanks to the Doppler effect, a moving target differs from stationary targets. Thanks to the transmitter, it is an active device. Radar can detect targets in all weather conditions. Radar has been applied all across the frequency range, starting with a few Megahertz up to hundreds of Gigahertz[24]. The original purpose for radar was military surveillance, but today it is enormously applied in the civil sector. One such field is, of course, space science.

While reading about SAR or radars in general, I found out that the reader needs to understand what is represented by C-band or get the translation between wavelength and frequency. Figure 2.2 represents a helpful dictionary for this purpose.

Next paragraphs are picked from [6][45], if not mentioned otherwise.

The scattering of vegetation consists of multiple parts. Firstly the dispersion is from the top of the tree. Secondly, it is the volume scattering from the tree crown and branches, and lastly, the surface and volume scattering from the ground. For simplicity, I explained it on the tree, but the scattering amount differs for each structure.

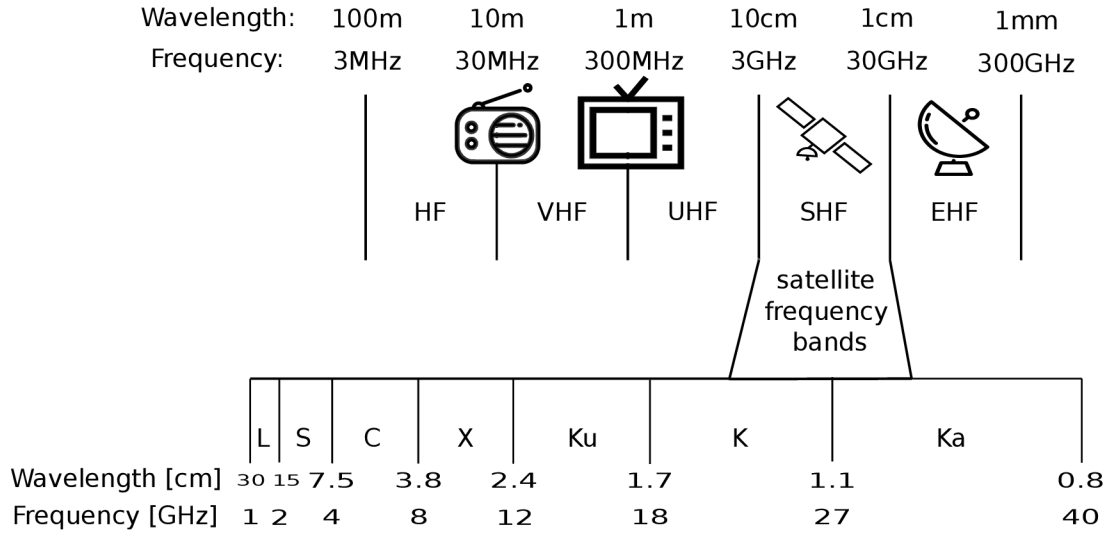

Figure 2.2: Part of radar spectrum, which is part of electromagnetic spectrum. In upper side of figure there is a naming by ITU convention. In lower side are bands used by satellite. Inspired by [7] and [12].

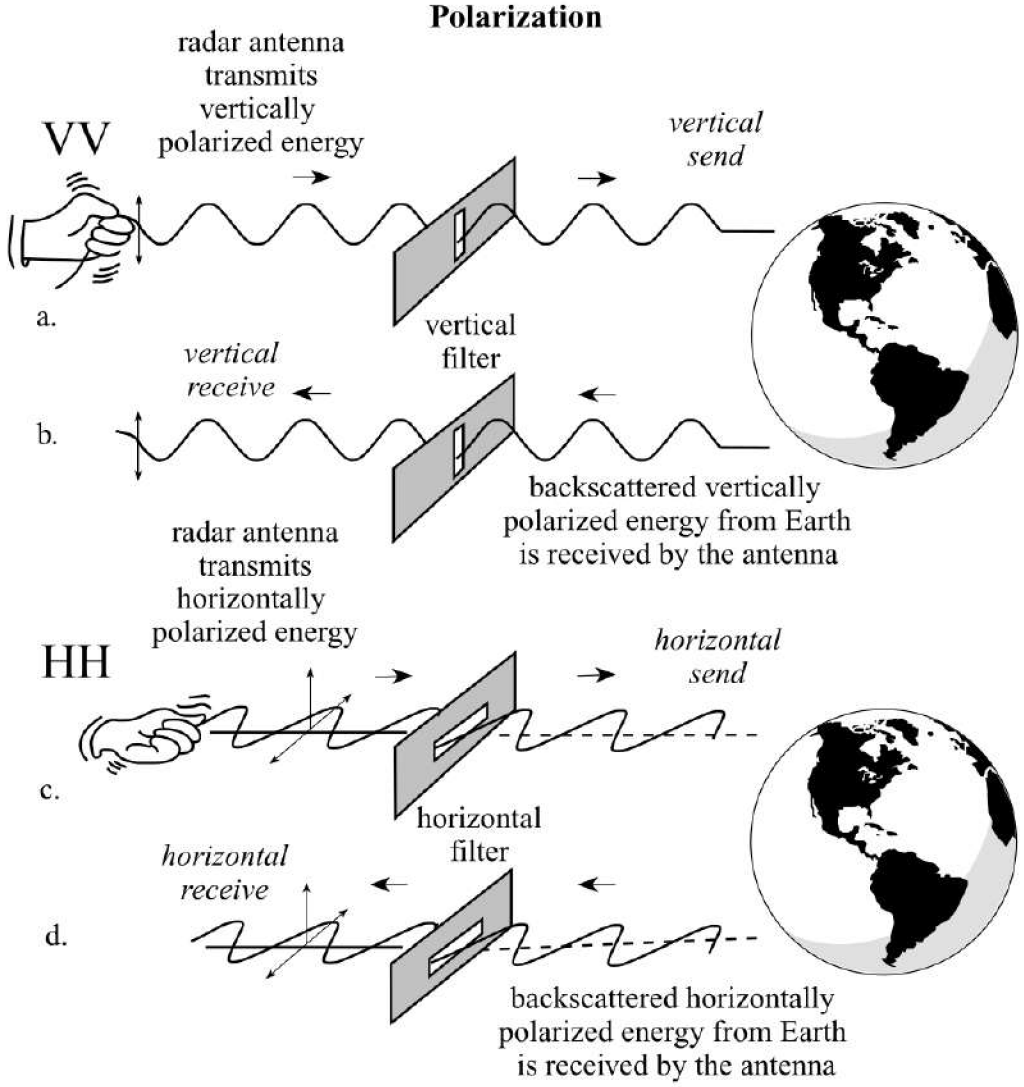

Figure 2.3: Transmitting and receiving polarization process. In a) and c) antenna transmits the signal. In b) only some of the vertically polarized energy is not depolarized by the terrain and stays vertical. Antenna records only vertically polarized energy. Similarly, in d) for horizontally polarized energy. Source of the figure[32].

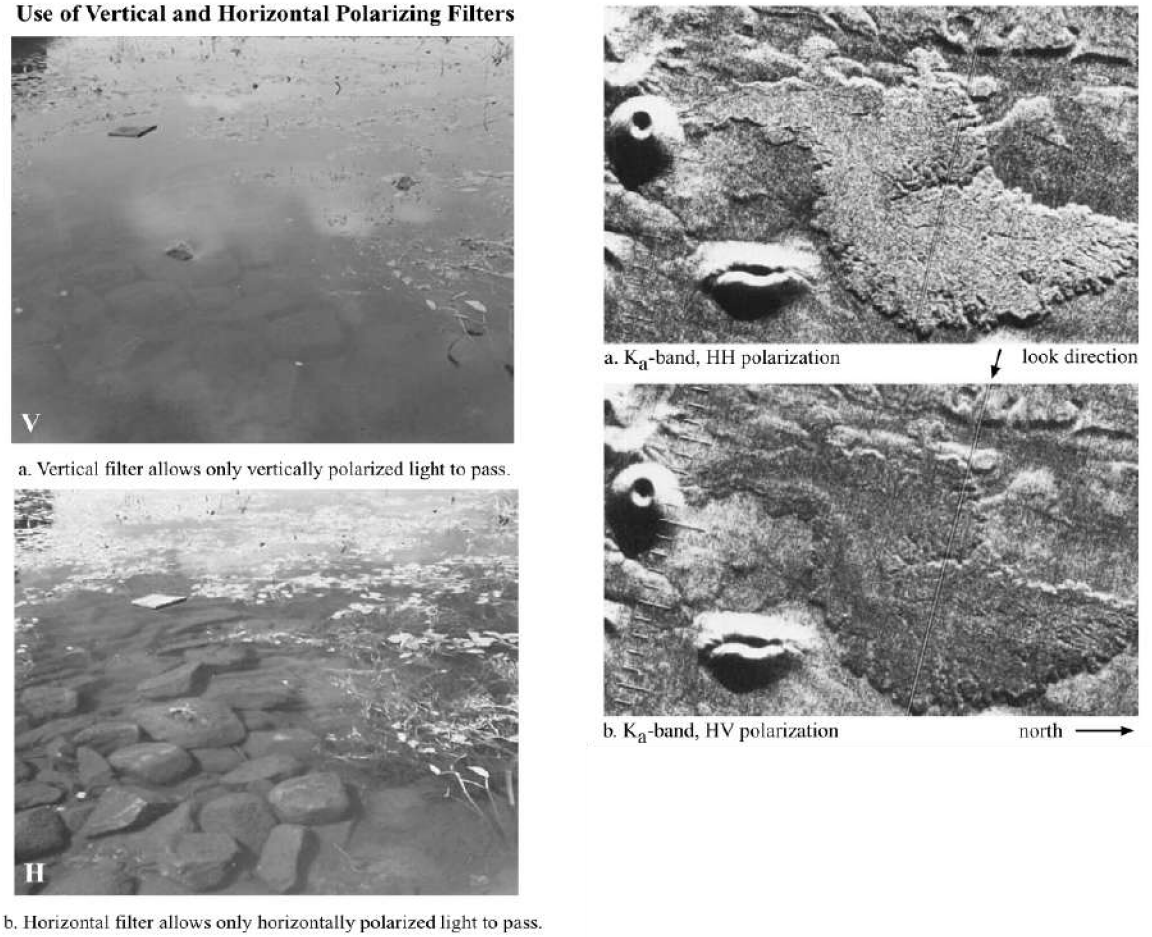

Figure 2.4: A practical example of the difference between vertical and horizontal polarization. On the left side of the figure, a) vertically polarized light provides much less information about objects under the ice. Contrary, b) the vertical filter shows reasonable information. On the right side of the figure, the radar image shows a volcano cinder cone and basalt lava flow. Both images have the same direction of look. Strong response in a), (HH) in comparison to b), (HV). We can observe the response thanks to a wavelength that is large enough to reflect lava blocks directly. Source of the figure[32].

There are plenty of parameters and factors affecting the backscatter. One category is parameters affected by **sensor** and the other is affected on the ground by **target**.

1. **Sensor** – radar parameters

   (a) **Wavelength/Frequency** – (e.g. X, C), shown in figure 2.2 as satellite frequency.

   The primary scattering objects in a tree are relatively small elements (leaves, branches, and stems). These are important from an agricultural perspective. The primary objects produce backscatter when the element's size is equal to or bigger than the wavelength. Otherwise, when elements are smaller, little to no backscatter is produced, but it can attenuate the signal. For example, when the P band is used with a wavelength of around 70 cm, the tree backscatter is produced by larger branches, twigs, limbs, and a trunk, but not leaves or fruits. Oppositely, with a small wavelength of X band around 3 cm, scatterers are all parts of a tree, mainly the crown, where the leaves are. With a shorter wavelength, the vertical structure is less sensitive. This is caused by so many scatterers on top of vegetation.

   (b) **Polarization** – represents the orientation of the electric field. When radar can transmit either horizontally (H) or vertically (V). Likewise, the antenna can receive vertically or horizontally polarized backscatter energy. Most of the time, it can receive both. There are four combinations of polarization: HH, VV, HV, and VH, where the former letter describes the transmitted and the latter received signal. The whole process is described nicely in figure 2.3.

   HH and VV transmit and receive the signal of the same polarization. Such polarization is called like-polarized. Oppositely HV and VH are called cross-polarized. The polarization is orthogonal one to another. For better understanding, in figure 2.4, we can see real-life examples of polarization and the important difference between it.

   (c) **Angles and ranges** – satellite geometry terms. Shown in figure 2.5. Taken from [2][33].

   - α – incidence angle – angle between the radar beam and the imaginary perpendicular to the ground surface.
   - γ – look angle – radar angle to the surface.
   - Slant range distance – the distance between the radar and target on the surface.
   - Ground range distance – true horizontal distance along the ground to the target.
   - Azimuth and range is the notation used per axis. Along-track is sometimes used instead of azimuth.
   - Size of the ellipsis around the target, along the azimuth dimension, is called the antenna footprint.
   - Distance between near range and far range is called swath.

   The next thing to discuss is relief displacement. Consequences of relief displacement are foreshortening and layover. Both of them occur with a small incidence angle and mainly in mountainous terrain. In agriculture, most of the ground is flat terrain. Therefore, there is no need to explain it further. On the other hand, the reader should be aware of displacements.

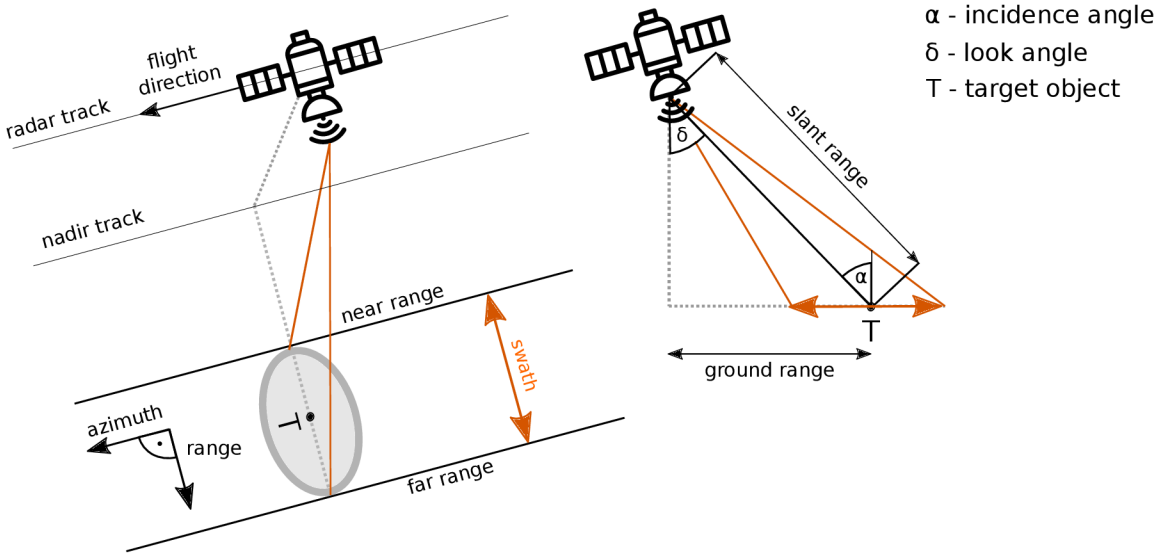

Figure 2.5: Explains satellite geometry terms. Three dimensional figure on the left, with additional terms on the right 2D figure.

(d) **Resolution** – Gradually explained spatial, spectral and temporal resolution.

Spatial resolution – the size of the smallest recognizable object in the image. The image consists of pixels. If each pixel represents one square meter, it is called a one-meter spatial resolution. The higher resolution of the image, the more details can be seen on each pixel, and the picture becomes sharper. Spatial resolution is shown in figure 2.6.

Figure 2.6: Difference between resolutions. With a smaller grid, more information about figures is known. Taken from [38].

Spectral resolution – refers to the number of bands and the wavelength width of each band. A band is a narrow portion of the electromagnetic spectrum. Shorter wavelength widths can be distinguished in higher spectral resolution images. Multispectral imagery captures several bands with different wavelengths, such as red, green, or blue, and frequencies beyond the visible light range, like near-infrared (NIR).

Temporal resolution – how often is it doable to remote sense the object. In other words, how much time it will take until the satellite can picture the same area. [37]

2. **Target** – vegetation on the ground

   (a) **Structure** – Vegetation around the world has different forms of growth. One of the divisions is by stems to herbaceous and woody stems. Herbaceous stems are gramineous (grass) or stalk dominated (corn). For woody stems, we have shrubs and trees. Trees can be excurrent (pine), decurrent (maple, oak), or columnar (palms). Vegetation with woody stems can live from multiple to hundreds of years. Of course, during their life cycle, their structure changes. And not only during but also afterlife. This is just a taste of the structure topic. An important message is that structurally, everything is changing the backscatter, even the slight change.

   (b) **Surface roughness** - average height variation between the surface cover and the plane ground. It depends on the incidence angle and wavelength. [2]
   Figure 2.7, shows how with increasing wavelength, roughness necessitates greater height variation. It also shows how with increasing incidence angle, roughness necessitates greater height variation.

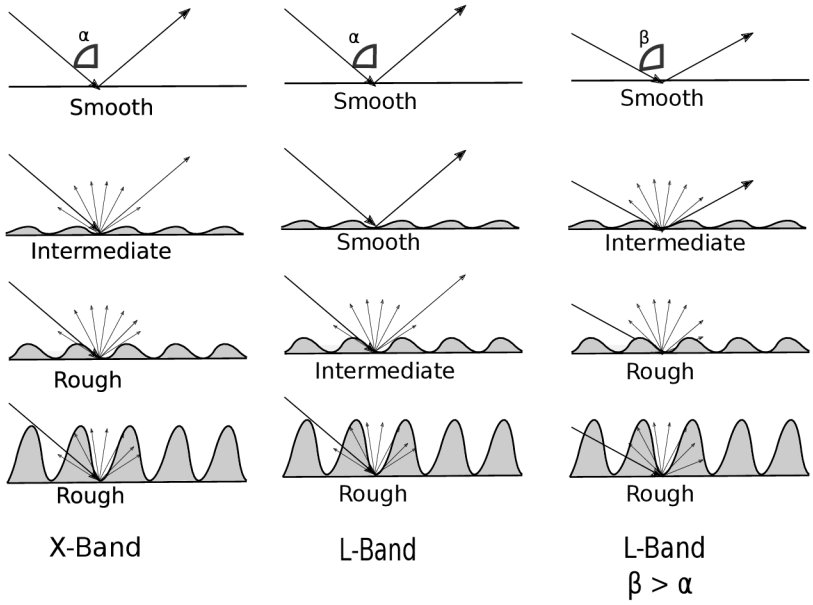

Figure 2.7: Observation of different surface roughness at band change and incidence angle. With rough surface, there is not enough backscatter (missing outgoing arrow). Inspired by [45].

   (c) **Dielectric constant** – Humidity and amount of water in the soil is influencing the absorption and propagation of backscatter. Higher the humidity content the lower the penetration through the vegetation.

To sum up the different radar factors, I created figure 2.8, which shows the different backscatter per factor.

Before we move on to the next section, there is a fundamental downside of a radar approach called speckle. A speckle is a form of noise that degrades the image quality. It is caused by the sum of multiple scatterers phases and amplitudes inside one resolution cell. Comparing similar

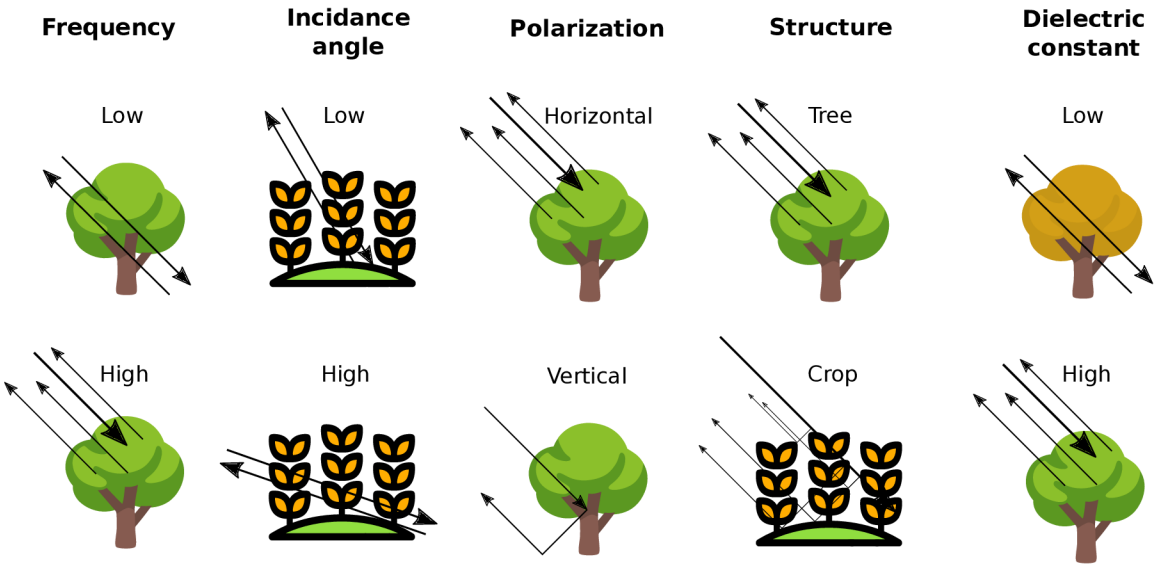

Figure 2.8: Radar backscatter influenced by different factors. Inspired by [25].

cells, when each cell has strong fluctuation, results in non deterministic final image. Speckle noise appears in areas where surface roughness is comparable to radar wavelength. Techniques that reduce the speckle always worsen the spatial resolution. An example of such a technique is a multi-look. Speckle and multi-look are shown in figure 2.9. [36]

2.2 Synthetic aperture radar – SAR

SAR is mounted to a satellite moving around the Earth. An enormous antenna has a better resolution than the small one. But getting such a big antenna is, let's say, problematic. Instead of it, the movement created by Earth's gravity is used. The moving satellite has the targeted object in the swath during the time interval. During the interval, the same object has echoed multiple times. An appropriate combination of the received signals allows the virtual construction of an enormous synthesized antenna, called a synthetic aperture. Synthesized antenna is shown in figure 2.10. Such a workaround improves the resolution without unnecessary enlargement of the antenna. [36] [12]

2.3 Optical remote sensing

This section is based on [44] [5]. Utilizes the visible spectrum, the near-infrared and short-wave infrared sensors to get the surface images. Such a spectrum is shown in figure 2.11. The radiation source is mainly from the sun, and detected radiation scatterers are from the surface or the atmosphere.

Optical remote sensing with one spectral band is called panchromatic. Optical remote sensing with multiple bands is called multispectral, and the one with hundreds of bands is called hyperspectral image. As of today, all satellites are at least multispectral. To easily understand, we can imagine each spectrum as an extra layer. Each additional layer multiplies the size of the information. Multispectral satellites provide an excellent start for data processing and analysis.

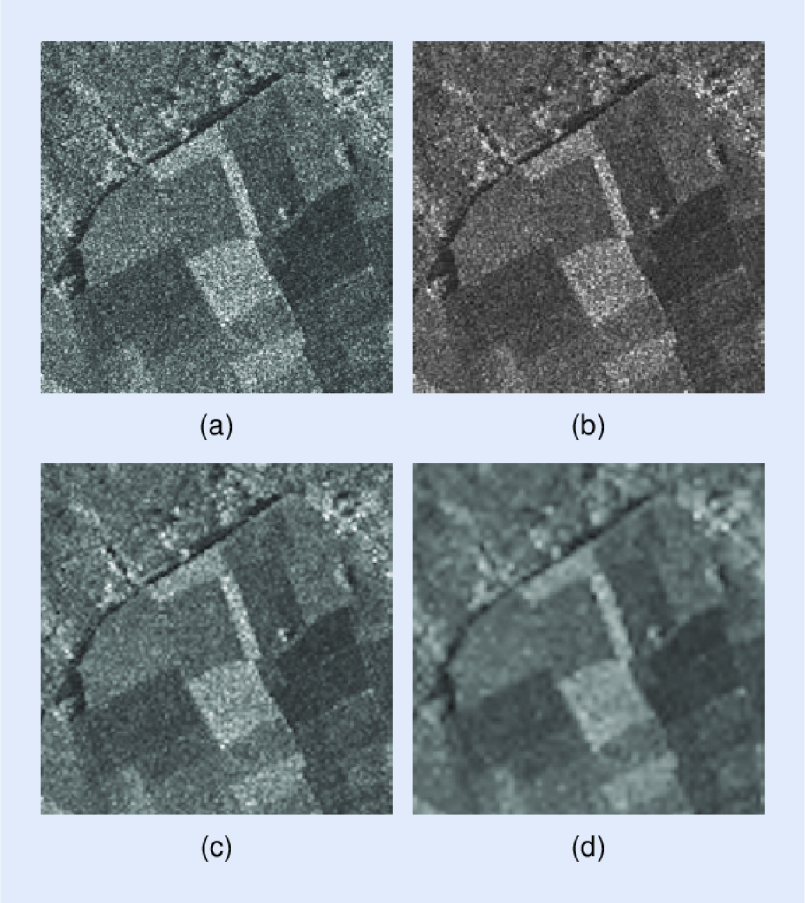

Figure 2.9: Multi-look applied on radar image to reduce the speckle noise. (a) Without multi-look, (b) 2x2 multi-look, (c) 4x4 multi-look, (d) 8x8 multi-look. Taken from [36].

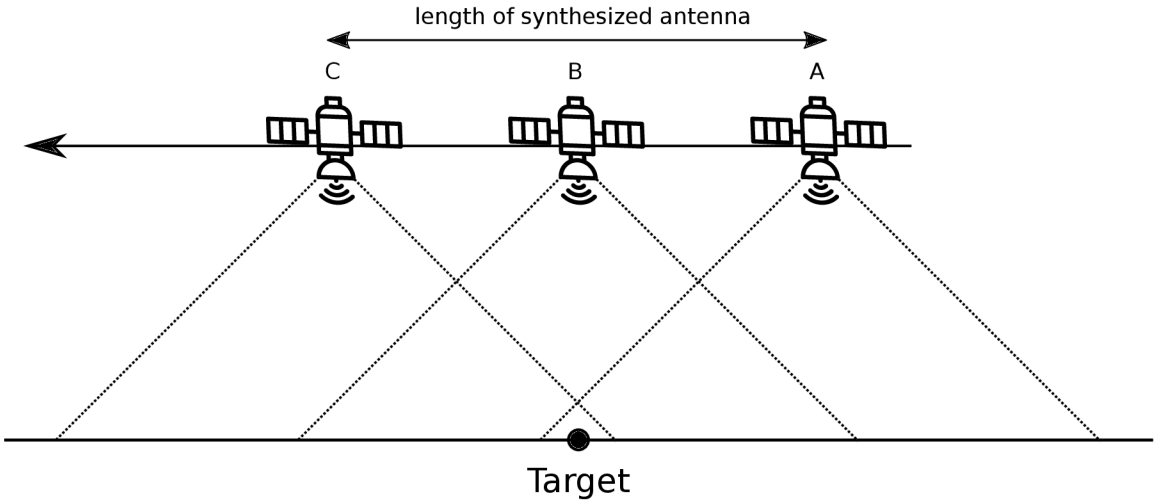

Figure 2.10: SAR stands for synthetic-aperture radar. The figure explains what is meant by synthetic. The satellite is moving from A to C. During this movement period, it can always observe the target. Data are processed as if the distance from A to C was the aperture of a huge radar antenna.

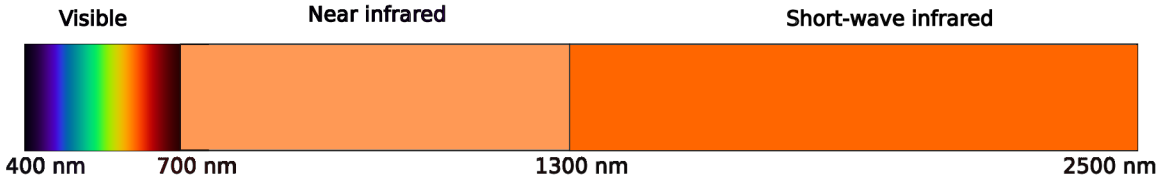

Figure 2.11: Part of the electromagnetic spectrum used in optical remote sensing.

Vegetation is easily recognizable from other land cover types. It peaks in green colour across the visible spectrum because of the chlorophyll used in photosynthesis. From the non-visible spectrum, near-infrared reflectance has even higher reflectance than green, thanks to the leaves cellular structure.

Now we know how to recognize vegetation and what spectrum is available. Vegetation indices benefit from multispectral images. Let's look at a vegetation index chosen for this thesis.

2.4 Normalized difference vegetation index – NDVI

As we learned in section 2.3, multispectral images have multiple layers. Multiple layer pictures can create a composite. Certain patterns or features can help us during the analysis within each band. Combining extracted patterns from multiple layers into new pictures can enhance representations of desired ground objects. There is more than a hundred vegetation indices derived from multispectral imagery. So, if the desired ground object is vegetation, there are many indices to choose from. NDVI, one of it, has great use in agriculture due to the rapid delineation of vegetation and vegetative stress. [30]

Normalized Difference Vegetation Index (NDVI) is expressed as:

$$NDVI = \frac{(NIR - RED)}{(NIR + RED)} \tag{2.1}$$

where, RED is visible red reflectance and NIR is near infrared reflectance. Example of such calculation is shown in figure 2.12.

NDVI is a normalized index with values between -1 and 1, with sensitive responses to even areas with less vegetation. The index is widely popular in regional or global vegetation assessments. The downside of the NDVI is soil sensitivity, not to mention the atmosphere, clouds, cloud shadows and leaf shadows. [47]

2.5 Sentinels

Sentinel 1 and Sentinel 2 missions are parts of the Copernicus program, previously called the GMES program, which was initiated and managed by and European Commission with the help of the European Space Agency. The Copernicus Open Access Hub[1] (previously known as Sentinels Scientific Data Hub) provides open and complete access to Sentinel 1 and Sentinel 2. Note

[1] https://scihub.copernicus.eu/

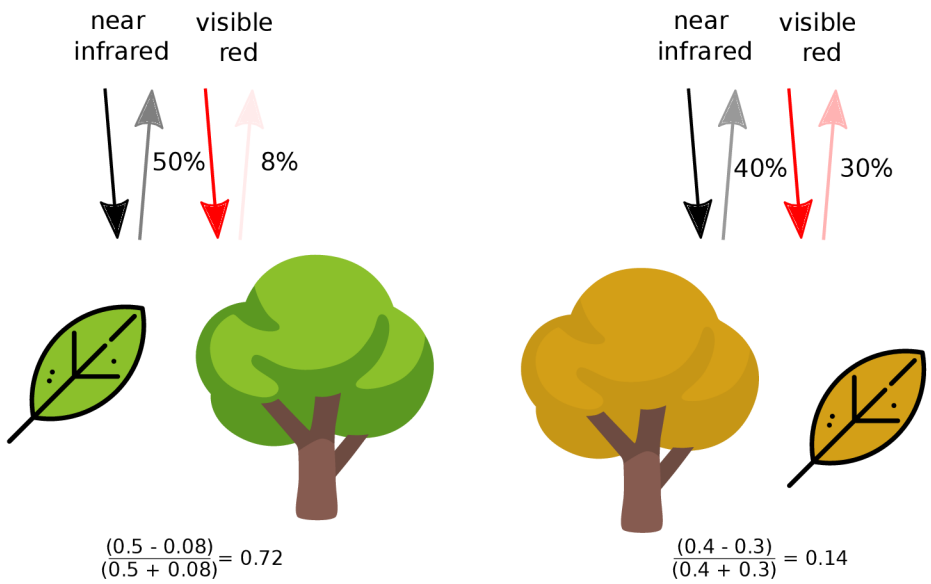

Figure 2.12: NDVI calculation for both trees. On the left, the healthy tree absorbs most of the visible red light. On the right, the unhealthy tree reflects less infrared and more of the visible red light. Inspired by [4].

for the reader that there are more sentinel missions, but only the data from Sentinel 1 and Sentinel 2 are used in this thesis. [1]

Sentinel 1: Right now, there are two active Sentinel 1 satellites, Sentinel 1A and Sentinel 1B. They share the same sun-synchronous orbit plate with a 180-degree orbital phasing difference. A single Sentinel 1 satellite has a 12-day repeat cycle, so two have a repeat cycle equal to six days, half of it. Sentinel 1 satellites are at low Earth orbit with an altitude of 693 km. Each satellite carries a C band SAR. Radar centre frequency of 5.405 GHz, corresponding to the wavelength of around 5.55 cm. SAR supports the following polarizations: HH+HV, VV+VH, VV, HH. The incidence angle range is 20°– 46°.[15][8][9]

Sentinel 2: This paragraph and the table data is based on [10][11].
Sentinel 2 is a wide swath, high resolution, multispectral imaging mission. Similarly to Sentinel 1, Sentinel 2, right now, has two active satellites 2A and 2B. Satellites also share a sun-synchronous orbit plate with a 180-degree orbital phasing difference. The repeat cycle is five days, less than Sentinel 1, and the swath is 290 km. A multispectral instrument carried by the satellite measures the reflected radiance in 13 spectral bands. Table 2.1 shows in detail the band specification together with spatial resolution.

The revisit frequency halves for both Sentinel 1 and Sentinel 2 as satellite A ascends and satellite B descends the Earth orbit. The revisit frequency is 3 days at the equator. But because the orbit track spacing varies with latitude, the revisit frequency is significantly greater at higher latitudes than at the equator. Europe is in mid-latitude and has a revisit frequency of around two days. [10][9].

With Sentinel 1 providing the radar data and Sentinel 2 providing the optical data, I had resources for dataset creation. From optical imagery provided by Sentinel 2, it is possible to calcu-

Sentinel 2 bands	S2A wavelength (nm)	S2A bandwidth (nm)	S2B wavelength (nm)	S2B bandwidth (nm)	spatial resolution (m)
Band 1 – Coastal aerosol	442.7	21	442.2	21	60
Band 2 – Blue	492.4	66	492.1	66	10
Band 3 – Green	559.8	36	559	36	10
Band 4 – Red	664.6	31	664.9	31	10
Band 5 – VNIR	704.1	15	703.8	16	20
Band 6 – VNIR	740.5	15	739.1	15	20
Band 7 – VNIR	782.8	20	779.7	20	20
Band 8 – NIR	832.8	106	832.9	106	10
Band 8A – Narrow NIR	864.7	21	864	22	20
Band 9 – Water vapour	945.1	20	943.2	21	60
Band 10 – SWIR – Cirrus	1373.5	31	1376.9	30	60
Band 11 – SWIR	1613.7	91	1610.4	94	20
Band 12 – SWIR	2202.4	175	2185.7	185	20

Table 2.1: Sentinel 2 spectral band sensors. S2A and S2B stay for Sentinel 2A and Sentinel 2B, respectively.

late NDVI, where bands four and eight are used. Both of them with 10-meter spatial resolution. Now we have enough knowledge to build the dataset. Navigate to chapter 4 to read about the dataset, while the text will continue chronologically with chapter 3.

Chapter 3

Deep learning

The main topic in this chapter is Generative adversarial network, but because this is a more advanced part of the deep neural network domain, prior knowledge is required. I believe that readers of this thesis should have such knowledge. Therefore, I decided to skip commonly repeated terms: neuron, perceptron, deep neural network, activation function, loss function, cost function, forward pass, stochastic gradient descent, backward propagation, chain rule, convolution neural networks, therefore convolution, pooling, dense layers and so forth. These terms can be easily searched in the following literature. [22] [46] [27] [16].

In this chapter, we will start with a generative adversarial network. Then we move on to the deep convolutional and conditional generative adversarial networks. Next, an introduction to Pix2Pix, which I have chosen as an existing solution, and I will build on the proposal in the chapter 5. At the end of this chapter, I mention other approaches that have been considered for this thesis.

3.1 Generative adversarial network – GAN

If not mentioned otherwise this section is based on [28].

Why generative adversarial networks when there are other generative methods? Boltzmann machines, generative stochastic networks, variational autoencoders or fully visible belief networks.

GAN advantages compared to other generative methods:

- parallel samples generation
- design of the generator has very few limitations
- no Markov chains needed nor the variational bound
- produce better samples than other methods, subjectively regarded

It also brings a new disadvantage. It is no longer one player game as a traditional deep neural network. Normally the objective function is optimized, but here we are looking at Nash equilibrium between two forces, which is a difficult problem.

Adversarial network, as the name suggests, has some opposition. The network is a composition of two models. Generative and discriminative models compete against each other. The generative model learns to create the best possible result, while the discriminative model determines whether a sample is from data distribution or model distribution. In analogy, the generative model is a counterfeiter that creates fake money. On the opposite side, the discriminative model is police trying to detect falsified currency. Competition improves both sides methods until the falsification is unrecognizable from the original. From now on, generative and discriminative models will be referred to as a generator and a discriminator. Typically both of them are multilayer perceptrons. [29] The discriminator uses traditional supervised learning to decide the input of origin, whether a class is real or fake.

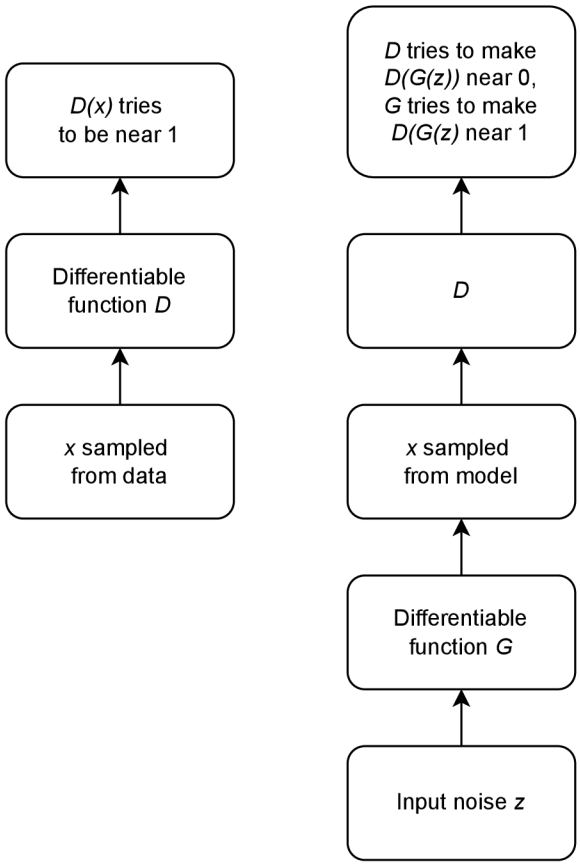

Figure 3.1: Two separated scenarios. Left scenario shows the discriminator process, whereas right scenario combines process of both the generator G and discriminator D.

The game plays in two scenarios. Scenarios separately are shown in figure 3.1. But, both scenarios are often represented by one graph, shown in figure 3.2. On the left side, training examples x are randomly sampled from the training set and used as input for the discriminator D. D uses $\theta^{(D)}$ as parameters. The discriminator goal is for $D(x)$ to be near 1. On the right side, input z is randomly sampled from the model's prior. G uses $\theta^{(G)}$ as parameters. Generator G creates a fake sample $G(z)$, which is passed on to the discriminator input. Imagine the discriminator and generator to be played by players. In the next step, both players participate. Discriminator aims to achieve $D(G(z))$ to be equal to 0, while Generator wants for the $D(G(z))$ the opposite, to be equal to 1. Both players wish to minimize the cost function J, which is defined by both player parameters. For discriminator it is $J^{(D)}(\theta^{(D)},\theta^{(G)})$ and discriminator can control it by $\theta^{(D)}$ and for

generator $J^{(G)}(\theta^{(D)},\theta^{(G)})$ controllable by parameters $\theta^{(G)}$). So, each player depends on the other player but cannot control the other player's parameters. In ideal conditions, Nash equilibrium occurs when $G(z)$ is created from the same distribution as the training data, and $D(x) = \frac{1}{2}$ for all x. Or with other words Nash equilibrium is a tuple $(\theta^{(D)},\theta^{(G)})$, that has local minimum of $J^{(D)}$ with respect to $\theta^{(D)}$ and a local minimum of $J^{(G)}$ with respect to $\theta^{(G)}$. GAN might not achieve Nash equilibrium at all. More about that, just a few lines below in the problems paragraph.

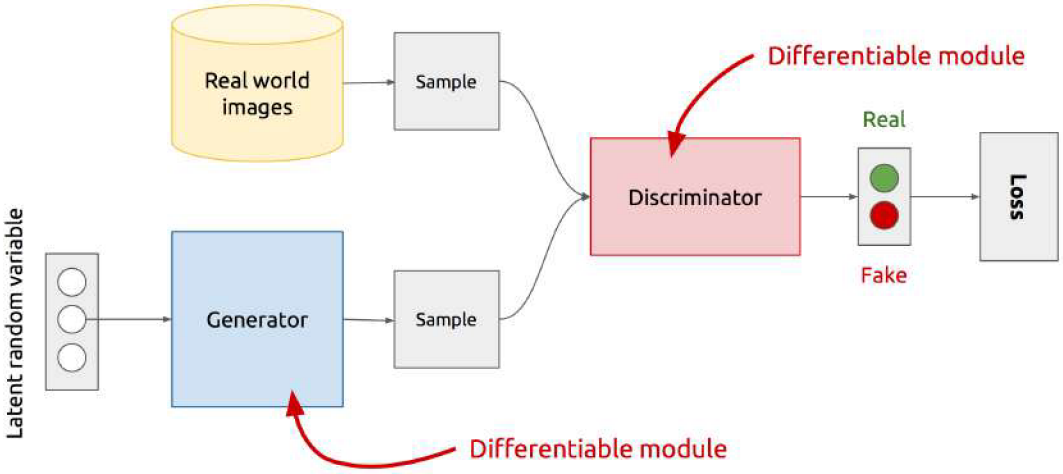

Figure 3.2: Input output flow across GAN architecture. Taken from [3].

GAN loss function:

$$\min_G \max_D V(D,G) \tag{3.1}$$

where:

- Discriminator is trying to maximize its reward $V(D,G)$
- Generator is trying to minimize Discriminator's reward (or maximize its loss)

$$\min_G \max_D V(D,G) = \mathbb{E}_{x\sim p_{data}(x)}\Big[log(D(x))\Big] + \mathbb{E}_{z\sim p_z(z)}\Big[log(1-D(G(z)))\Big] \tag{3.2}$$

where:

- $\mathbb{E}_{x\sim p_{data}(x)}$ – is expected value over all real data instances
- $D(x)$ is the discriminator's estimate of the probability that real data instance x is real.
- $G(z)$ is the generator's output when given noise z.
- $D(G(z))$ is the discriminator's estimate of the probability that a fake instance is real.
- $\mathbb{E}_{z\sim p_z(z)}$ is expected value over all generated fake instances
- log is just log probability
- $log(1-D(G(z)))$ is the inverse, searching for minimum instead of maximum

The training process: It is simultaneous stochastic gradient descent, with one step for each player. On each step, two mini-batches are sampled, one x, from the dataset and the other z from latent variables of models prior. The gradient step is updated simultaneously. Where $\theta^{(G)}$ is updated to reduce $J^{(G)}$ and $\theta^{(D)}$ is updated to reduce $J^{(D)}$.

Adam is a suitable gradient-based optimization algorithm, but others can be used.

Problems:

- non-convergence problem – even with enough time and capacity, it is possible that a solution to differential equations will never converge, e.g. sinusoid, remedy attempts – adding noise to discriminator inputs [13], or penalizing discriminator weights [41].
- mode-collapse – generator fails to output diverse samples, remedy attempts – Wasserstein GAN [14] or Unrolled GAN [34].
- evaluation of generative models – it is not clear how to quantitatively evaluate generative models. Results show that it is necessary to make an evaluation with respect to the intended application [43].

3.2 Deep convolution GAN – DCGAN

Section based on [39] if not mentioned otherwise. Deep convolution GAN is the bridge between CNN and GAN. DCGAN is mainly focused on improving the architecture constraints. An example of such architecture is shown in figure 3.3.

Key constraints:

- fully connected hidden layers are removed
- use batch normalization in both the generator and the discriminator
- replaces all pooling layers with fractional-strided convolutions for the generator and strided convolutions for the discriminator.
- activation function of generator for output layers is tanh and ReLU for all other layers.
- activation function of the discriminator is LeakyReLU for all layers

Historically, LAPGANs were one of the first attempts to scale up high-resolution images. LAP stands for Laplacian pyramid[18], which is a cascade of CNNs, learning in a coarse-to-fine manner. Each level contains a separate GAN. [23]

On the other hand, DCGAN were the first to achieve high-resolution images combined with CNN and GAN. Successfully shows that GANs are competent to learn and to generate good looking representations on restricted domains, e.g. bedrooms. In figure 3.4, DCGAN learnt vector arithmetics, and such ability opens the possibility to train more complex distributions with fewer data.

Deep convolutional GAN is one of the most important parts. Without it, there would not be an easy and efficient way to use CNN together with GANs. Another important upgrade is mentioned in the next section.

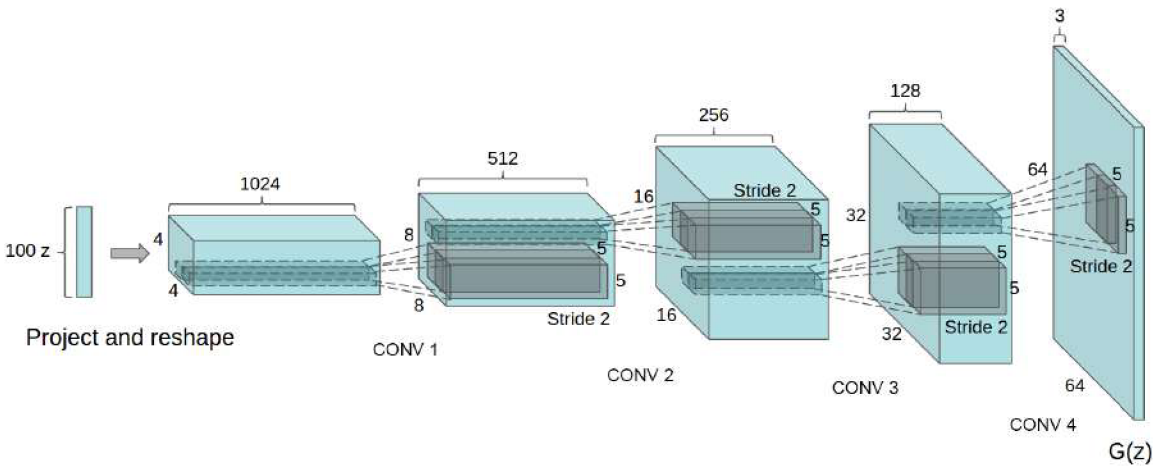

Figure 3.3: DCGAN gnerator architecture. From the left, one hundred dimensional uniform distribution z mapped out to convolutional representation with many feature maps. Four fractionally-strided convolutions with 64 x 64 output image as a result. Taken from [39].

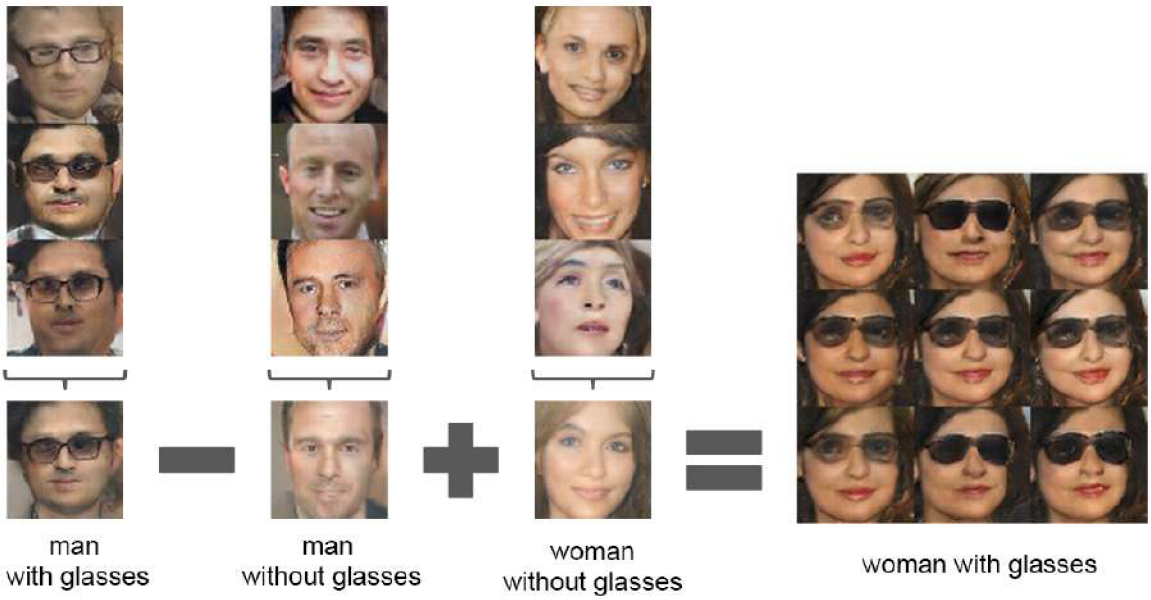

Figure 3.4: The visual concept of vector arithmetics. DCGAN learn to distinguish the concept of glasses from the gender concept. Taken from [39].

3.3 Conditional GAN – cGAN

A simple modification to GAN, where a model is conditioned on extra information for better multi-modal learning or simply because we wish to condition both parts of GAN. The difference between Vanilla GAN and cGAN is shown in figure 3.5.

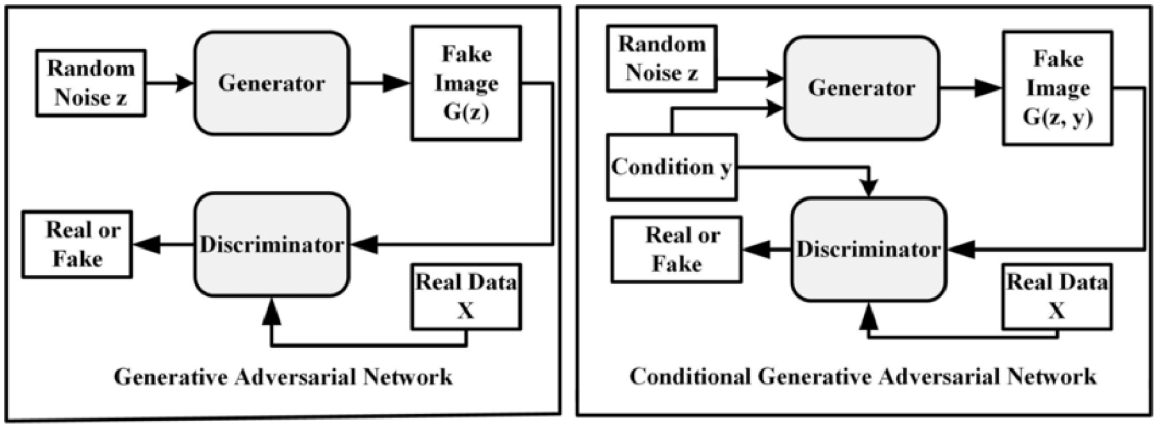

Figure 3.5: Difference between GAN and conditional GAN. In the conditional GAN graph, 'Condition y' is added. Another change is in 'Fake image', which now has two variables instead of one. Taken from [20].

It is possible to use cGAN architecture only if both the generator and discriminator are conditioned on additional information y, where y could be anything: class label, important data or auxiliary information. The objective function look like this:

$$\min_G \max_D V(D,G) = \mathbb{E}_{x\sim p_{data}(x)}\Big[log(D(x|y))\Big] + \mathbb{E}_{z\sim p_z(z)}\Big[log(1-D(G(z|y)))\Big] \tag{3.3}$$

where, the change for the discriminator is x conditioned by y. Similarly, the generator input noise z is conditioned by y.

This section was based on [35]. Conditional GAN is a cornerstone of many other research applications. One of such is our next topic.

3.4 Pix2Pix – Image-to-image translation

This paragraph is taken from [31] if not mentioned otherwise.

Image to image translation learns a mapping from the input image to the output image. Pix2Pix, the name of the cGAN, is the shortcut for mapping pixel to pixel. Such a task is common in computer vision. It introduces a wide range of applications where the image to image translation is used. Instead of a specific loss function for each application, this approach uses the same loss function to achieve the best Pix2Pix result for every scenario. During the training, the Pix2Pix loss function learns the mapping between input and output. Results of this approach are shown in figure 3.6.

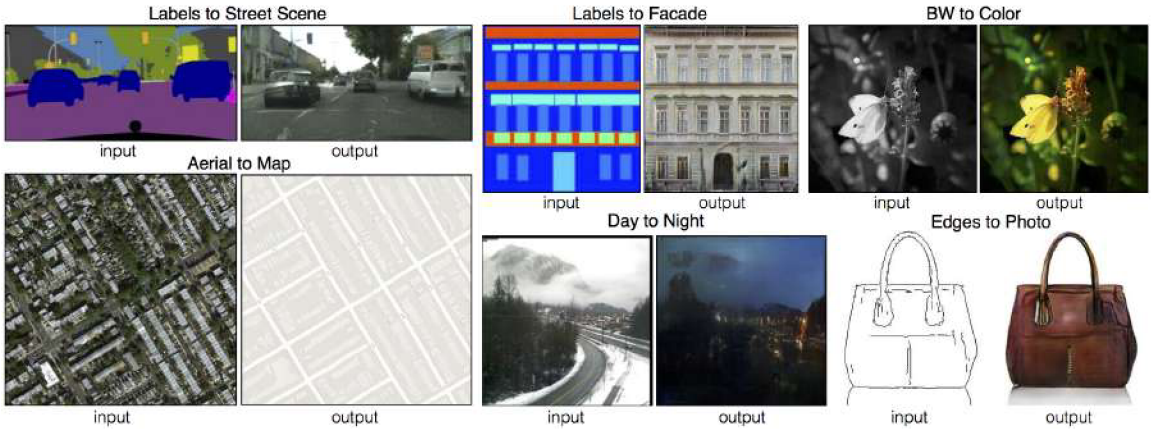

Figure 3.6: Pix2Pix cGAN usage variety. All usage cases use the same architecture where just training data are changed. Taken from [31].

The architecture is based on DCGAN mentioned in section 3.2. The main changes are: U-Net architecture is used for the generator, and a convolutional PatchGAN classifier is used for the discriminator. Conditional GAN is suitable for the task where the condition is the input image.

U-Net, introduced in [40] and shown in figure 3.7, is an encoder-decoder network with skipped connections. It ensures that information can be bypassed across the network and it is no longer required to propagate the information across all layers of the network. The encoder is downsampling the image to feature maps by unpadded convolutions followed by a max-pooling operation with stride 2. Therefore, the next level has two times more feature channels. Oppositely, the decoder upsamples the multi-channel feature maps to the segmentation map. Each level obtains half the feature channels from upsampling, concatenated with a copied feature map from the encoder. Concatenation is followed by double convolution, each using ReLU. In the end, there is a $1x1$ convolution mapping feature vector with a desired number of classes.

PatchGAN – Introduced in the same paper as Pix2Pix [31], arguments that L1 already enforce the correctness at low frequencies. To fill the gap of high frequencies, they designed discriminator architecture called PatchGAN, which classify by $N * N$ patches, where N is the size of the patch. The discriminator runs over the whole picture and, for each patch, says whether the patch picture is real or fake. The average of all responses is the final output D.

Before we can start to talk about an **objective** function, let's short dive into some regression loss functions. The mean square error or L2 distance is considered to create more blurry results. Instead of it, the mean absolute error is used or L1 distance. Mean absolute error formula, where n is number of errors, x is the true value and x_i is the measurement.

$$\frac{1}{n}\sum_{i=1}^{n}|x_i - x| \tag{3.4}$$

The cGAN formula (3.3) stays the same, the L1 loss in the cGAN context:

$$\mathcal{L}_{L1}(G) = \mathbb{E}_{x,y,z}\left[||y - G(x,z)||_1\right] \tag{3.5}$$

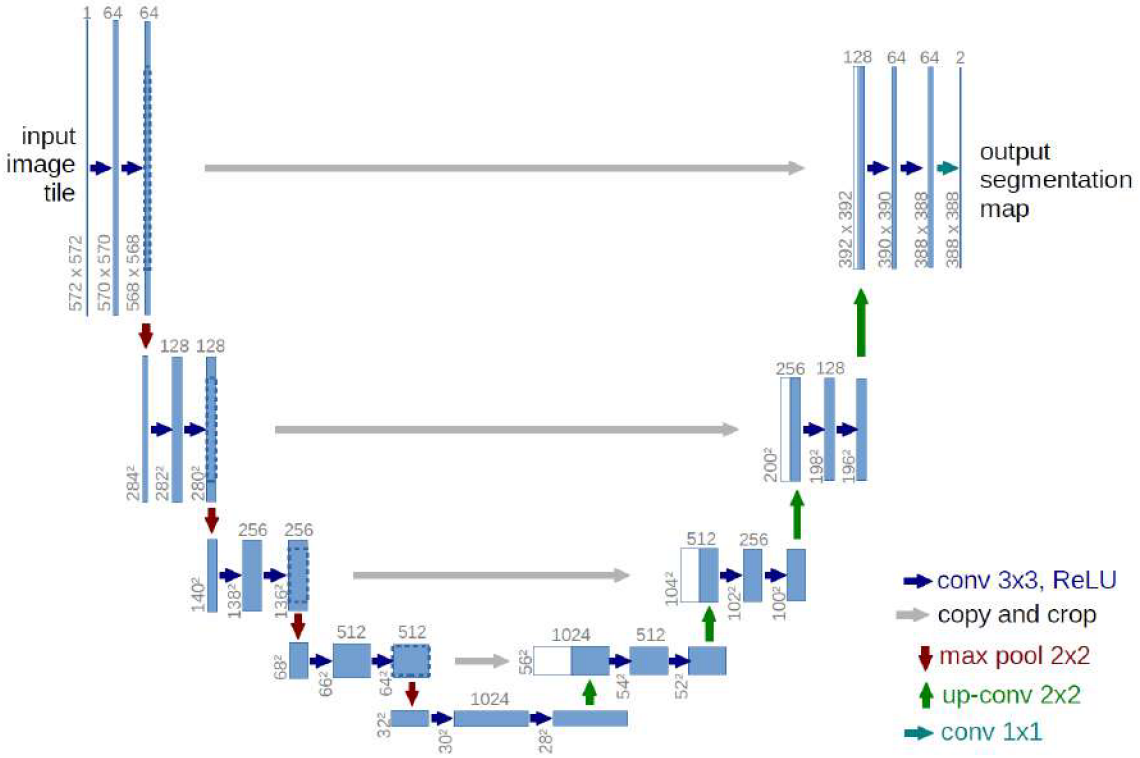

Figure 3.7: U-Net architecture. The left side of the image is an encoder and the right is a decoder. White boxes corresponds to copied feature maps and blue boxes corresponds to a multi-channel feature map. Arrows represent different operations. Taken from [40].

With that let's look at final objective, generator loss:

$$G^* = arg \min_G \max_D \mathcal{L}_{cGAN}(G,D) + \lambda \mathcal{L}_{L1}(G) \tag{3.6}$$

where: $\mathcal{L}_{cGAN}$ is the right side of cGAN equation (3.3), and λ value is equal to one-hundred.

Noise is no longer used in the generator. In figure 3.5, noise z is used as input in the generator in both GAN and also cGAN. The noise was found to be ignored by the generator, therefore, ineffective. As a replacement, the dropout is used for several layers of the generator.

The nearest solution example to this thesis is Aerial to map shown in figure 3.6. With this approach, the best results were achieved when batch size equal is set to 1, and training length is set for 200 epochs.

3.5 Other approaches

Even though the image-to-image is used later in the practical part of this thesis, there are approaches that were studied and considered not to be prefered.

Style transfer – can separate the content and style from a source image and apply it to a target. The only non-GAN approach mentioned. There are two images: style and content image, where the style is extracted from the style image and applied to the content image. Example is shown in figure 3.8. It is achieved by a deep convolution neural network building on top of the VGG network. Style transfer is considered to be achieved if the generated image 'looks like an image' and when the content is kept and style is applied. Mathematically viewed, such identification is not precise at all. More about this approach, which this paragraph is based on, can be found in [26].

Figure 3.8: Example of style transfer. Content image shows Neckarfront in Tübingen, style image is The Starry Night by Vincent van Gogh. Modified from [26].

CycleGAN – based on the [49], building upon Pix2Pix, mentioned in section 3.4. The main difference is usage of unpaired data, so no need for paired dataset. To understand a difference take a look at figure 3.9.

With an assumption that there is a connection between the domains of unpaired data, CycleGAN is trying to learn such a connection. It is achieving it by two mapping functions that are introduced: $G : X \longrightarrow Y$ and $F : Y \longrightarrow X$. But, the trick is the cycle consistency loss, which tells us that we should arrive at the same place where we have started. An example can be the translation between languages. We should have the same sentence when we transfer a French

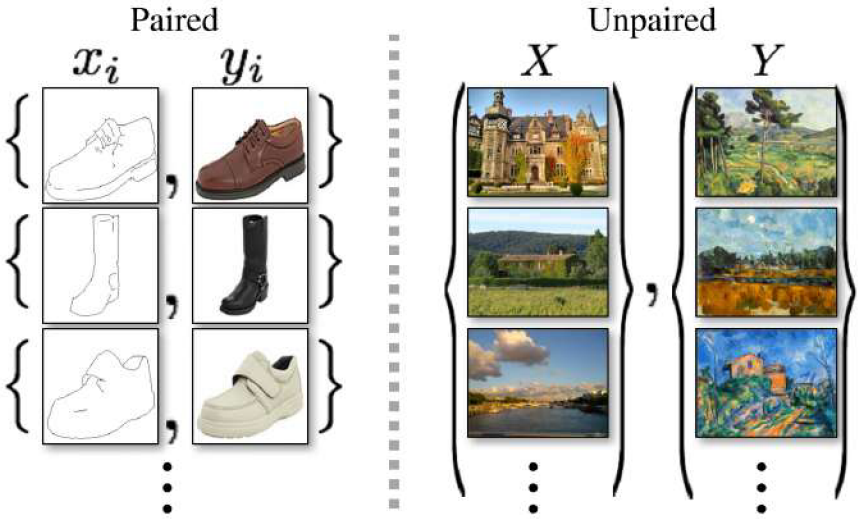

Figure 3.9: Paired vs unpaired images. Paired images, as naming suggest are pairs where, each x_i is connected with y_i. Whereas in unpaired data a source set and a target set has no information which x_i corresponds to which y_i. Taken from [49].

sentence to English and then back to French. The same is done vice-versa. The sentence from English can be translated into French and back. Many tasks can be written as a transfer between domains. Examples of what the CycleGAN is capable of are shown in figure 3.10. Architecture is more complicated than Pix2Pix, and I have the image pairs, so there is no need to use unpaired data. Although it can be an exciting experiment, it is not part of this thesis.

There is a lot of research around the GANs, and there are other GAN types that did not make it to the thesis. Although, I think the important ones for this thesis were already mentioned.

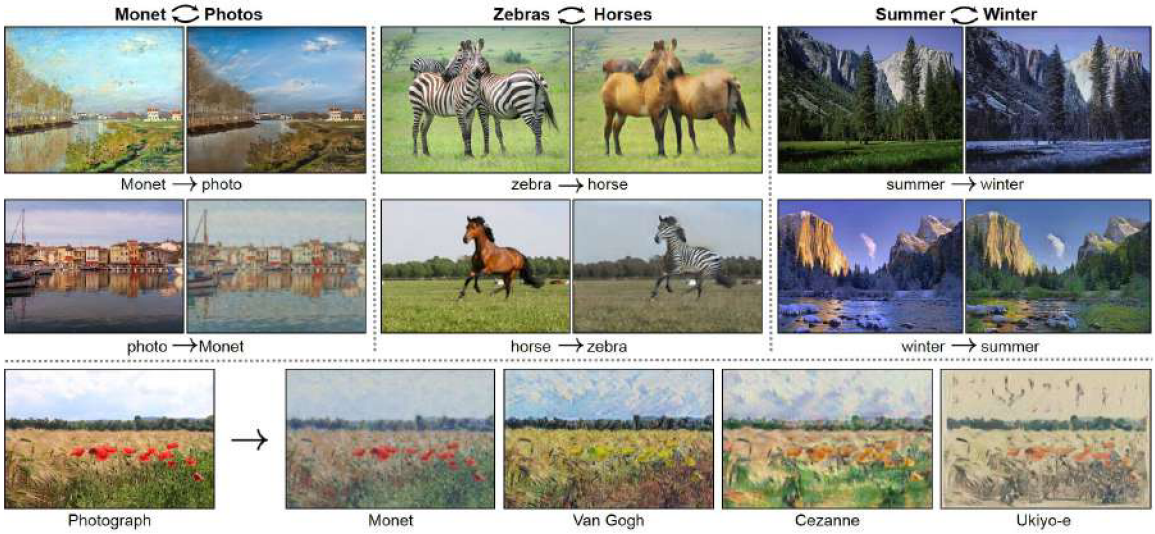

Figure 3.10: CycleGAN usages. Bold text at the top are domains and description with arrow is explaining which picture is real and which is generated. At the bottom photograph to painting by different famous artists. Taken from [49].

Chapter 4

Datasets

The chapter describes all information about the dataset. The company World from Space[1] provided data. Given data are explained in the section received data, where only a small amount of work was done by me. On the contrary, in section 4.2, the whole dataset was created by me. Final section of this chapter 4.3, provides insights into dataset properties.

4.1 Received data

Data are divided into optical and radar. Optical data originate from Sentinel 2 program. Similarly, radar data originate from Sentinel 1. Example of Sentinel 2 image is shown in figure 4.1.

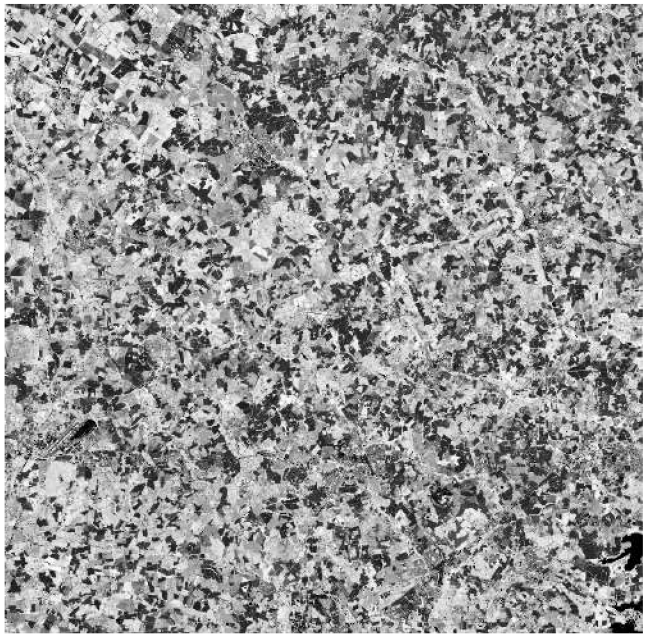

Figure 4.1: Example of an optical image from Sentinel 2. The figure shows an area close to Cambridge in the United Kingdom and was taken on the 16th of April. The area width is 6588 pixels, and the height is 6309 pixels with a pixel resolution of 10x10 meters.

[1] https://worldfrom.space

Data consists of different places around Europe. Each place has its bounding box and has a different size. The bounding box is an area defined by two longitudes and two latitudes.

These places are located in the following countries: Bulgaria, Czech Republic, Denmark, France, Germany, Italy, Latvia, Romania, Russia, Spain, Sweden, Ukraine, United Kingdom.

The condition for optical data is cloudless weather. It can never be really cloudless, but with as infrequent clouds as possible. The decision of what is cloudy enough was made by humans and not by some algorithm or heuristic. Cloudless weather was set to at least three measurements in a row. Length between measurements varies, but it is mostly five days. Such a requirement of three measurements in the row enables an option to pass knowledge of past data to the neural network.

Optical and radar data are connected to pairs based on the date. Because the satellites are not static and rotate around the globe for some period, it is hard to have the exact dates. Therefore the difference in date variation is a maximum of one day. So if the optical data are from Tuesday, the radar data can be from Monday, Tuesday or Wednesday.

Three consecutive optical and radar data measurements are classified into the seasons. Seasons are an important factor in agriculture, and the crops and fields look different in every season.

Division of seasons and number of measurements per season:

1. spring – [01.03, 31.5] – 10
2. summer – [01.06, 31.8] – 10
3. fall – [01.09, 30.11] – 9

Winter is omitted because there was not enough data for the dataset. If the consecutive measurement is across multiple seasons, the season is chosen based on the start of the measurement.

Complete example: Place in Bulgaria with sentinel 2 dates 31.08.2019, 04.09.2019, 09.09.2019 and sentinel 1 dates 30.08.2019, 03.09.2019, 08.09.2019 is classified as summer.

Radar data from Sentinel 1 were prepared for dataset creation, but optical data were not. So additional work was done by me. World from Space provided me access to their code, where I could download optical data from Sentinel 2 with minor code modifications. S2 data have only one band. The band is showing the NDVI. The download process and all prepossessing steps would cost me much more time without World from Space company's help.

Big chunks of radar and optical data are waiting for dataset creation, described in the next section. Both radar and optical data are in tiff format. More about the tiff format is mentioned in the section 5.3.

Country and area	Season	S1 dates	S2 dates
BULGARIA_SUMEN	spring	2020-03-05, 2020-03-17	2020-03-06, 2020-03-11, 2020-03-16
BULGARIA_SUMEN	summer	2017-07-31, 2017-08-12	2017-08-01, 2017-08-06, 2017-08-11
BULGARIA_SUMEN	summer	2018-08-13, 2018-08-25	2018-08-14, 2018-08-19, 2018-08-26
BULGARIA_SUMEN	fall	2020-09-01, 2020-09-13	2020-09-02, 2020-09-07, 2020-09-14
BULGARIA_SUMEN	winter	2019-12-10, 2019-12-22	2019-12-09, 2019-12-17, 2019-12-22
CZECH_REP_OLOMOUC	spring	2020-03-27, 2020-04-08	2020-03-28, 2020-04-02, 2020-04-07
CZECH_REP_OLOMOUC	summer	2019-08-25, 2019-09-06	2019-08-26, 2019-08-31, 2019-09-05
CZECH_REP_OLOMOUC	fall	2018-09-29, 2018-10-11	2018-09-30, 2018-10-05, 2018-10-10
DENMARK_HOLSTEBRO	summer	2018-07-06, 2018-07-18	2018-07-07, 2018-07-12, 2018-07-17
FRANCE_REIMS	fall	2020-09-11, 2020-09-23	2020-09-12, 2020-09-17, 2020-09-22
GERMANY_ERFURTH	spring	2020-03-31, 2020-04-18	2020-04-01, 2020-04-06, 2020-04-11
ITALY_PESARO	spring	2020-04-02, 2020-04-14	2020-04-03, 2020-04-08, 2020-04-15
ITALY_PESARO	summer	2019-06-24, 2019-07-06	2019-06-25, 2019-06-30, 2019-07-05
ITALY_PESARO	fall	2018-09-22, 2018-10-04	2018-09-23, 2018-09-28, 2018-10-03
KALININGRAD_GUSEV	spring	2018-04-07, 2018-04-19	2018-04-08, 2018-04-13, 2018-04-18
LATVIA_LITHUANIA	spring	2019-04-17, 2019-04-29	2019-04-18, 2019-04-23, 2019-04-28
LATVIA_LITHUANIA	summer	2019-05-29, 2019-06-10	2019-05-30, 2019-06-04, 2019-06-09
LATVIA_LITHUANIA	fall	2020-09-18, 2020-09-30	2020-09-19, 2020-09-24, 2020-09-29
MOLDOVA_ROMANIA	spring	2020-03-28, 2020-04-09	2020-03-29, 2020-04-03, 2020-04-08
MOLDOVA_ROMANIA	summer	2019-08-20, 2019-09-01	2019-08-19, 2019-08-27, 2019-09-01
MOLDOVA_ROMANIA	fall	2020-09-10, 2020-09-22	2020-09-10, 2020-09-15, 2020-09-22
SPAIN_BADAJOZ	spring	2019-03-11, 2019-03-29	2019-03-11, 2019-03-16, 2019-03-23, 2019-03-28
SPAIN_BADAJOZ	summer	2017-07-13, 2017-07-25	2017-07-14, 2017-07-19, 2017-07-26
SPAIN_BADAJOZ	summer	2018-08-01, 2018-08-25	2018-07-31, 2018-08-08, 2018-08-13, 2018-08-20, 2018-08-25
SPAIN_BADAJOZ	fall	2018-10-18, 2018-10-30	2018-10-17, 2018-10-24, 2018-10-29
SPAIN_BADAJOZ	winter	2019-12-30, 2020-01-11	2019-12-31, 2020-01-05, 2020-01-12
SWEDEN_MALMO	spring	2020-04-10, 2020-04-22	2020-04-09, 2020-04-16, 2020-04-21
UKRAINE_KONOTOP	spring	2020-03-12, 2020-03-24	2020-03-13, 2020-03-18, 2020-03-25
UKRAINE_KONOTOP	summer	2020-07-15, 2020-07-27	2020-07-16, 2020-07-21, 2020-07-28
UKRAINE_KONOTOP	fall	2020-09-01, 2020-09-13	2020-09-01, 2020-09-06, 2020-09-14
UK_CAMBRIDGE	spring	2020-04-15, 2020-04-27	2020-04-16, 2020-04-21, 2020-04-26
UK_SALISBURY	fall	2020-09-14, 2020-09-26	2020-09-13, 2020-09-21, 2020-09-26
UK_WALES	summer	2018-06-21, 2018-07-03	2018-06-22, 2018-06-27, 2018-07-02

Table 4.1: Complete information about the data source. The S1 dates column have only start and finish date. Based on the date range in the S1 dates, I downloaded the S1 data. S2 data were already downloaded and provided by the World from Space company.

4.2 Dataset preparation and creation

To clearly separate given from created, everything from now on was implemented by the author of the text, if not mentioned otherwise.

The dataset creation consists of the multiple steps, where all the steps are repeated for each region:

1. Merge tiles – Satellite provides the parts of the whole place. Parts are called tiles, and they need to be merged. S1 tiles are already merged, but S2 tiles are not. Figure 4.2 explains the process.

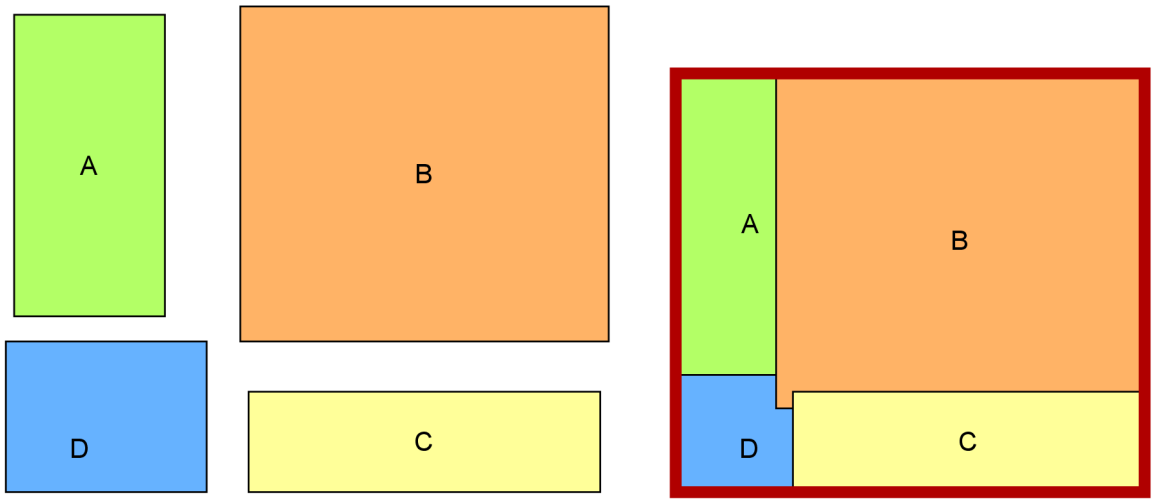

Figure 4.2: Multiple overlapping tiles are merged into one image. Each tile has its georeferenced position, therefore is simple to merge them.

2. Clip S1 and S2 by the same bounding box, S1 and S2 are captured at different angles and differ in size. Based on the given bounding box, both images are cut to the same size. Notice that in figure 4.3, the bounding box is intentionally not as big as it could be. This is because the bounding boxes were created manually, not by an algorithm.

3. Split big areas to small pictures – Size of all pictures is 256x256. The 256x256 grid can be seen in figure 4.4, there is still some space on the edges of the bounding box but not enough for another image. Often clouds and water are unwanted elements in S2 picture. Mentioned elements have value near zero, which is based on NDVI calculation shown in equation 2.1. The other reason for value to be zero in S2 pictures are dead pixels[48], [21].

   The aim of this solution is to be used for agriculture, so we do not aim for water areas, so one less problem. To remove dead pixels but keep a reasonable dataset size, I wanted to aim for a dataset size of around 10 thousand samples for each season. I created a small experiment for S2 data. The result of this experiment is shown in table 4.2. Five percent tolerance results in 891 pictures for the measured on the *spring* season with *BULGARIA_SUMEN*. Multiplied by the number of measurements in the spring dataset, we get $891 * 10 = 8910$ pictures. I decided on 5% tolerance on zero pixels to keep the dataset rel-

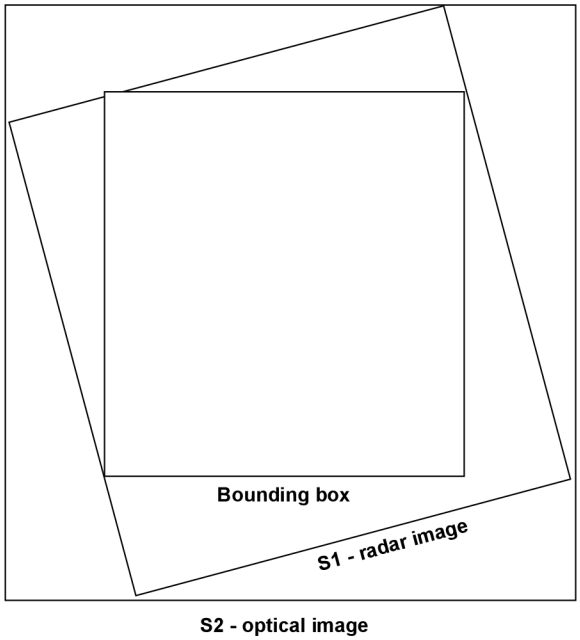

Figure 4.3: Both pictures have same size, thanks to bounding box.

atively big. The decision is quite inaccurate because each place covers a different surface size, and I experimented on only one area.

S1 values can have exact zeros in the picture only if the picture is outside of the bounding box. Other values were considered valid. However, there is some margin of error, mostly with pixels with the highest and lowest values. Additional experimentation would be needed to filter S1 data.

percentage	**number of tiles selected**	**percentage selected**
0 %	90/1140	7.9 %
1 %	563/1140	49.4 %
2 %	700/1140	61.4 %
3 %	791/1140	69.4 %
4 %	850/1140	74.6 %
5 %	891/1140	78.2 %

Table 4.2: Tolerance experiment results on S2 images. The table shows the number of tiles selected for the dataset based on increasing tolerance. Measurement was done on all dates in Bulgaria from the spring season.

4.3 Dataset properties

In this stage, one image from the dataset has a height and width of 256 pixels, and it is still a tiff file with one band for the optical image and two bands for the radar image. Pixel resolution is 10x10 meters, so each dataset image displays an area of 2560 square meters. Dataset properties are shown in table 4.3. Normalization values are -38.0/31.0 for radar pictures and 0.0/1.0

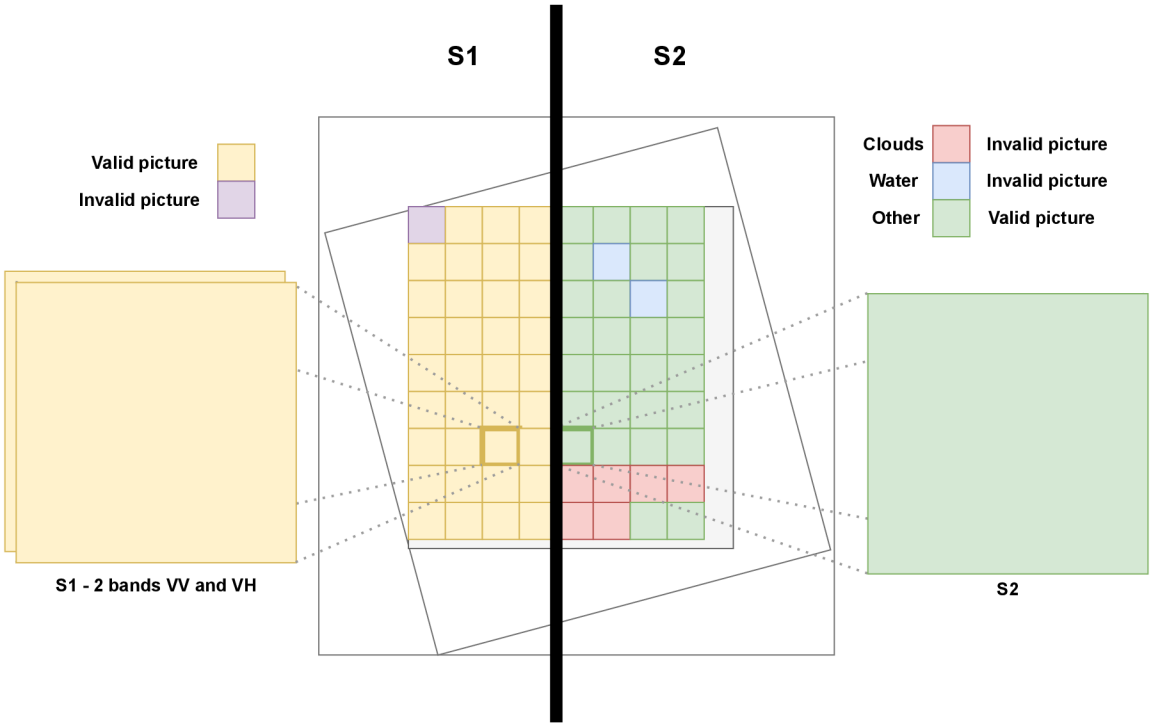

Figure 4.4: Grid creation to cut pictures to the desired size of 256x256. Also shows the difference between S1 and S2 picture creation. S1 picture is invalid when it is outside of bounding box, where S2 is also invalid if the image is mainly covered by water, clouds or dead pixels.

for NDVI images. That means no other value in the entire dataset is greater than a normalization maximum and lower than a normalization minimum. Interestingly, all values in NDVI are positive even though NDVI allows negative values. From the dataset or generated by the neural network, all pictures in this thesis are normalized into $< 0 - 1 >$ interval.

Season	# of pairs	Size
spring	11128	11.6 GB
summer	13699	14.4 GB
fall	8674	9.2 GB
all	33501	35.2 GB

Table 4.3: Shows the size of the datasets per season. Each pair consist of S1 and S2 image with the same filename. Size is summing up the S1 and S2 data. The tiff files are taking up a huge amount of space. That makes the work with them more computationally intensive and, therefore, time-intensive.

Example of the dataset filename with key, with date format YYYY-MM-DD:

```
SPAIN_BADAJOZ_2019-03-11_2019-03-29-2-7-16.tiff
<Country>_<Area>_<DateFrom>_<DateTo>-<MeasurementID>-<Row>-<Column>.tiff
```

Measurement ID is based on which index is the picture from. Our example has four measurements. Look at the row with Country and area SPAIN_BAJADOZ and Season spring in table 4.1. Measurement 2, with indexing from zero, means that we are working at the third date in S2 column = 2019-03-23. Row and column are square ids from figure 4.4.

Example of dataset images is shown in figure 4.5. All generated images by our model and dataset pictures in this thesis are shown with a Viridis colour map, shown in figure 4.6. The higher the data value, the lighter pixel. On the contrary, the pixel with the lower value is darker. The zero is interpreted as dark blue and one as yellow.

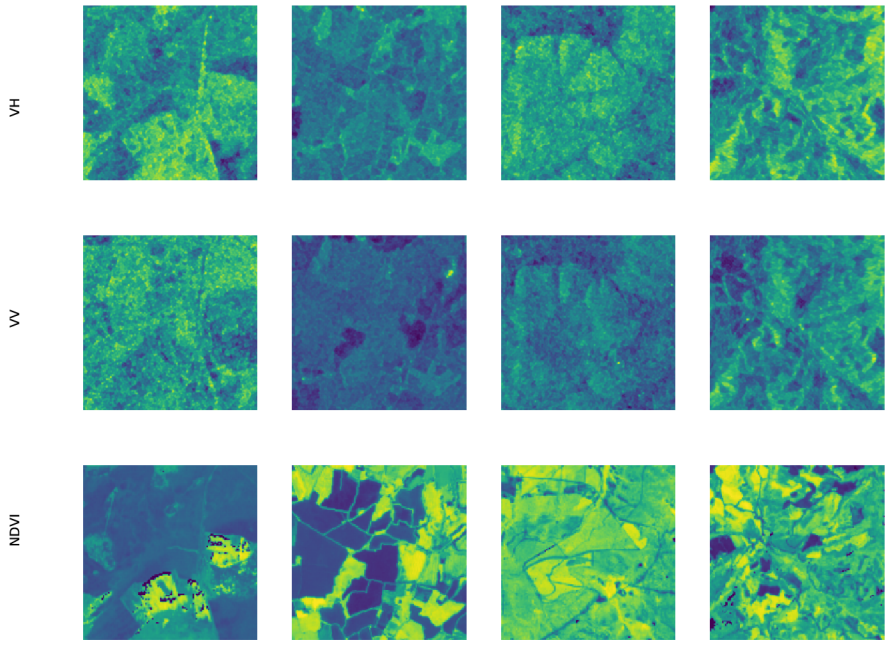

Figure 4.5: Examples from the Spring dataset. The first two rows show two bands of radar pictures, and the third row shows the optical image with calculated NDVI.

Figure 4.6: Viridis colour map.

Chapter 5

Proposal, Implementation and Training

The purpose is to generate the optical data with past optical and radar data and actual radar data, but no actual optical data. Optical data are not available during bad weather conditions, mainly when clouds occur. In theory, radar data should bring additional beneficial information to the neural network. Models are built up with iterations starting from zero, from the simplest variant. The difference between models is the number of inputs. The results between the model versions will provide us with information on whether the neural network was capable of learning such output, whether the more inputs influenced the results and how. We also want to determine how much the network uses the difference between actual and previous radar data. Such a comparison is made with the first and the second model results. With an understanding of motivation, we can move on to the practical part of this thesis.

Firstly, the Pix2Pix model will be introduced as our base model. I will dive into the architecture details of all parts, generator, discriminator and the whole network. Secondly, the reader will find out what has changed architecturally by adding more and more inputs. After that, the difference between model 2 and model 3 is explained, which brings us to the problem of choosing files for the dataset when multiple inputs with different dates are added. More implementation details are shared right afterwards. At the end of this chapter, I will share the training details and used technologies.

5.1 Models proposal

The idea was to start simple. Implement the Pix2Pix network with radar picture as input and optical picture as output. I called it model 0. The most straightforward model architecture is explained in the following section. The modifications will be explained afterwards.

In next paragraphs the Pix2Pix paper [31] will be mentioned multiple times. A base of my implementation is inspired heavily by Pix2Pix tutorial [17]. The architecture is based on the Pix2Pix paper. We have already learned that there are two parts for each GAN, generator and discriminator, so we start to discuss architectural properties with them.

Discriminator architecture, shown in figure 5.1 with convolutions:

`C64-C128-C256-C512-C512`

The patch size is calculated as follows.

$$K_{l-1} = K + S * (K_l - 1) \tag{5.1}$$

Where:

- K_{l-1} – receptive field of $l-1^{th}$ layer
- K – kernel size
- S – stride value
- $(K_l - 1)$ – output receptive field of l^{th} layer

It is calculated recursively starting from the output. Always the layer before is calculated. The calculation should be made for both width and height, but in our case, all convolution layers have the same width and height, so it is enough to calculate just one of the numbers.

$$\begin{aligned} Output &= 4 + 1 * (1 - 1) = 4 \\ C512_{last} &= 4 + 1 * (4 - 1) = 7 \\ C512 &= 4 + 2 * (7 - 1) = 16 \\ C256 &= 4 + 2 * (16 - 1) = 34 \\ C128 &= 4 + 2 * (34 - 1) = 70 \\ C64 &= 4 + 2 * (70 - 1) = 142 \end{aligned}$$

Based on the Pix2Pix paper, it is recommended to use $70x70$ PatchGAN, but my architecture uses $142x142$. You may ask, why not change it back? I have already trained plenty of networks with this setup. This unintentional difference made by inattention error should not cause a big difference. In the paper, there is mentioned that a patch with a size of $286x286$ does not provide better visual results compared to $70x70$ and has many more parameters. Therefore it may be harder to train.

The discriminator is using Binary Cross-entropy as a loss function. The discriminator's weights are divided by 2, so the discriminator is slowed down to not adapt as fast to the generator as it usually would. All convolution layers have padding set to the same. Same means zero-padding, but not quite. The image is padded with zeros to get fully covered by a specified filter and stride. Stride is 1, if not mentioned otherwise in graphs. Two images with the size of 256x256 pixels and two bands as input to the network.

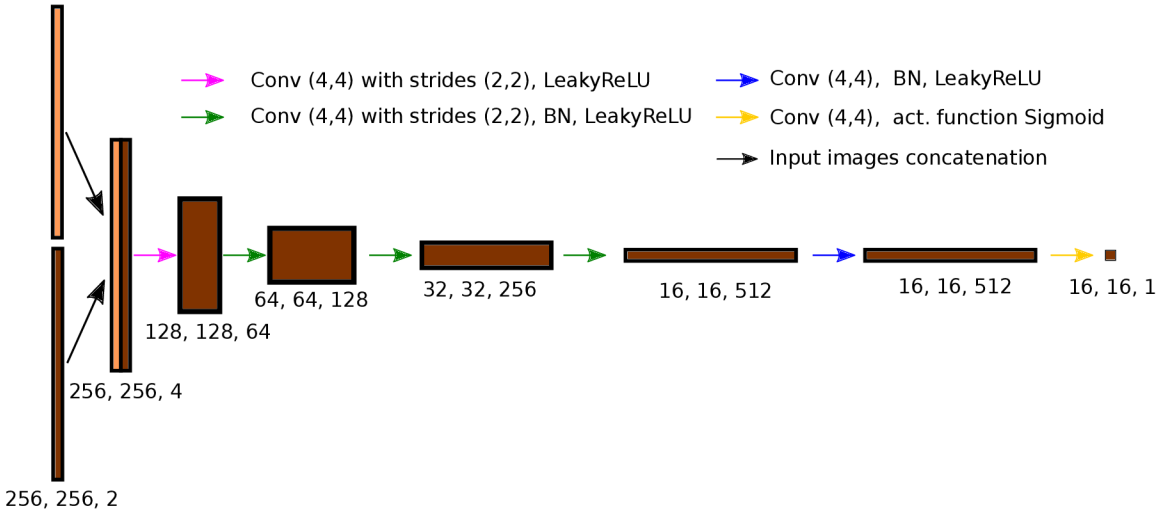

Figure 5.1: Discriminator architecture. PatchGAN 142x142. Takes two pictures as input, generated image and ground truth. Each pixel in the 16x16 output image represents the believability of a 142x142 from the input picture. Patches overlap as a picture is not big enough. In other words, output values represent the likelihood probability that a patch in the generated image is real.

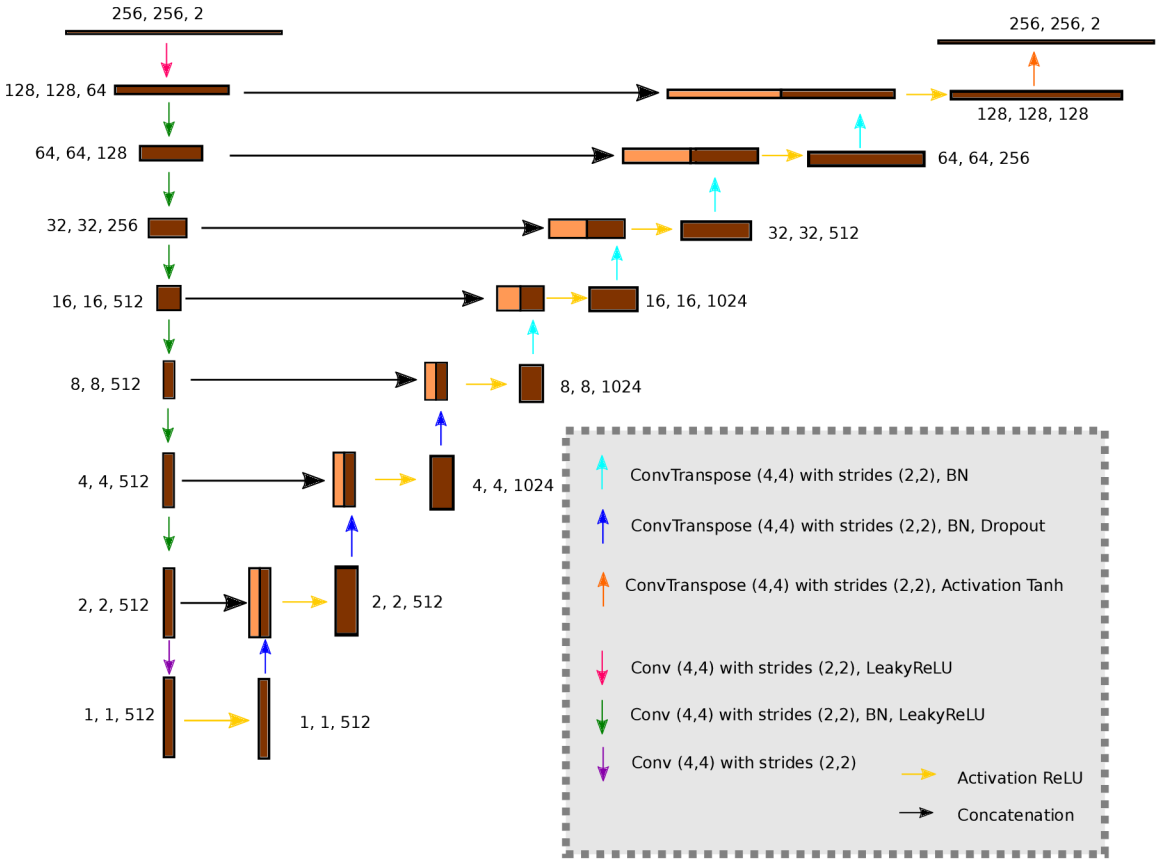

Figure 5.2: Generator U-Net architecture. Batch normalization is not applied to the first encoder layer. The activation function at the end is hyperbolic tangent function. Based on Pix2Pix paper dropout is set to 50% and is part of blue arrows.

The generator architecture shown in figure 5.2 with following convolutions structure, where C is convolution and D is dropout.

Encoder:

`C64-C128-C256-C512-C512-C512-C512-C512`

Decoder:

`CD512-CD512-CD512-C512-C512-C256-C128-C64`

Simply sad, the generator is trained by the discriminator. As we already know, to get adversarial loss, the first part on the right side of equation (3.6), the discriminator output is needed. To train the generator using the results from the discriminator new composite model was created, shown in figure 5.3. The new composite model uses the discriminator output to update generator weights. Discriminator weights in the composite model are set up as non-trainable.

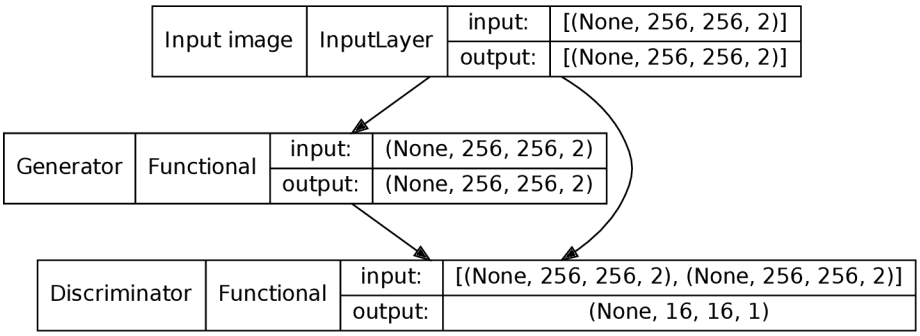

Figure 5.3: Composite GAN model. Input picture is passed to generator. Generator creates an image, and both input image and generated image are passed to discriminator. Discriminator output is not only used to classify whether the image is real, but also to update generator loss.

The following information was taken from Pix2Pix paper. Adam is used as an optimizer with learning rate equal to 0.0002 and $\beta_1 = 0.5$, $\beta_2 = 0.999$. All LeakyReLU functions have $\alpha = 0.2$. Weights are initialized with Normal distribution where mean is set to 0 and standard deviation is set to 0.02.

All about model 0. was said. Let's talk about other models from figure 5.4. In all of these models, the result is still the same. We aim to train the generator to create the most similar NDVI picture to reality. So we expect the generator to show us NDVI from the actual day. In model 0, only the radar picture from the same day was provided, which, as we know, has two bands. In addition to model 0, the older NDVI picture from the last revisit time was added, model 1. With old NDVI pictures, the pictures should look at least similar. The last revisit time is about 5 or 6 days before the output. In addition to model 1, the older radar picture was provided to the network input, model 2. This additional radar data might help because the network now has a comparison between the last and actual radar picture. The question to ask is, is model 2. capable of learning this relationship? If so, is the mentioned radar relationship strong enough to influence

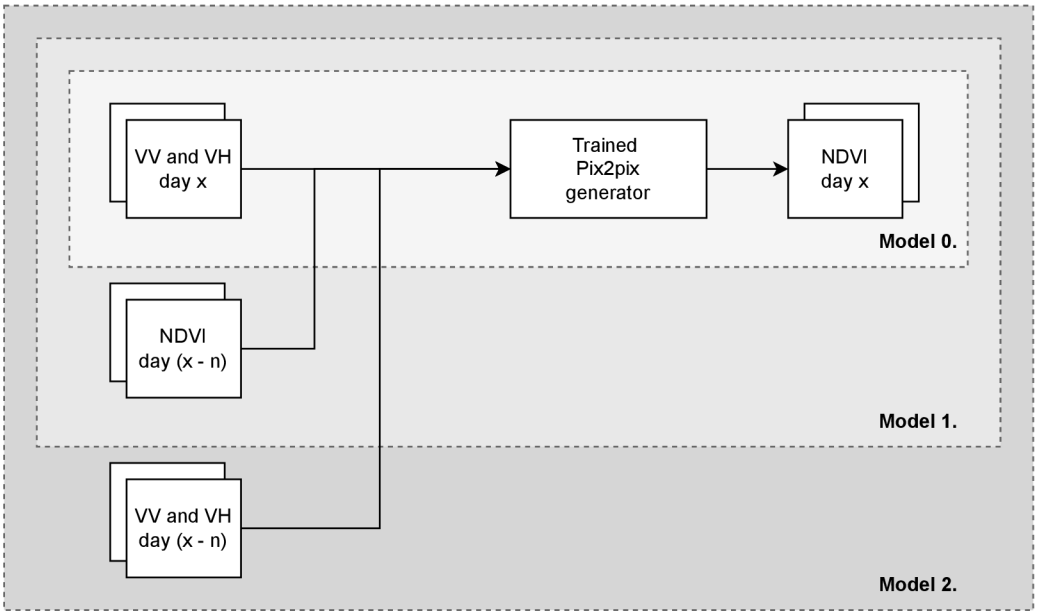

Figure 5.4: Multiple models with input and outputs of trained Pix2Pix network, where x is date and n is revisit time. It starts with the simplest model and gradually adds more inputs to compare the results.

NDVI picture? The answers are provided in section 6.3.

Let's get back to the architecture. How is the question of multiple inputs solved in architecture? The discriminator remains the same. The generator concatenates inputs before they are sent via an U-Net connection. An example with two inputs, correspondents to my model 1, is shown in figure 5.5. For more inputs, each input has its own copy of the encoder. The concatenation process is described as follows. For each copy of the encoder for each layer, two layers are concatenated. Then the result is concatenated with the third layer, and so on, until all layers are concatenated together. The result is passed via the U-Net connection to the decoder.

The concatenation result of the penultimate layer is passed to the ultimate layer of the encoder. My solution is to pass the concatenated data to the last encoder layer from all copies. I consider this as my architectural mistake.

Also, the composite GAN model is changing with more inputs. The discriminator remains the same, where only the input image and generated image are passed to the discriminator. But all input images are passed to the generator.

As my work progressed, at this point in time, I already had results from models 0, 1 and 2. I have added model 3. The reason was better data distribution, more about that in the results section 6.4. The model is shown in figure 5.6, where the only difference compared to model 2 is a longer revisit time period. The revisit time period has doubled, so we skip over one measurement when choosing the data to train on. For example, the first measurement date is compared with the third, and the second is ignored.

I retrained model 3 with PatchGAN 70x70 and fixed concatenation in the generator. This model will be mentioned as fixed model 3. In the figure 5.5, I had concatenated correctly until

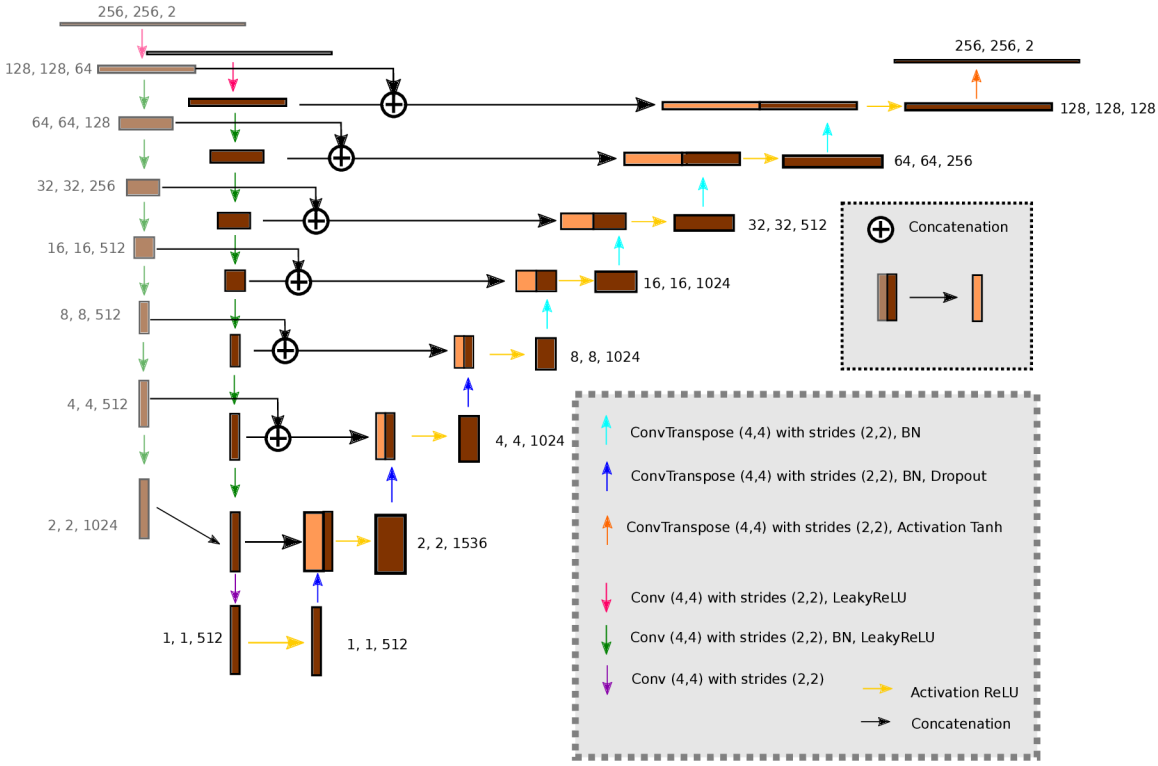

Figure 5.5: Model 1 generator U-Net with two inputs. In comparison to one input the concatenation operation was added between the inputs before they are passed via U-Net connections between encoder and decoder.

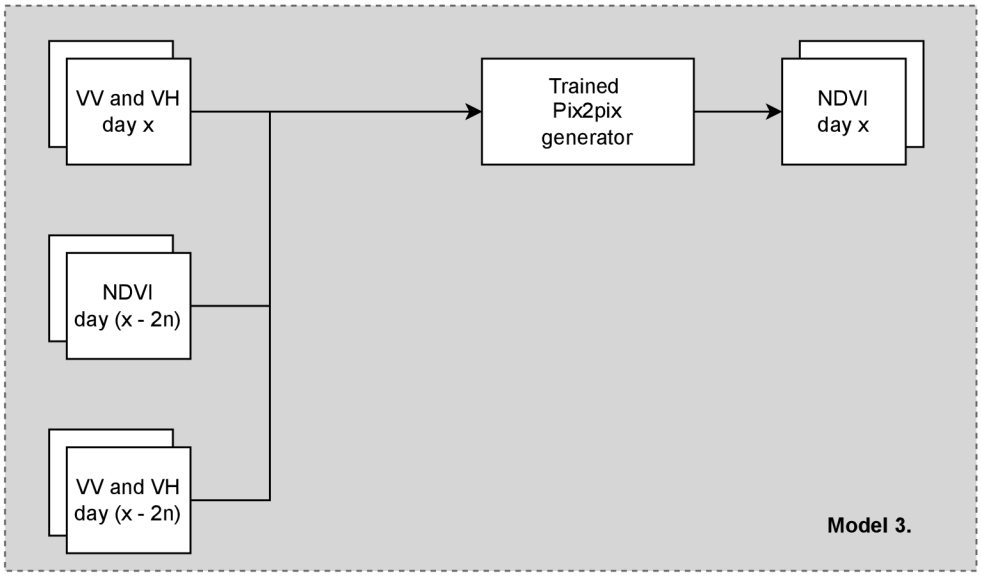

Figure 5.6: Model 3 inputs and outputs.

the penultimate encoder layer. That is fixed in this model, and the concatenated feature map from the penultimate layer is no longer sent to the ultimate layer. The concatenation of the penultimate encoder layer looks the same as those from the top. For the ultimate encoder layer, convolution (purple arrow) is done for each input, and then the concatenation is applied. The output of concatenation is passed to the ReLU activation function. The decoder remains the same.

5.2 Loss functions

As the name suggests, the main topic is to explain the calculation of the loss function.

Discriminator loss Discriminator weights are updated twice. Once from input and target image which is labelled as „Real“. The second time with input and generated image labelled as „Fake“. The real label is represented as an array of ones, and oppositely, the fake label is an array of zeros. This process is shown in figure 5.7. The weights are then optimized via optimizer, and discriminator weights are updated. Note that the total discriminator loss calculation in the figure 5.7 is the sum. This sum of input/target and input/generated pairs is done in one step. Still, the loss is updated twice in my implementation, and the framework automatically calculates the sum.

Generator loss from equation (3.6) has two parts adversarial loss and L1 loss. They are summed together, but the L1 loss is multiplied by $\lambda = 100$. L1 loss is calculated from generated and target images. Adversarial loss is evaluated as follows: Input image and generated image are inputs for discriminator, but discriminator label is set to „real“ instead of „fake“. This way, the generator is led to generate more real fake images. In the end, the L1 loss and adversarial loss are summed, and the final loss is calculated. This process is shown in figure 5.8. The weights are then optimized via optimizer, and generator weights are updated.

The trick is when the generator is trained on the output of the discriminator, the generator gradients are calculated through the discriminator. And while the discriminator improves on its own, the generator is trained to beat the discriminator. Note, even though I say the discriminator trains on its own, it still uses the generator to obtain the generated image.

When the discriminator improves, the generator improves as well. If the discriminator is good enough, the generator will be capable of creating desired outputs.

5.3 Implementation details

Before some implementation details are shared, let me firstly introduce the used libraries. It will help the reader to better understand how I solved the problems.

To parse parameters the Click[1] python package is used. Tensorflow[2] was used as a machine learning platform for the implementation. Tensorflow Keras known as `tf.keras` is used as high-level API. Rasterio[3] in our thesis is used to read GeoTiff files with suffix `.tiff` or `.tif`. It provides

[1] https://palletsprojects.com/p/click/
[2] https://www.tensorflow.org/
[3] https://rasterio.readthedocs.io/en/latest/

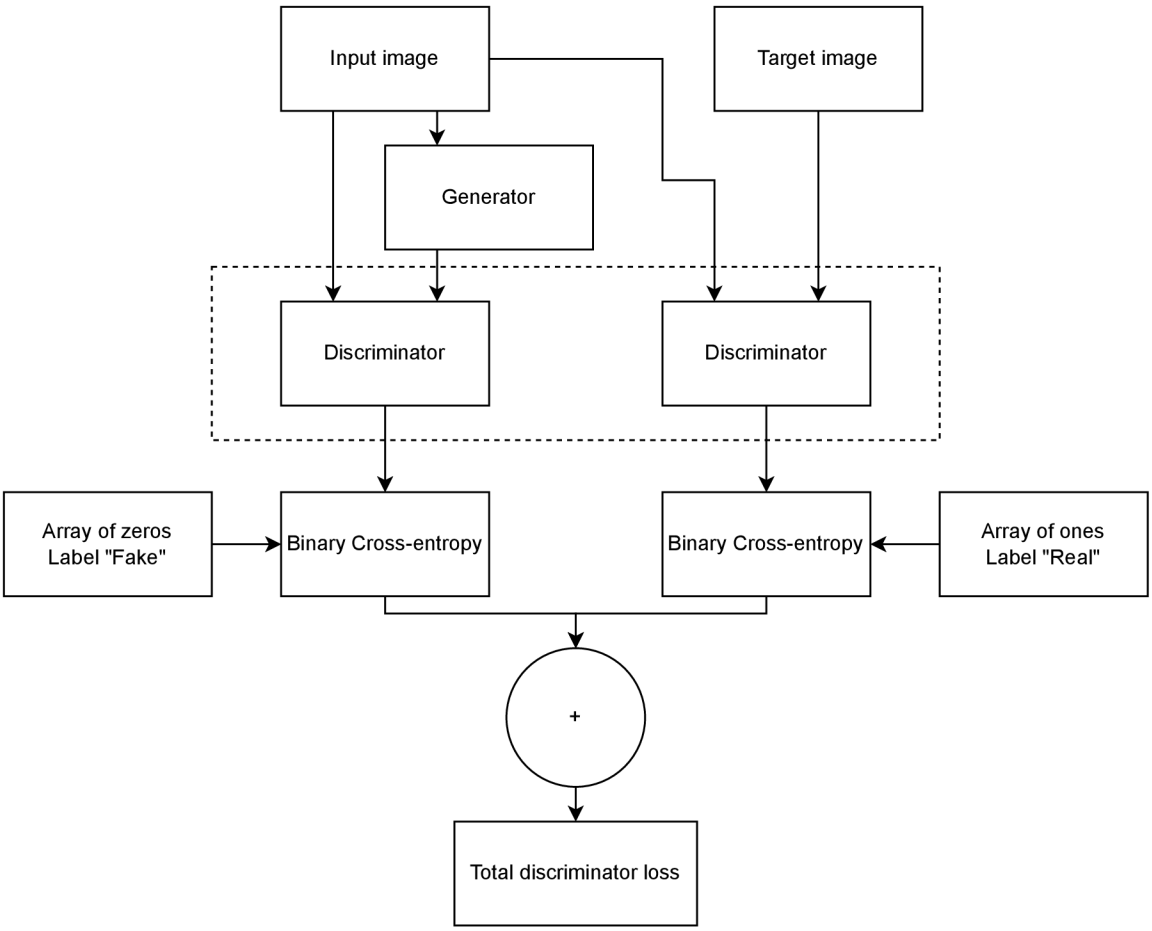

Figure 5.7: Discriminator loss calculation process. The dashed bracket signifies that there is only one discriminator, but is updated twice.

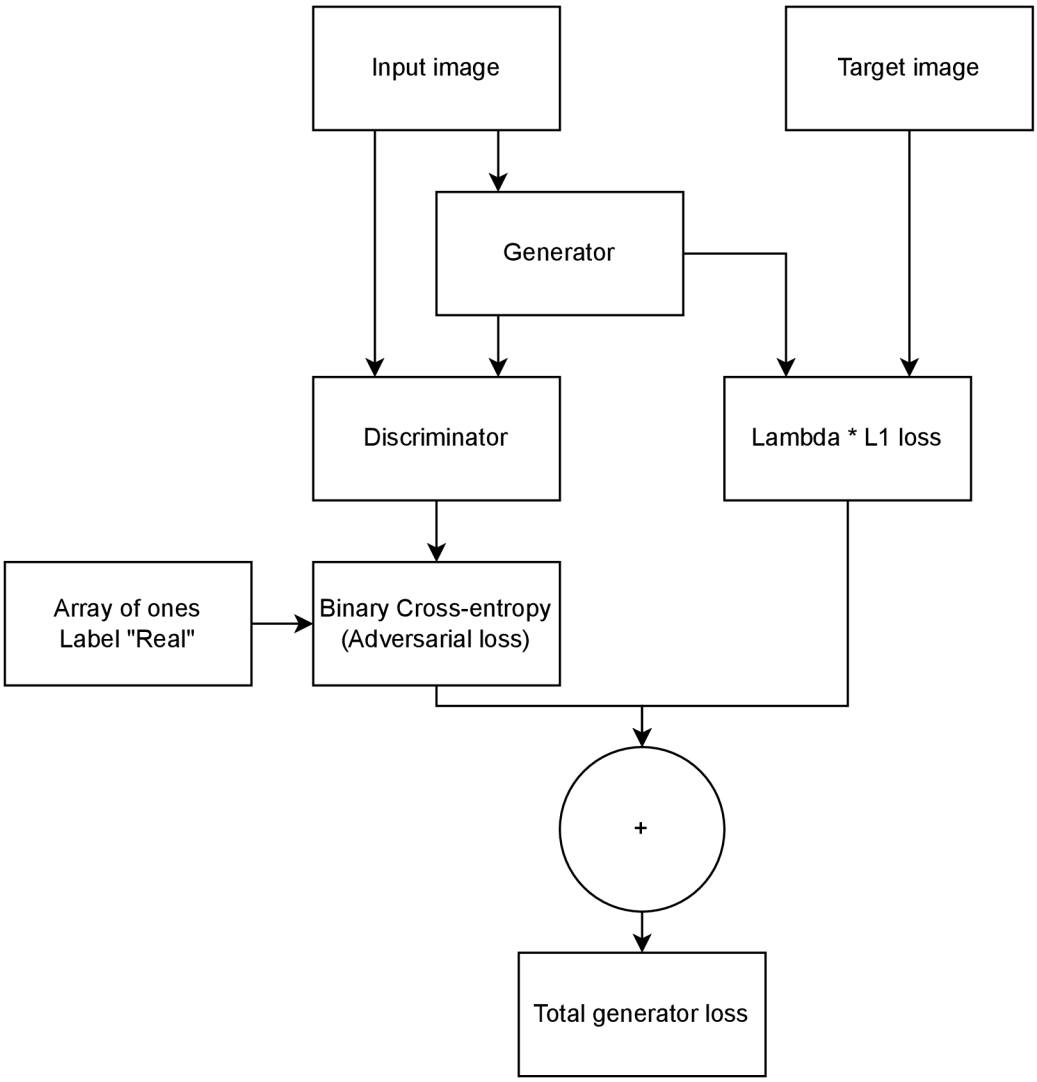

Figure 5.8: Generator loss calculation process.

the option to read a number of bands, the geographic location, georeferenced system and many more. With the picture loaded via Rasterio, the content of the picture is saved to NumPy[4] arrays. NumPy was used for numerical work in my project. For visualization of the results Matplotlib[5] was used. Even though it is not a library, the Open Source Geographic Information System – QGIS[6] is worth mentioning. I find it useful for manual work with tiff files to understand the data better even before the dataset was created.

The most interesting and important problem is choosing the training and evaluation data. The data can not be spitted just by the percentage 80/20 rule. It is important to train the network with one area and test with the other. When the pictures from the same area are used for the training, even though they might be from a different season, the testing will never be valid as the network already saw parts of the area. So, the data are grouped into categories, and once they are in categories, they could be split randomly and approximately into 80/20 groups for training/testing. The problem is: some areas/categories have large amounts of data, leading with Spain Bajadoz, which has 12 measurements, while other areas like Sweden or France have only 3. With such unbalanced groups there is no 80/20 balanced train/test dataset shown in table 5.1. Group split was implemented by `GroupShuffleSplit` from `sklearn.model_selection`. Sklearn, also known as Scikit-learn, is another used library. In my thesis, I did not use a validation set. The evaluation of generated images is done by the eye and by the analysis of generated results compared with ground truth.

Another problem I have encountered is that I often ran out of memory because I was trying to load a massive amount of data into the RAM. I solved this issue by defining `DataGenerator` class. It inherits from the `Sequence`, which runs each sample once per epoch. DataGenerator class returns a batch of images from the dataset, so once we are finished with the batch, another batch is called. It also allows multi-processing and multi-threading. In order to use multi-threading and to keep the order of the returned batches, the `OrderedEnquerer` is used.

```
# initialize the DataGenerator
data_gen = DataGenerator(data, version, batch_size=1)
# create an enquerer
enq = OrderedEnqueuer(data_gen)
# start the enquerer
enq.start(workers=4)
gen = enq.get()
# iterate over all epochs,
# to run over one epoch - divide dataset size by the batch size
    # call next batch
    batch = next(gen)
# et the end stop the enquerer
enq.stop()
```

The code example above is part of my implementation. The code uses four workers. I tried to run a simple version of my data generator to find the best number of workers. The four workers solution was the fastest one. The attentive reader sees that the batch size is one. Why? In the Pix2Pix paper [31], it is recommended to use such batch size for the scenario where input is

[4] https://numpy.org/
[5] https://matplotlib.org/
[6] https://qgis.org

an aerial photo and generated output is a map. I believe this example has a lot in common with ours, so I decided to keep the recommended batch size for our model as well.

With this example, we slightly touched on the training process. Let's continue with it in the next section.

5.4 Training

The network was trained on MetaCentrum[7]. Membership in MetaCentrum is free for students of academic institutions in the Czech Republic. When the user wants to train on GPUs, the preferred option is the Singularity container. The version of Singularity[8] container I used is `TensorFlow:21.09-tf2-py3.SIF`. The provided container uses Python 3.8.10 and TensorFlow 2.6.0. Occasionally the assigned machine could not find the container, and I had to re-run the job. The user can not control which machine is assigned to him, nor can he control when the job starts. There is a queue of jobs with priorities. The jobs are balanced by availability and priority. It also depends on how much time the job is already in the queue. All models were trained with the same hardware requirements:

- 24 GB RAM
- 2 GPU's with compute capability at least 7.5
- 8 CPU's

I trained all models for 200 epochs. The only exception is model 2, trained on the whole dataset, which was trained only for 100 epochs. The reason behind this is time efficiency. This complex model with nearly 140 million trainable parameters is difficult to train. It took eight days to train 100 epochs. The train time is shown in table 5.1.

Note that model 3 has the same amount of parameters, but it is trained faster. The reason is simple, fewer data inputs with a double revisit period. In comparison, model 1, with 200 epochs and around 100 million trainable parameters, was trained for nine days with the exact same dataset. Comparing the length of the training process is not that simple because the machines you get from the MetaCentrum are usually different from the previous one. Provided machines typically do not have the same GPUs and, therefore, different performance. More powerful can do much more work but are often in use. So in order to get machines more often, I did not specify the GPU capability for the best-performing ones. And, of course, GPUs are not the only factor in the performance, so I do not consider training length comparison a valuable resource of information.

The maximum allowance time on MetaCetrum for a GPU task is 24 hours. So every day, I manually re-ran the models to continue the learning. It was very inefficient. To guarantee that the network learning will continue where it finished the last day, I implemented the usage of the Checkpoint class in my solution. Checkpoint is part of the TensorFlow, and thanks to it, the training can run from the last point when the checkpoint was saved. The checkpoint in my implementation is saved after each epoch. Checkpoint also allows the developer to save global parameters inside the checkpoint object. I used such an opportunity to count the epoch number, not to overcome the magic number 200.

All models generator loss shown in graph 5.9 are smoothly decreasing with value but are far from the equilibrium of one half. They are separated into three pairs. The first pair of models

[7] https://metavo.metacentrum.cz/
[8] https://sylabs.io/

Model and dataset	training data	testing data	number of days to train
Model 0 - spring	7892	3296	5
Model 1 - spring	5001	1559	3
Model 1 - all	12312	8435	9
Model 2 - all	12312	8435	8*
Model 3 - all	5411	4782	5
Model 3 fixed - all	5411	4782	6

Table 5.1: Number of training and testing data for each trained model. *=100 epochs, otherwise 200. Based on the model inputs availability is checked.

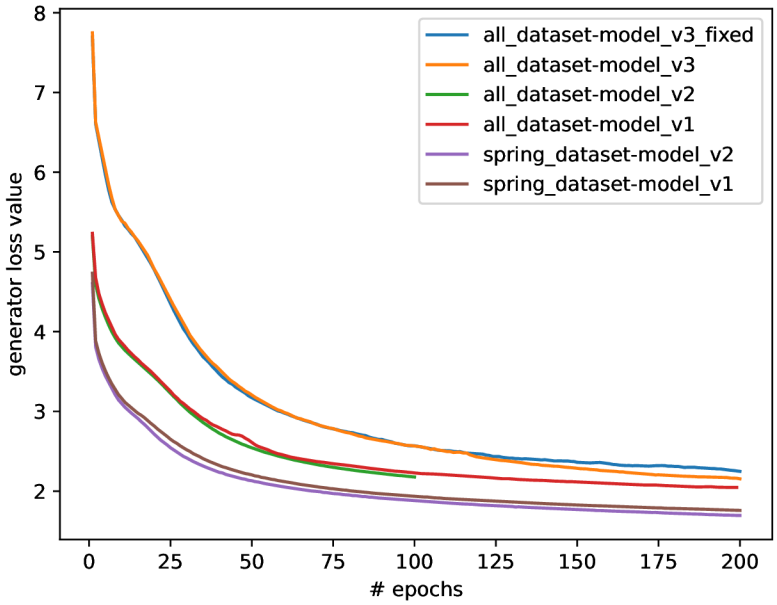

Figure 5.9: Generator loss value for all trained models. The loss value per epoch is the average of all losses in the epoch.

with the lowest training loss value is trained on the Spring dataset. The second pair, model 1 and model 2, is trained on the All dataset. During the training, there was no difference between model 1 and model 2. Lastly, model 3 and its fixed variant, with double revisit time. The fixed version of model 3 did not drop as efficiently as model 3. The discriminator loss graph 5.10 is more attractive due to spikes. It is important to understand that the spike range is negligible, but the 70x70 PatchGAN has the best performance. On the other hand, model 2 seems not to improve at all. Also, models trained on the Spring dataset from a discriminator point of view are not approaching the equilibrium. Better said, they do not improve at all.

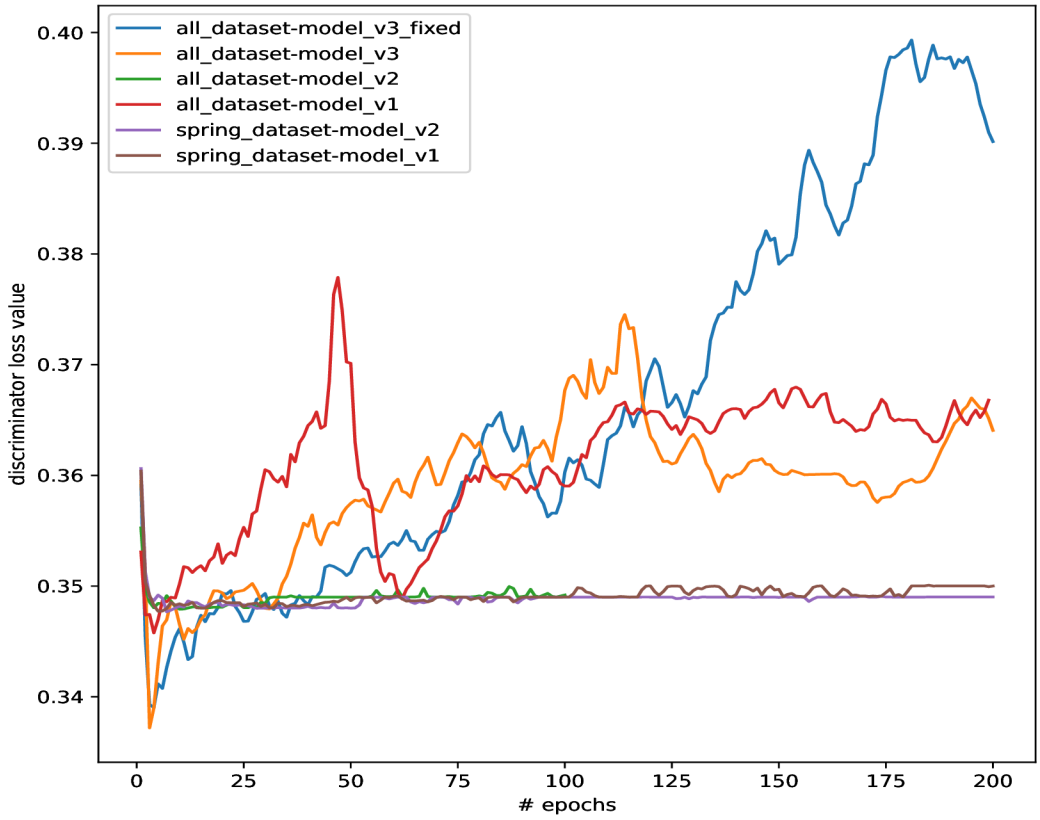

Figure 5.10: Discriminator loss value for all trained models. The loss value per epoch is the average of all losses in the epoch.

Model 0 was trained but was not added on purpose to graphs 5.9 and 5.10. Model 0 had values in different range than other models. To see more details in graphs I rather decided to show it separately. Graphs C.2 and C.1 are placed in appendix.

Chapter 6

Results

In this chapter, we will look at the trained results. Each model has the same template. Firstly, the reasoning behind model creation is explained. Then the results are discussed and compared with previous models. Lastly, additional details are described and complete comparison is made.

During the training, the generator model after each epoch is saved. If not mentioned otherwise, the results described in this chapter are trained for 200 epochs.

One of the GAN common problems is how to evaluate them. In this thesis, the pixel to pixel comparison was found useful even though this approach still has drawbacks. One of them is the difference between predicated NDVI values. Mean absolute error only considers the difference between predicted and real values. In real life, the difference between NDVI 0.8 and 0.9 is not as big as between 0.1 and 0.2. Therefore I also used relative difference as part of the evaluation. In the future, I would like to observe how the network reacts to the relative loss function.

The results are provided by a trained generator which can generate new NDVI pictures. Both the precision of the result and image appearance play a role in evaluating the best result.

6.1 Model 0.

Model 0. was created as a proof of concept to see whether the network works. The assumption was that model would not be able to learn the optical data due to speckle and other noise factors. We will observe the learned features.

As we know from the training section 5.4, model 0 was trained on the Spring dataset. In figure 6.1, we can see the results of model 0. The third row generated image is not looking the same as the fourth. There are parts of the pictures that we can assume are recognized correctly. For example, the last column is the right bottom part of the picture. The field was recognized, but the value differs. In this chapter's result pictures, the last two columns represent the generated and targeted image.

From bar plot graph 6.2, we can see the relative difference per pixel between generated and targeted images. Data are averaged over test data. The sum of pixels in the graph is hence $256 * 256 = 65536$. It has a lot of misleading pixels.

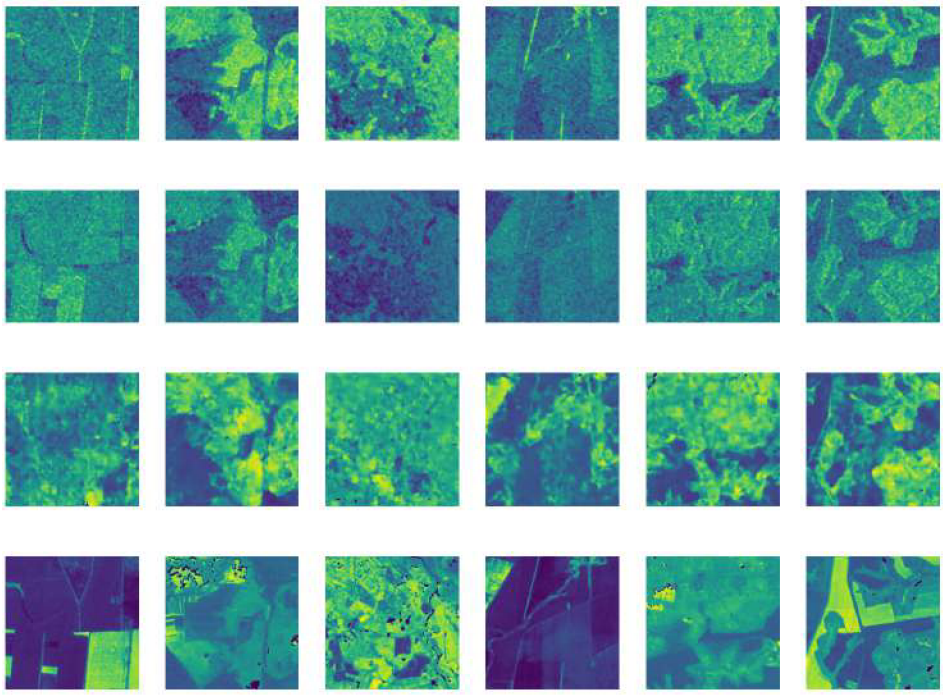

Figure 6.1: First six pictures generated on spring test data from model 0. Each column represents a picture. The rows from the top: are radar VH, radar VV, generated NDVI, and ground truth.

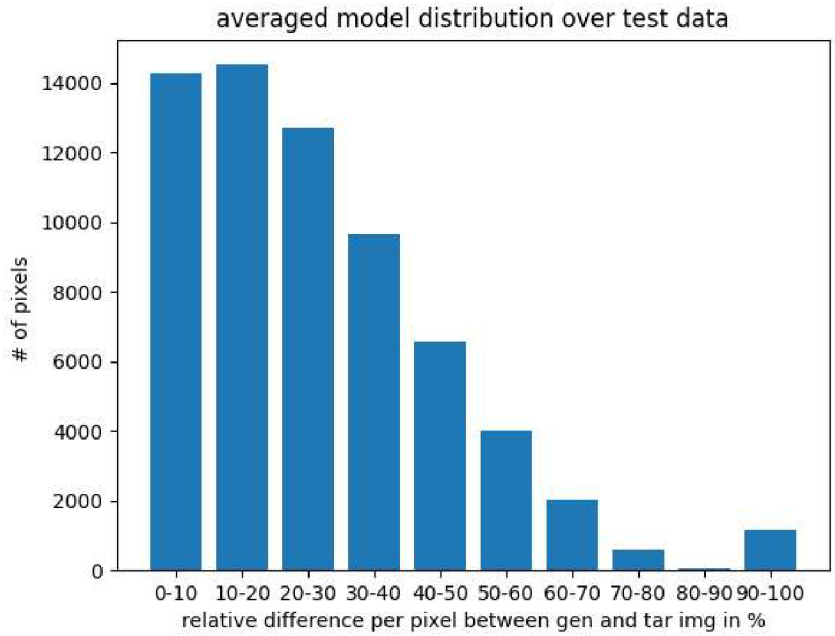

Figure 6.2: The relative difference between generated and targeted image of model 0. Calculated for each pixel over the whole spring test data. The sum is averaged over images and shown on the graph.

The numerical results - **average MAE: 0.1417** and **average of RD: 0.266**,
where RD is the relative difference between pixels, basically what is shown in 6.2, but average. Numbers are rounded to 4 decimal places. All other numerical results are rounded the same way.

We have a starting point, and now we know that the average MAE result of 0.14 is insufficient. Our assumption was correct. The model is unable to learn the results.

6.2 Model 1.

The optical data, from the last revisit, were added to the model to get better results. With our dataset, we have perfect conditions, and we always have the data from a week ago.

This model expects to create a sharp optical image as a network has the optical image from the past.

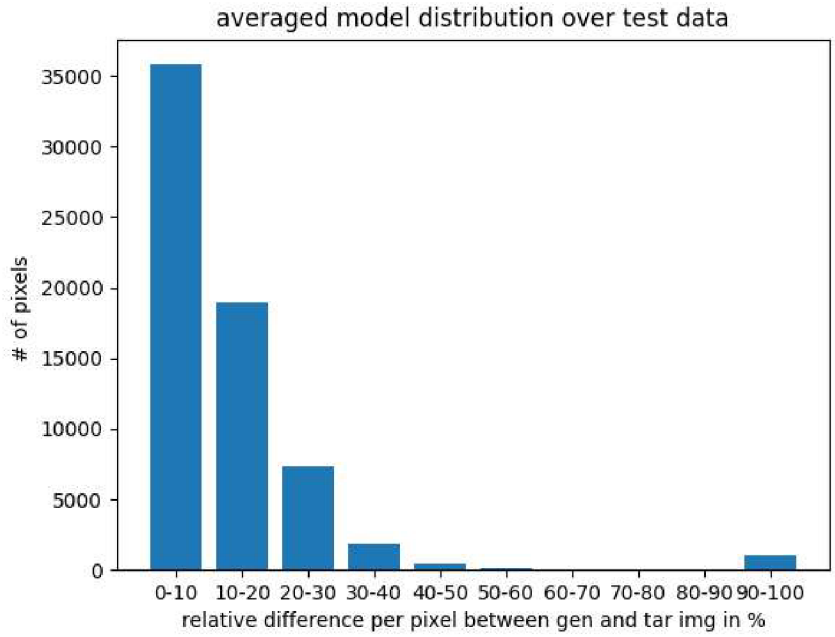

Figure 6.3: Relative difference between generated and targeted image of model 1.

The numerical results:
average MAE: 0.0466
average of RD: 0.1111

In graph 6.3 we can see huge improvement compared to 6.2. The last bin 90−100% difference is still in place, but that is noise. We will see such noise within all results across all models.

At a glance, model one results shown in figure 6.4 are almost indistinguishable. What worries me is that even the last revisit time, NDVI looks the same as the ground truth.

Another problem, if you look closer at the first and last columns and compare the generated image with ground truth, the first image is a bit brighter, and the last column is a bit darker. To recall, the Viridis spectrum with our images goes from zero – dark blue to one – yellow. From what I see, the change of shade occurs mainly for values closer to zero. I did create the image

multiple times, and other pictures were generated with the proper colour. The generator has some variance, but more about it can be found in experiments in the section 7.1.

I believe the attentive reader should have a question. If images shown in figures 6.1 and 6.4 are showing the first 6 images, why the images vary ? The hint here is the dataset size difference, shown in table 5.1. Consider that during dataset creation, I filtered images from the grid, shown in figure 4.4. In model one, the requirement for the *measurement ID* and *measurement ID bigger by one* is to have the same column and row. It is possible that one of the pictures was filtered out, and so was the pair.

The first models I trained, model zero and model one, were trained on the Spring dataset for time purposes as they have fewer data. There is a too big matrix of options: seasons, models, parameters, and loss functions. The best season to use is not any of them, but all of them. So, I decided that I will train only by using the All dataset from now on.

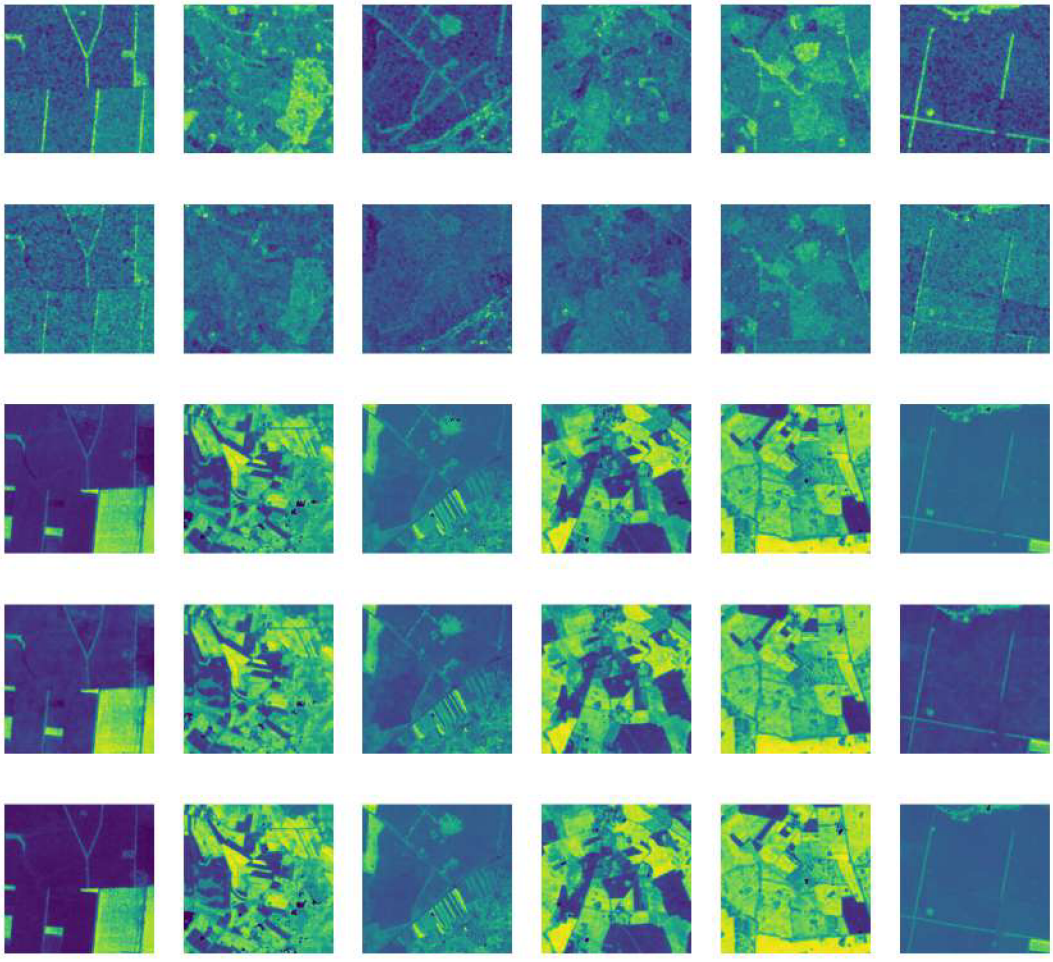

Figure 6.4: First six pictures generated on spring test data from model 1. Each column represents a picture. The rows from the top: are radar VH, radar VV, last revisit time NDVI, generated NDVI, and ground truth.

The expectation of model 1 was met, and the results seem very well. Next on, I want to compare the results with the **All dataset**.

An expectation: the results will slightly worsen because I believe the All dataset has a bigger variance compared to the Spring dataset.

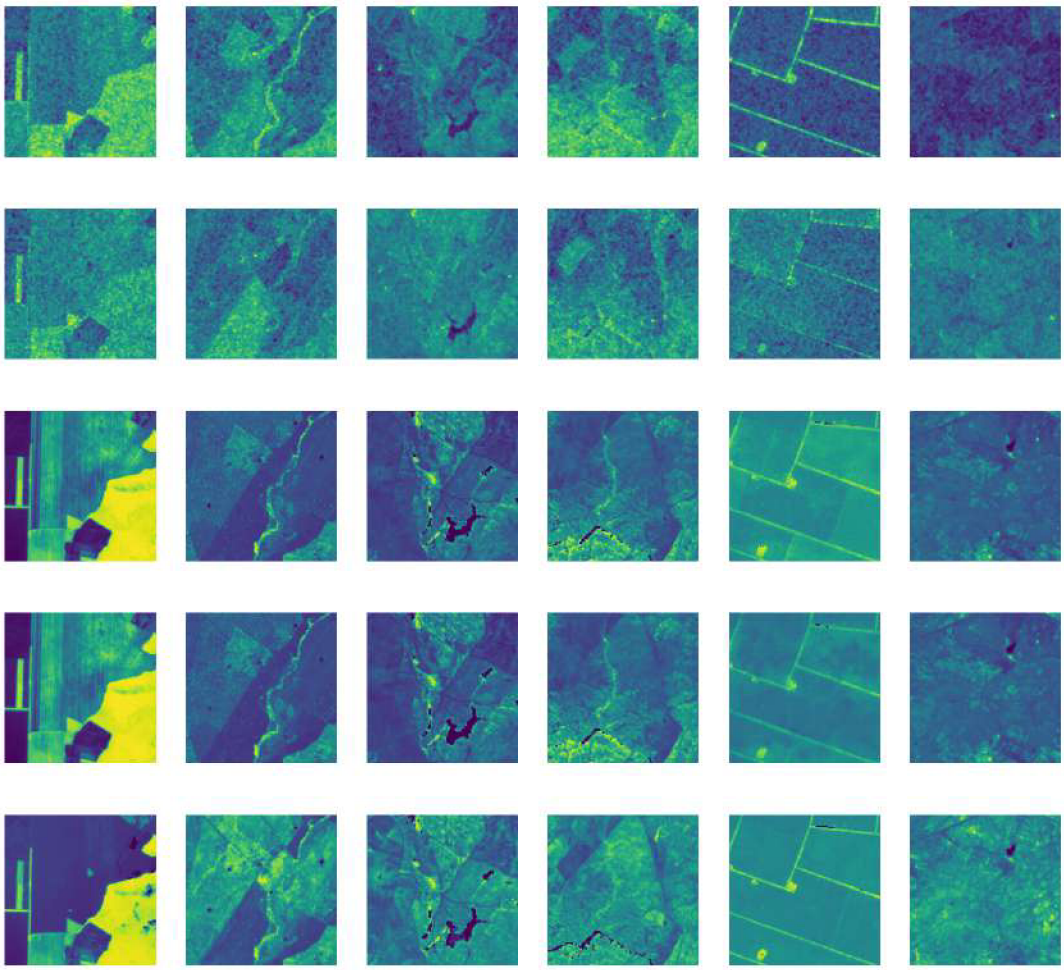

Figure 6.5: Worst 6 pictures generated by model 1 from test data of the All dataset. Each column represents the picture. The rows from the top: radar VH, radar VV, last revisit time NDVI, generated NDVI, ground truth.

The numerical results:
average MAE: 0.0454
average of RD: 0.1036

There is a change in figure 6.5, which displays result images. The results are now the worst images. Let me explain what is meant by the worst images. The relative difference between pixels is summed up. The same is done over all pictures and the whole test dataset. Results are sorted, and the images with the highest values are the worst ones. This simple way of calculation should provides the worst images.

Analyzing the last revisit time and ground truth of the first two columns is shown in figure 6.5.

1. filename – *BULGARIA_SUMEN_2017-07-31_2017-08-12-2-3-0.tiff*. It is a typical example of the harvest at the beginning of august when the crops are gathered. The NDVI value decreased as there were no longer crops.

2. filename – *SPAIN_BADAJOZ_2018-10-18_2018-10-30-1-14-0.tiff*. Opposite to the previous image, here, the NDVI value increased. The measurement ID is 1. It can be found in the table 4.1 and the exact dates are *2018-10-17* as the date before and *2018-10-24* as the actual S2 NDVI image. I took a look at the weather in Spain during this time period. In Spain, there are long summers, the temperature is high even the whole of September and often the October as well. But temperature began to decrease on 18 October and 19 October caused by 3mm and 4mm of rain respectively. The vegetation waking up after the rain and colder temperatures explain the increase that we can see. Of course, this is just my belief.

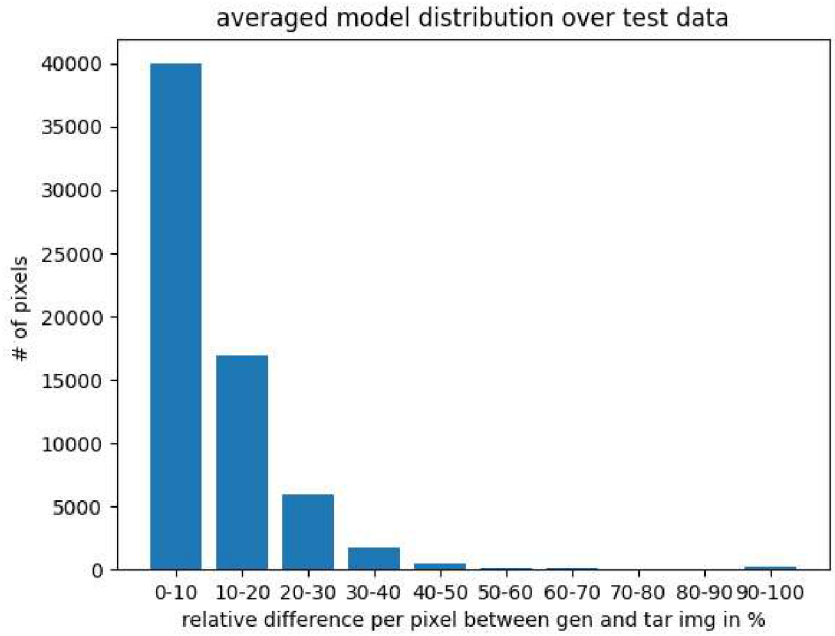

Figure 6.6: Relative difference between generated and targeted image of model 1.

In a perfect world, we would like to see these results be caught by the network. But I do not expect the network to learn it because there is not enough training data that differ so significantly as the examples we just saw. Another way to improve it, is to change the way how the network weights are updated. More about this is written in the experiments chapter, in the section 7.2.

The difference between model 1 bar graph 6.3 trained on the spring dataset and model 1 bar graph trained on the all dataset 6.6 shows that there are even more pixels in the first pillar, around 40 thousand, to the previous 35. It also shows that there is a variation because pillar with $60-70\%$ risen.

My expectation was refuted. From numerical results, we can see the results of model 1 trained on the All dataset are even better even though some new variance was added. The pre-

sumption is that with additional data, there is even more data that differ just a little. To find out whether my presumption is true, I compared the Spring and the All dataset in graph 6.7.

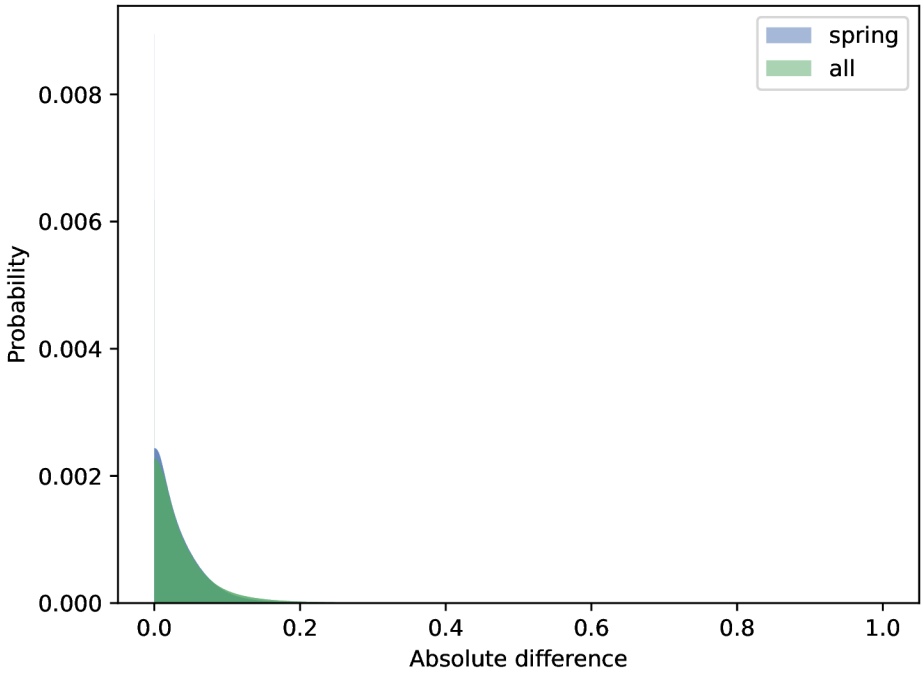

Figure 6.7: Data distribution shows the absolute difference between measurement data. Data distribution shows the Spring dataset and the All dataset. Revisit time period is one. Probabilities are so low because there is so many bins in the histogram.

The training and testing set is taken together, and the absolute difference between pixels is calculated for each pixel in the image across all pictures in the dataset. This time there is no sum or average. The enormous list of pixels is plotted as a histogram. The two histograms are plotted into one. Thanks to that, we can observe the distribution difference between spring data and all data. They are literally almost the same. An interesting detail about the data is that the maximum difference within NDVI value is around 0.2, occurring minimally.

My presumption was confirmed. With more data, the network had more attempts to generate the correct output. That is why the numerical results are better.

6.3 Model 2.

The purpose of model 2. is to compare it with model 1. We want to observe whether the network is using information about the difference between radar pictures, as such knowledge should help to improve the network. As mentioned in the previous section, from now on, models are trained only with the All dataset. It is not exactly the truth, I did train model 2 on the Spring dataset, but I decided not to elaborate on the results because we have the same comparison between model 1 and model 2 on the All dataset.

The bar graph did not add any significant difference. Bar graph C.3 was added to the appendix for comparison purposes. The worst generated pictures are shown in figure 6.8.

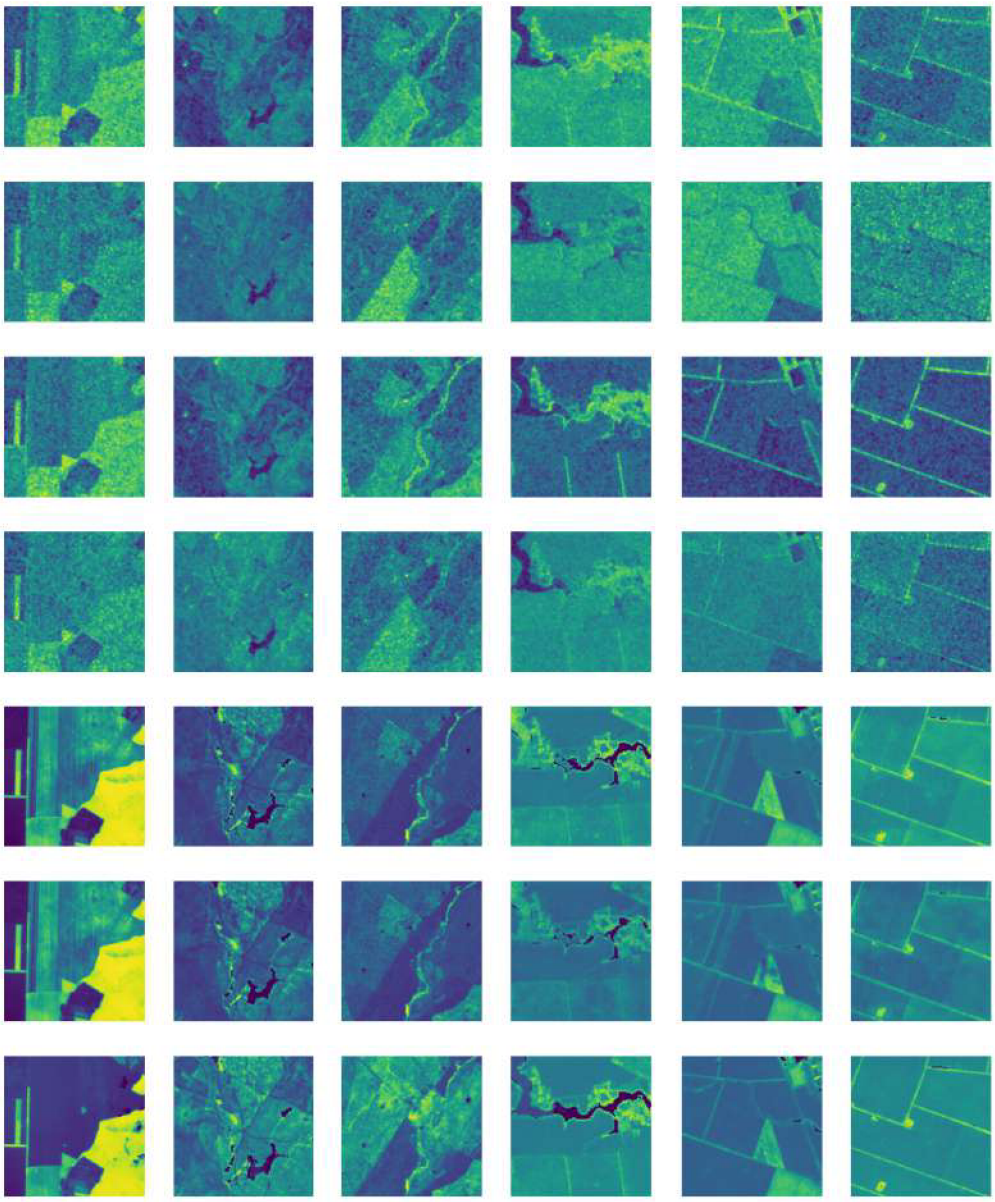

Figure 6.8: Worst 6 pictures generated by model 2 from test data of the All dataset. Each column represents a picture. The rows from the top: last revisit time radar VH, last revisit time radar VV, actual radar VH, actual radar VV, last revisit time NDVI, generated NDVI, and ground truth.

Some may advocate that I am going to compare results of the model 2, which was trained only for one hundred epochs, with model 1, which was trained for 200 epochs. To deny an accusation, I decided to evaluate model 1 with the generator model after 100 epochs. Results are shown in table 6.1. Now we can compare model 1 after 100 and 200 epochs. The numerical values are exactly the same. Model 1 was unnecessarily trained for another 100 epochs.

	average MAE	average of RD
Model 1	0.0454	**0.1036**
Model 2	0.0454	0.104

Table 6.1: The numerical result comparison. Both models are evaluated after 100 epochs.

In the next paragraphs, there is more about radar to NDVI correlation problems.

When the reader takes a look at figure 6.8 and the difference of radar pictures in the first column, the harvested field from a radar point of view, there is not enough difference between pictures. Or, look at the last column where the actual radar VH picture shows significantly lower values inside the fields than a previous VH. The difference is not interpreted correctly by the network. The network tries to do both, create a duplicate of the optical picture and apply some radar knowledge, and as a result, there are spots inside the fields. The network also does not know which radar data are connected to which date.

The correlation between optical and radar images is a challenging task for the network, mainly caused by speckle. But even with smoothened speckle, I doubt that network is capable of learning the correlation between radar and NDVI because there is a lot of factors that differ: structure, surface roughness, incidence angle, humidity and geographical location. Also, it depends on which Sentinel radar takes the picture, as one is using ascending and another descending orbit. Dataset would need sufficient representation for each of these factors, which it does not have. But firstly, I need to improve my understanding of radar to NDVI correlation and then I can start with improvements to my dataset and network.

The network did not understand the correlation between radar bands and the NDVI layer. Now we kind of mix the apples with pears. I believe transfer logic can be acquired with two separate models. One of which creates an optical image, and the other applies the radar difference to an optical image. In the future, I would continue with speckle removal, added date specification and separate transfer logic.

Model 2, compared to model 1, brings little to no value to results. The difference between old radar data and actual radar data was taken into consideration, but the effect it did is unsatisfying. It is correct that generated results are outweighed by the optical image, but the radar information should be transformed into the generated picture.

6.4 Model 3.

Considering the result from the previous section, it would be easier if I had trained model 3 from model 1. I did it with model 2 as I have planned in the model's proposal section.

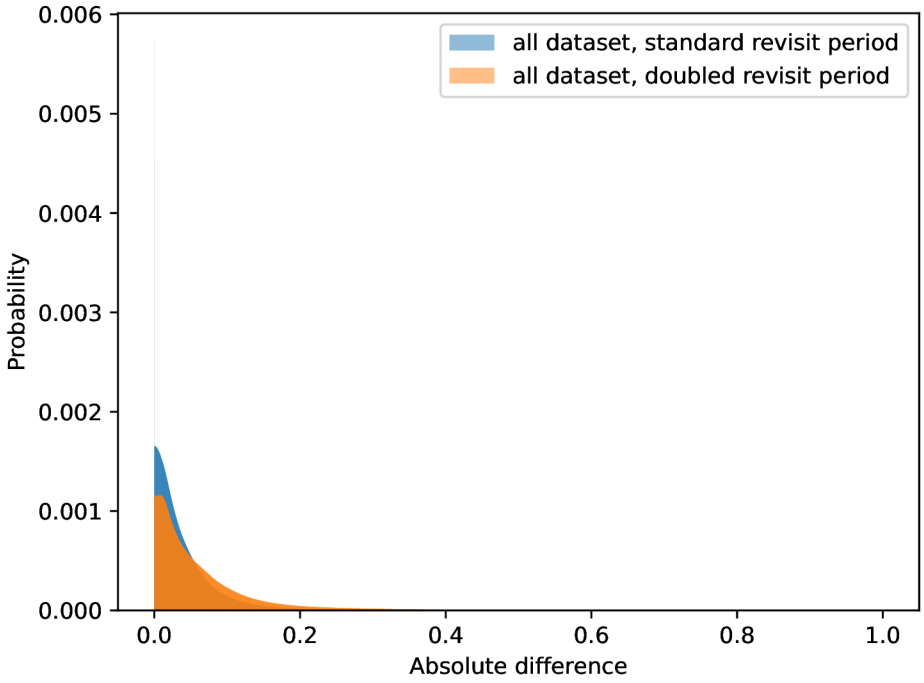

Figure 6.9: Data distribution showing the absolute difference between pixels, for each picture through dataset.

Model 3 is the same as model 2. The only difference compared to model 2 is that the revisit time period has doubled. We saw the worst examples of model 1 and model 2. I want to learn the network to catch vegetation changes after harvesting or when raindrops occur after a long dry period. The problem I see is not enough diversity within the data. With the time revisit period doubled, it should be better. Graph 6.9 confirms my hypothesis. The graph shows the All dataset data distribution. The range on the x-axis is broader for the doubled revisit period. The graph shows that the distribution looks like half of the normal distribution, with most results having an absolute difference near zero. I calculated the standard deviation for the data distribution. It is 0.0627 for the standard revisit period and 0.0881 if the revisit period is double.

This time I was unsure about my expectation. Has the dataset enough variety? With more variety, it is harder for the network to get everything correct. On the other hand, it will be trained to understand pictures with more variety, so the network should consider it.

The illustration of the results is shown in figure 6.10. Figure 6.11 shows the bar graph for model 3. The first bar has dropped, and others did proportionally grow. Numerical results are: **average MAE** – 0.0702 and **average of RD** – 0.1535. Results still look valid even though the nu-

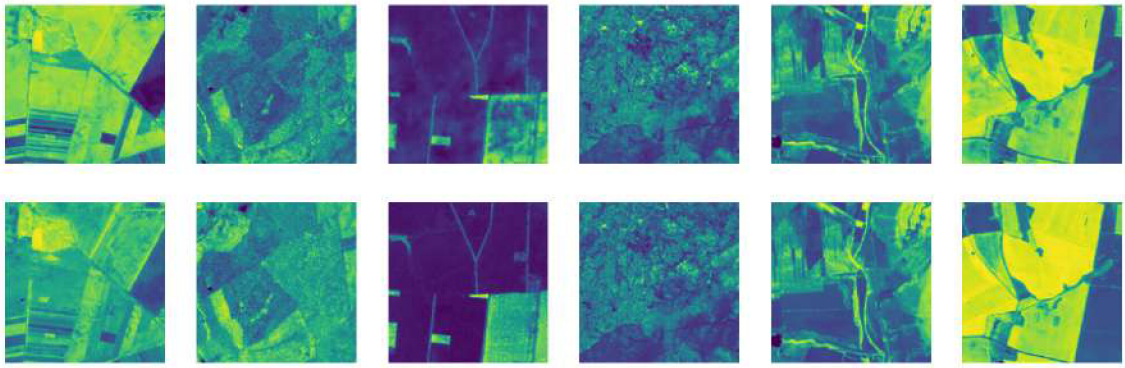

Figure 6.10: First 6 pictures generated by model 3 from test data of the All dataset. First row is generated image and the second is ground truth.

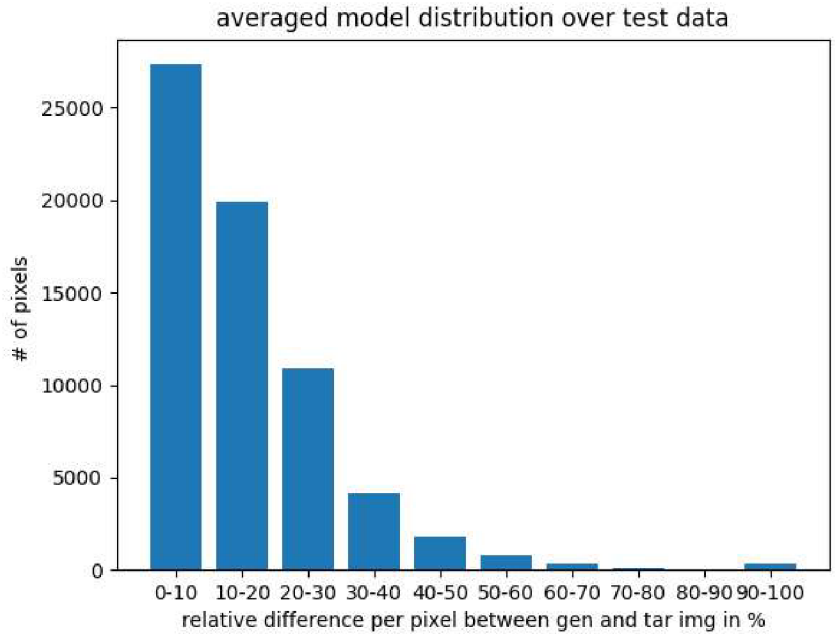

Figure 6.11: Relative difference between generated and targeted image of model 3.

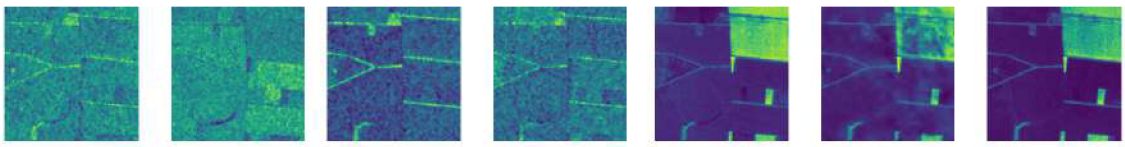

Figure 6.12: From the left: are past VH, past VV, actual VH, actual VV, past NDVI, generated NDVI and ground truth. The actual radar picture value has decreased in both bands. The network understands that the value has to be decreased, but the generated result is spotted. The main contrast can be seen inside the top right field, but the whole image has speckle areas with a slightly different shade.

merical values for MAE are now around 7% compared to 4% in model 2, although some results are not that great.

Figure 6.12 shows that model understands that value in top right corner has to be decreased, but does not understand by how much. Furthermore, it can not identify the correct applicable range, in this case, the whole field. As fields in radar pictures are speckled, it is hard for the network to realize that, in reality, the radar backscatter across the field should be fairly similar. I believe that if the speckle noise was smoothened, the network would provide better results.

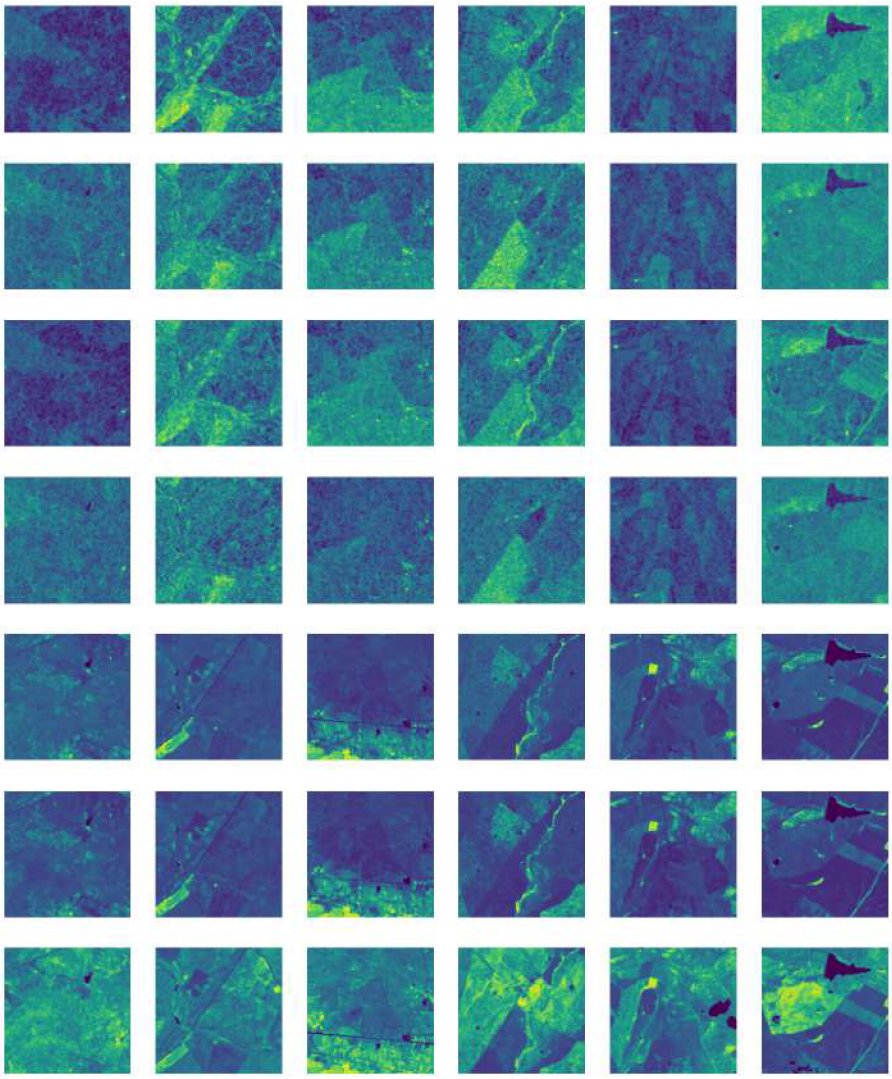

Figure 6.13: Worst 6 pictures generated by model 3 from test data of the All dataset. Each column represents the picture. The rows from the top: last revisit time radar VH, last revisit time radar VV, actual radar VH, actual radar VV, last revisit time NDVI, generated NDVI, and ground truth.

Worst results are shown in figure 6.13. Model 3 has the same problems as I found in model 2. The worst generated results do not reflect the change between optical images. But now, with bigger distribution, there is more mistaken images than in the previous model. However, how can we even ask to reflect the change if there is no difference between the past and the actual radar image? Such example is the first image in figure 6.13. Or radar information is saying the opposite

to NDVI? The example is the third image in figure 6.13. The past and actual radar comparison are where the past radar picture is brighter than the actual one. While between NDVI, change is the opposite. The actual picture is brighter than in the past. In the future, I would need help from a radar expert to answer the questions.

Model 3 did not improve its own ability to generate pictures with more variety. Let's look at whether model 3 with fixed architecture has achieved some improvement.

6.5 Model 3 fixed

A fixed model was created to compare the results with all previous models, but mainly to observe the difference between model 3. Model 3 fixed differences are the discriminator, using patch 70x70 and generator concatenation which is also applied to the last encoder level. I named this model fixed, as my expectation was an improvement in the outcome.

	average MAE	average of RD
Model 3	**0.0702**	0.1535
Model 3 fixed	0.0725	0.1583

Table 6.2: The numerical result comparison.

From the table 6.2, we might see that the values for the fixed model are slightly greater, which means the results are somewhat worse. The feature map of the last encoder layer in the generator is not providing any beneficial information, which would be reflected in the generator results.

There is no need to show additional results and graphs, as result differences are negligible. Curious readers can find them in the appendix: C.6 – bar graph and C.8 – figure with the worst 6 results. It would be interesting to experiment more with the patch size in the future.

6.6 Additional results

There was one more problem I realized. Bar graphs are not providing information about pixels directly. Was the expected value of pixel higher or lower? What are the occurrences of such pixels across the dataset? To answer my questions, I have created the Confusion matrix graph and re-evaluated the models.

To generate a matrix, I modified the already existing python module Pretty Confusion matrix[1]. The confusion matrix is calculated from the test dataset.

Model 1 shows the best results with almost 61% correctly assigned values. Model 1 and model 2 have similar results. Confusion matrices figures C.4 and C.5 for these models can be found in the appendix. Most realistic model 3, shown in figure 6.14, with 45%. The confusion matrix is calculated for ten categories. Of course, there will be a bigger chance of missing the diagonal of the confusion matrix with more categories. For proper analysis, more categories are required. But, it would be hard to read the graph from the paper with more categories. With ten categories, it is already hard to digest.

[1] https://github.com/wcipriano/pretty-print-confusion-matrix

Figure 6.14: Confusion matrix for model 3. Compares predicted pixels with present NDVI for test data of the All dataset. Provides more insights into pixel distribution. The axis shows a value range between 0 and 1 with steps equal to 0.1. At the edge, aggregated data are shown. Aggregated values show the percentage of how many pixels were recognized correctly and incorrectly per row/column. The black and red colour in the graph core represents correctly and incorrectly assigned values. Unlike the percentages on edge, the core of the graph shows overall percentages.

Confusion matrices across models have multiple common properties. The most used value is between 0.2 and 0.3. Other often-used ranges are from 0.1–0.2, 0.3–0.4 and surprisingly, 0.8–0.9. The most used values have the best prediction results. Another thing in common across models is that mistaken categories of pixels are disproportional. The network more often predicts the higher value than it is. Disproportion is most problematic for values ranging from 0.3 to 0.7 for actual data and from 0.3 to 0.8 for predicted values. With model 3, values in these ranges do not surpass the 40% mark. I did not consider the first and last categories on the diagonal, which have minimal occurrences across the dataset.

There are also some positives. The network is on a good path. Almost all error pixels are right next to the correct solution, thanks to MAE. The ideal solution is to strengthen learning in the middle range and sufficiently reward the model if the predicted value is in the proper category. The question to solve here is, what NDVI value ranges are the important ones from the agricultural point of view? Further study is required to answer the question.

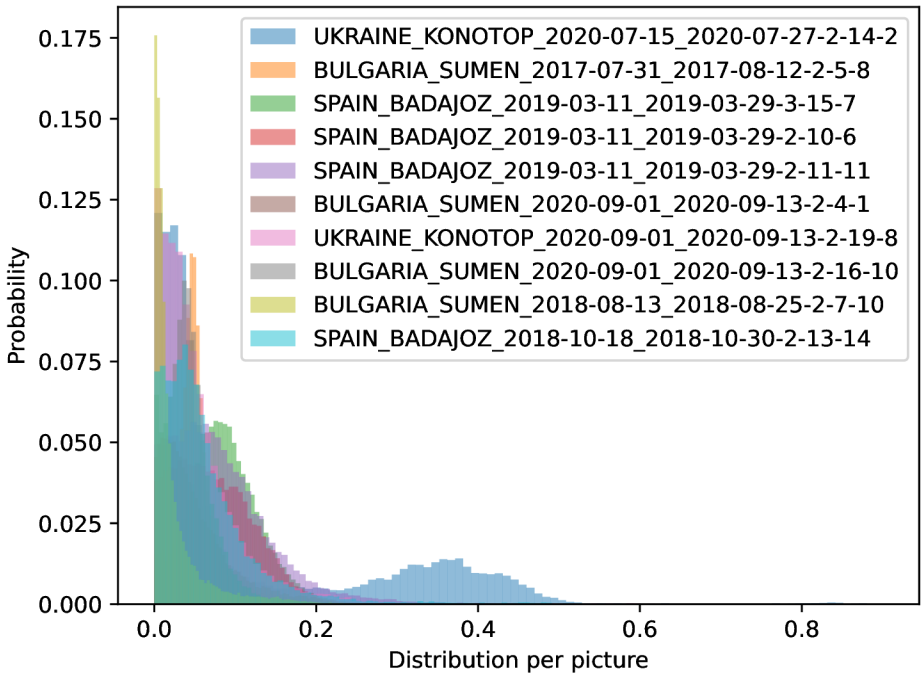

Figure 6.15: Model 3. generated 10 pictures from test dataset. Each picture shows distribution of pixels of absolute difference between the generated picture and the ground truth.

The main problem of the network I have mentioned in almost all result sections is results with more significant change. The network did not generate the desired output. We are trying to generate too complex images. Our dataset consists of more than just a green area. And also green area often did not change from week to week.

In figure 6.15, there are 10 random pictures from the test dataset. Most of them vary just a little, and I have no interest in those images. On the other hand, the *UKRAINE_KONOTOP* is the only picture which seems to have almost 2 distributions. While the main part of the distribution

is still under the 0.2 mark, there is an increase in distribution in an interval between 0.2 and approximately 0.5. To look closer at what is the cause, let's look at figure 6.16. Two factors play the role – invalid pixels on the top left side of the past NDVI and harvested field. Generally, there might be more factors, and it is crucial to recognize the difference between them. Here, the invalid pixels in past NDVI have a value close to zero, and the harvested field in the ground truth picture has a similar value for all pixels in the area.

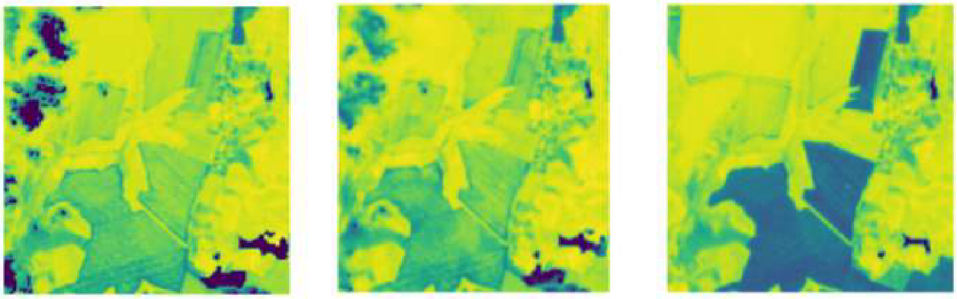

Figure 6.16: An example of an insufficiently well-generated image. From the left: past NDVI, generated picture, ground truth. Generated by model 3.

Why all these details about the picture? If we can recognize the factors, we can mask only the areas we are interested in. Basically, instead of two distributions, we would train the network to understand just one, and that is a much simpler task. And I just showed on *UKRAINE_KONOTOP* example how the factor recognition process could be achieved.

	dataset	train/test data	average MAE	average of RD
Model 0	spring	7892/3296	0.1417	0.266
Model 1	spring	5001/1559	0.0466	0.111
Model 1	all	12312/8435	0.0454	0.1036
Model 2	all	12312/8435	0.0454	0.104
Model 3	all	5411/4782	0.0702	0.1535
Model 3 fixed	all	5411/4782	0.0725	0.1583

Table 6.3: Results summary.

To sum up, numerical results for models are shown in 6.3. The best numerical results were achieved with model 1. Additional radar data in model 2 did not bring the desired result. Model 3 may be considered the most versatile model from what I have trained. Unfortunately, versatility worsens the quality of the results. The data augmentation can help a bit, but a new, more diverse dataset would be better. I would still like to encourage readers to use model 3 because the real world is an unpredictable place with many unexpected factors. The main improvement in the future would be the use of a mask. Most of the time, most of the pictures do not change. Mask would identify the part of the picture that has changed, and the network would be trained on such pictures. The network would be trained on the changed areas, and it would also be great at recognizing them. Another main improvement would be divided logic, where one part recognizes changes in radar picture and the second part generates the predicted optical image, and in the end, they will be merged. Multiple models would be required for this approach.

Chapter 7

Experiments

In the following two sections, two experiments are described. The first one measures how much the GAN prediction deviates when the same picture is measured multiple times. The second experiment implements a simple change to MAE loss.

7.1 Experiment 1 – deviation

How much will the same picture differ if I run it on the same model? This experiment will provide the answer to this question.

The experiment ran on model 3. It ran 200 times, always with the same 100 images. Images were chosen randomly from the testing dataset. The following steps were done for each picture and for each iteration to get the numerical results.

1. Pictures are compared with ground truth. The outcome is an absolute difference.
2. Mean of absolute difference.

With the results, we calculated the standard deviation for each image. The maximum and minimum deviation from the images is saved. In the end, the mean across standard deviations is calculated. The same process was done with generated pictures but without comparison to ground truth. Numerical results are shown in table 7.1.

	Maximum σ	**Minimum σ**	**Averaged σ**
Generated images	1.16E-003	1.43E-004	3.07E-004
Gen. Images compared with ground truth	5.51E-004	2.19E-005	1.94E-004

Table 7.1: Standard deviation across 100 images and 200 iterations. Maximum and minimum from all images, whereas average is the mean of all standard deviations.

Figure 7.1 shows the deviation between 11 copies of the same image. It is not a proper standard deviation, but it visualizes and represents the idea behind the deviation value.

Compared images have a lower value because the average difference is generally lower than generated images. As we learn in section 4.3, the NDVI values in our dataset are between zero and one. The maximum standard deviation from one hundred generated images, ran two hundred times, is around **0.0012**, but averaging **0.0003**. If the generator generates normal distri-

bution, the generated results will primarily differ by 3σ, which is less than **0.001**. With such low NDVI deviation in images, there is no hesitance that the network will always create a pretty similar image.

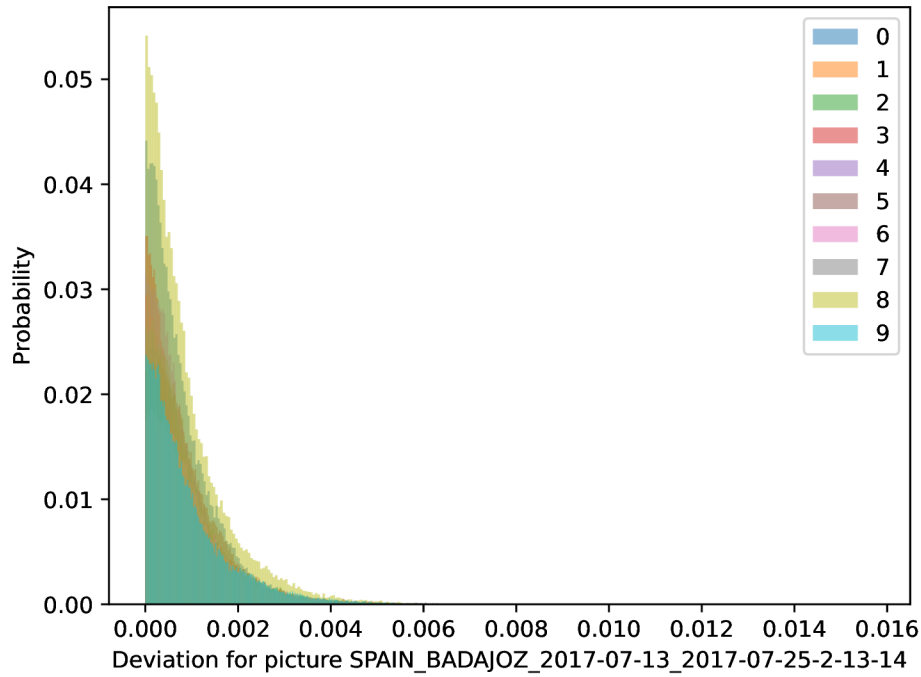

Figure 7.1: The same picture is generated 11 times. The first picture is taken as a subtract. Each other picture is subtracted with the subtract. Figure shows the absolute difference of the subtraction.

7.2 Experiment 2 – custom loss

The experiment changes MAE, which is one part of the generator loss function. I did not change the whole loss function, only the loss of MAE. The idea of this experiment is based on All dataset data distribution. Most of the data is too similar, and I would like to get better results on the data with more significant changes, even though that might degrade overall results.

I used normal distribution for the All dataset because they are very similar. Only absolute values were taken into consideration. Figure 7.2 shows an absolute normal distribution with standard deviation on the All dataset. The training is strengthened, multiplied by 1.25 on the interval between σ and 3σ. Training is weakened, multiplied by 0.75, for the data with a lower value than σ. Data with a bigger value than 4σ are multiplied by 0.5. During the training, the results of the generator seem promising. Graph 7.3, shows slightly lower generator loss value for model 3 with custom-loss. The discriminator loss graph stabilized sooner, and it can be found in the appendix, in figure C.7.

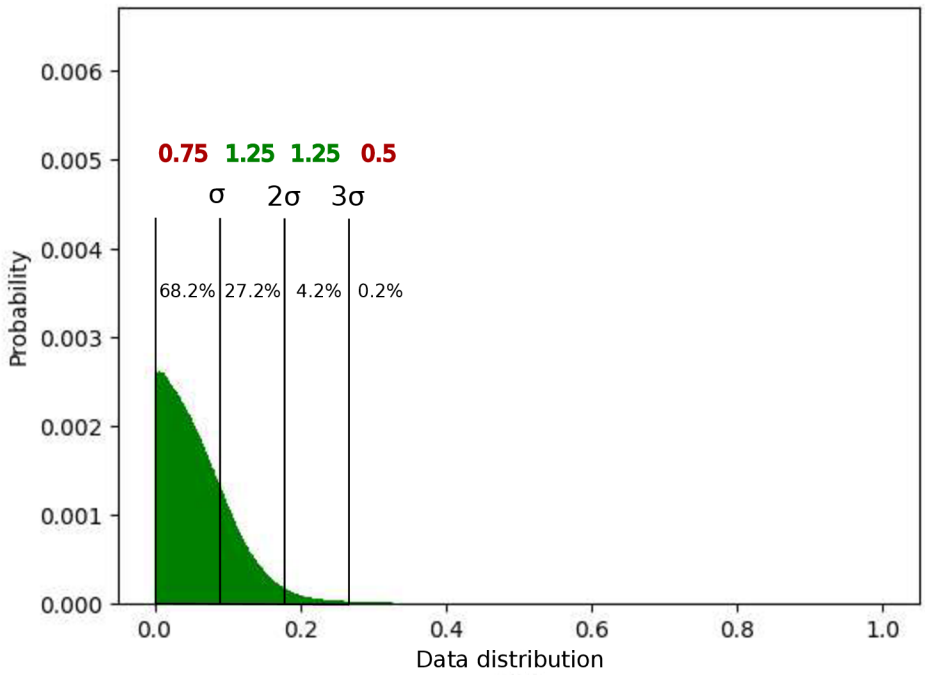

Figure 7.2: Data distribution from the All dataset used for model 3. Standard deviation $\sigma = 0.0881$. Distribution is similar to absolute normal distribution. Using standard deviation to divide the distribution into brackets. Each data value based on the bracket is multiplied by the number under the brackets to worsen or strengthen the importance of the data.

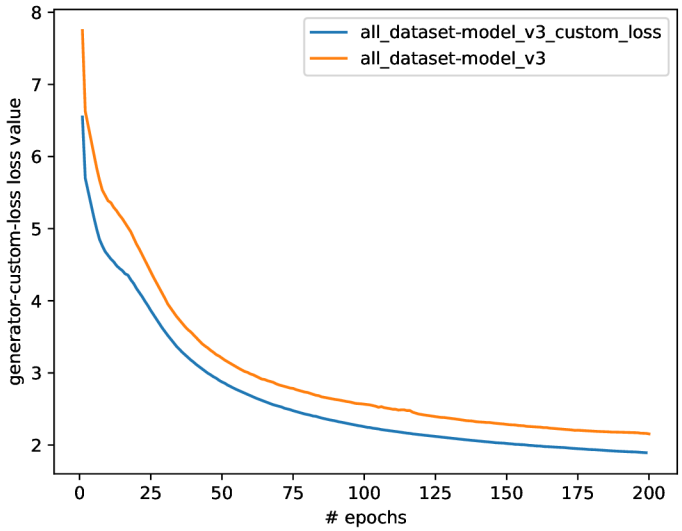

Figure 7.3: Generator loss value during the training. Compares model 3 with model 3 using the custom MAE loss.

Numerical results for model 3 with custom loss are **0.0702** for average MAE and **0.1538** for RD. Results are pretty similar to model 3, (0.702 for MAE and 0.1535 for RD). Unfortunately, I can not see any marginal difference in the picture results. Confusion matrix figure 7.4 represents model 3 with custom loss. It shows the aggregated percentages differ only in percent units compared to original model 3. For a better visibility, I rewrote the data to the table 7.2. The table shows that custom loss improved the results in the interval from 0 to 0.6. Based on the results, I presume that values within this interval are more often exposed to a greater change than a standard deviation.

	0.0 – 0.1	0.1 – 0.2	0.2 – 0.3	0.3 – 0.4	0.4 – 0.5	0.5 – 0.6	0.6 – 0.7	0.7 – 0.8	0.8 – 0.9	0.9 – 1
Model 3.	26.24%	58.52%	55.28%	37.65%	32.47%	30.11%	38.89%	51.16%	69.14%	16.87%
Model 3. custom loss	26.26%	59.54%	53.54%	38.39%	32.92%	31.61%	38.15%	49.29%	66.77%	16.47%

Table 7.2: Comparison of how many actual results were predicted correctly between model 3 with custom loss and model 3. A column represents intervals. The actual and predicted value lies in the interval. Data were taken from confusion matrices.

Although the custom loss seems like a step forward to me, the results are questionable. The presented ideas did not prove enough to draw conclusions from them. Lambda parameter might be the cause of the small changes. In the future, the tuning of the data distribution multiplying values together with lambda would be required. Even though worsening the overall results to achieve better outcomes for pictures with more significant change might seem an incorrect proceeding, it might help with the design and tuning of the future model trained on the dataset with a similar distribution.

Before I move to the conclusion, I would like to mention that once one of the models provided the results, I met with people from World from Space and with my supervisor and we discussed the results. The more we discussed, the more ideas appeared. We always decided to stick with the proposal. To add more inputs to the model and to experiment with it. In retrospect, it would be better to branch the focus on multiple different ideas instead of concentrating on one solution.

World from Space company found the result interesting but not satisfying. Their objective is to create a general solution that would be able to predict actual optical pictures across Europe. Basically, they need to solve the issues I have mentioned in the thesis and many more as their case is more general. They were not satisfied with the results of model 2 respectively model 3, as each of them did not understand how to apply the radar change to the optical image. Their belief is that with more cost function experimentation, the desired output is achievable. They also missed that the network has no idea about the time sequence. With the time sequence, we could have added multiple measurements in a row to achieve a better generated actual image.

The World from Space continues to explore the deep neural network approach in remote sensing for agricultural use. With their ongoing development in this field, where my solution and results are used as a base, there is a solid belief that generated images would be used in practice.

Figure 7.4: Confusion matrix for model 3 with custom MAE loss. Compares predicted pixels with present NDVI.

Chapter 8

Conclusion

The main goal of this thesis was to generate an actual NDVI picture, as closest to reality, as possible. To achieve that, I used past NDVI images with past and actual radar images.

In the beginning, I researched the basics of remote sensing. Multispectral images, vegetation indices, and many parameters affecting radar imagery have enriched my knowledge, which I used in data processing and evaluation. After becoming familiar with the GANs, I chose the Pix2Pix network that uses image-to-image translation to resolve the primary goal. The proposal was to incrementally add the number of inputs to the generator consisting of different amounts and types of past data and then observe and compare the results. I created four datasets, each for the first three seasons of the year and the fourth that connects them all. But, during the training, I used only the All and the Spring dataset. During the work, I thought I found a mistake in my architecture of Pix2Pix, so I created the last model with fixed architecture. I was happy to discover that I was mistaken, and the fixed model provided a worse result. The best results were provided by model 1, using past NDVI and actual radar, trained on the All dataset with a standard revisit period. Although, the recommended was model 3, using past radar, NDVI and actual radar. The recommendation is based on the doubled revisit time period, which, from trained models, is closest to conditions in reality. During the experiments, I found that generator deviation was unexpectedly small and that the custom loss experiment moved slightly with the distribution, but there was no visual improvement.

I created multiple models that can generate images unrecognizable from the original. The network can predict the actual NDVI from the past data, where most of the pictures in the dataset develop the same way. Most of the time, from around week to week, or within two weeks, the plant growth under mediocre conditions is similar to its predecessor. There are a few instances in the dataset when this is not true. The network, in these cases, needs to generate a picture that should use most of the radar data difference. Instead, the network uses the most known measure that it knows, and it will create an almost identical picture to the previous NDVI picture. Luckily, there is a grain of hope. I proved that some output pictures show signs of radar information. In other words, information is passed via the network and can be improved.

The real challenge of this thesis, to create a solution for real-life use, still lies ahead. Fortunately, I have plenty of ideas for improvement that did not make it to this thesis. In the short term, I would implement data augmentation and GAN comparison with simple methods, like linear regression. Next, I would try to remove the speckle from the radar pictures. It should remove the spots from the images. Then I would create separate networks for the radar and NDVI pictures and fuse them at the end. Additional information about measurement order propagated to the network would also be beneficial.

In the longer term, there is more to achieve. Further study is needed to understand and interpret radar data better. I would seek a better understanding between radar and optical image correlation. Next, a more diverse dataset is required to obtain a proportional part of the data with a big difference between the measurements, like a harvested field. I would not look at the whole picture, where most area details do not change most of the time. A real icebreaker can be mask usage to identify segments. I believe such a network would provide more valuable pictures for real-life use.

There is plenty of research where generative adversarial networks are used for radar to optical image translation and vice versa. What differentiates this thesis from the others is a unique dataset. Most of the research papers rely on a few locations or specific crops. From this point of view, this thesis is unique as well. As far as I am aware, the singularity of this thesis also makes its result the best one. However, it is hard to evaluate the quality of the results objectively.

Bibliography

[1] *About Copernicus.* [Online; visited 18.03.2022]. Available at: https://www.copernicus.eu/en/about-copernicus.

[2] *Fundamentals of Remote Sensing.* [Online; visited 13.03.2022]. Available at: https://www.nrcan.gc.ca/maps-tools-and-publications/satellite-imagery-and-air-photos/tutorial-fundamentals-remote-sensing/9309.

[3] *Generative Models and Adversarial Training (D3L4 2017 UPC Deep Learning for Computer Vision)* . [Online; visited 21.03.2022]. Available at: https://www.slideshare.net/xavigiro/generative-models-and-adversarial-training-d3l4-2017-upc-deep-learning-for-computer-vision.

[4] *Normalized Difference Vegetation Index.* [Online; visited 19.03.2022]. Available at: https://earthobservatory.nasa.gov/features/MeasuringVegetation/measuring_vegetation_2.php.

[5] *Optical Remote Sensing.* [Online; visited 19.03.2022]. Available at: https://crisp.nus.edu.sg/~research/tutorial/optical.htm.

[6] *Radio Detection and Ranging (RADAR).* [Online; visited 02.03.2022]. Available at: https://rscc.umn.edu/lessons/datatypes/radar.

[7] *Satellite frequency bands.* [Online; visited 17.03.2022]. Available at: https://www.esa.int/Applications/Telecommunications_Integrated_Applications/Satellite_frequency_bands.

[8] *Sentinel-1.* [Online; visited 18.03.2022]. Available at: https://sentinel.esa.int/web/sentinel/missions/sentinel-1.

[9] *Sentinel-1 SAR User Guide.* [Online; visited 18.03.2022]. Available at: https://sentinels.copernicus.eu/web/sentinel/user-guides/sentinel-1-sar.

[10] *Sentinel-2.* [Online; visited 19.03.2022]. Available at: https://sentinels.copernicus.eu/web/sentinel/missions/sentinel-2.

[11] *Sentinel-2 MSI Technical Guide.* [Online; visited 19.03.2022]. Available at: https://sentinels.copernicus.eu/web/sentinel/technical-guides/sentinel-2-msi.

[12] *What is Synthetic Aperture Radar?* [Online; visited 17.03.2022]. Available at: https://earthdata.nasa.gov/learn/backgrounders/what-is-sar.

[13] ARJOVSKY, M. and BOTTOU, L. Towards principled methods for training generative adversarial networks. *ArXiv preprint arXiv:1701.04862.* 2017.

[14] ARJOVSKY, M., CHINTALA, S. and BOTTOU, L. Wasserstein generative adversarial networks. In: PMLR. *International conference on machine learning*. 2017, p. 214–223.

[15] ATTEMA, E., BARGELLINI, P., EDWARDS, P., LEVRINI, G., LOKAS, S. et al. Sentinel-1 - the radar mission for GMES operational land and sea services. *ESA Bulletin*. august 2007, vol. 131, p. 10–17.

[16] BISHOP, C. M. and NASRABADI, N. M. *Pattern recognition and machine learning*. Springer, 2006.

[17] BROWNLEE, J. *How to Implement Pix2Pix GAN Models From Scratch With Keras*. [Online; visited 15.04.2022]. Available at: https://machinelearningmastery.com/how-to-implement-pix2pix-gan-models-from-scratch-with-keras/.

[18] BURT, P. J. and ADELSON, E. H. The Laplacian pyramid as a compact image code. In: *Readings in computer vision*. Elsevier, 1987, p. 671–679.

[19] CAMPBELL, J. B. and WYNNE, R. H. *Introduction to remote sensing*. Guilford Press, 2011.

[20] CHENG, K., TAHIR, R., ERIC, L. and LI, M. An analysis of generative adversarial networks and variants for image synthesis on MNIST dataset. *Multimedia Tools and Applications*. may 2020, vol. 79. DOI: 10.1007/s11042-019-08600-2.

[21] CHENG, Q., SHEN, H., ZHANG, L. and PENG, Z. Missing information reconstruction for single remote sensing images using structure-preserving global optimization. *IEEE Signal Processing Letters*. IEEE. 2017, vol. 24, no. 8, p. 1163–1167.

[22] CHOLLET, F. *Deep learning with Python*. Shelter Island, NY: Manning Publications Co, 2018. ISBN 9781617294433.

[23] DENTON, E. L., CHINTALA, S., FERGUS, R. et al. Deep generative image models using a laplacian pyramid of adversarial networks. *Advances in neural information processing systems*. 2015, vol. 28.

[24] ELDARDIRY, H. *The Use of Multi-Sensor Quantitative Precipitation Estimates for Deriving Extreme Precipitation Frequencies with Application in Louisiana*. Dissertation.

[25] EMMERIK, T. van. *Water stress detection using radar*. Dissertation.

[26] GATYS, L. A., ECKER, A. S. and BETHGE, M. Image style transfer using convolutional neural networks. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, p. 2414–2423.

[27] GOODFELLOW, I., BENGIO, Y. and COURVILLE, A. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[28] GOODFELLOW, I. J. NIPS 2016 Tutorial: Generative Adversarial Networks. *CoRR*. 2017, abs/1701.00160. Available at: http://arxiv.org/abs/1701.00160.

[29] GOODFELLOW, I. J., POUGET ABADIE, J., MIRZA, M., XU, B., WARDE FARLEY, D. et al. *Generative Adversarial Networks*. 2014.

[30] HUANG, S., TANG, L., HUPY, J. P., WANG, Y. and SHAO, G. A commentary review on the use of normalized difference vegetation index (NDVI) in the era of popular remote sensing. *Journal of Forestry Research*. Springer. 2021, vol. 32, p. 1–6.

[31] ISOLA, P., ZHU, J., ZHOU, T. and EFROS, A. A. Image-to-Image Translation with Conditional Adversarial Networks. *CoRR*. 2016, abs/1611.07004. Available at: http://arxiv.org/abs/1611.07004.

[32] JENSEN, J. *Remote sensing of the environment : an earth resource perspective*. Harlow, Essex: Pearson, 2014. ISBN 978-1-292-02170-6.

[33] MARTONE, M. *Onboard Quantization for Interferometric and Multichannel Synthetic Aperture Radar (SAR) Systems*. 9 p. Dissertation.

[34] METZ, L., POOLE, B., PFAU, D. and SOHL-DICKSTEIN, J. Unrolled Generative Adversarial Networks. *CoRR*. 2016, abs/1611.02163. Available at: http://arxiv.org/abs/1611.02163.

[35] MIRZA, M. and OSINDERO, S. Conditional generative adversarial nets. *ArXiv preprint arXiv:1411.1784*. 2014.

[36] MOREIRA, A., PRATS IRAOLA, P., YOUNIS, M., KRIEGER, G., HAJNSEK, I. et al. A tutorial on synthetic aperture radar. *IEEE Geoscience and remote sensing magazine*. IEEE. 2013, vol. 1, no. 1, p. 6–43.

[37] NOWATZKI, J., ANDRES, R. and KYLLO, K. Agricultural remote sensing basics. North Dakota State University. 2004.

[38] PÁSZTO, V., REDECKER, A., MACKŮ, K., JÜRGENS, C. and MOOS, N. *Data Sources*. January 2020. 3-38 p. ISBN 978-3-030-26625-7.

[39] RADFORD, A., METZ, L. and CHINTALA, S. Unsupervised representation learning with deep convolutional generative adversarial networks. *ArXiv preprint arXiv:1511.06434*. 2015.

[40] RONNEBERGER, O., FISCHER, P. and BROX, T. U-net: Convolutional networks for biomedical image segmentation. In: Springer. *International Conference on Medical image computing and computer-assisted intervention*. 2015, p. 234–241.

[41] ROTH, K., LUCCHI, A., NOWOZIN, S. and HOFMANN, T. Stabilizing training of generative adversarial networks through regularization. *Advances in neural information processing systems*. 2017, vol. 30.

[42] SKOLNIK, M. I. Introduction to radar. *Radar handbook*. McGraw-Hill New York, NY, USA. 1962, vol. 2, p. 21.

[43] THEIS, L., OORD, A. v. d. and BETHGE, M. A note on the evaluation of generative models. *ArXiv preprint arXiv:1511.01844*. 2015.

[44] VERHOEF, W. *Theory of radiative transfer models applied in optical remote sensing of vegetation canopies*. Wageningen University and Research, 1998.

[45] WALKER, W. Introduction to RADAR Remote Sensing for Vegetation Mapping and Monitoring. *A Ph. D. presentation: Woods Hole Research Center*. 2014, vol. 22.

[46] WU, J. Introduction to convolutional neural networks. *National Key Lab for Novel Software Technology. Nanjing University. China*. 2017, vol. 5, no. 23, p. 495.

[47] XUE, J. and SU, B. Significant remote sensing vegetation indices: A review of developments and applications. *Journal of sensors*. Hindawi. 2017, vol. 2017.

[48] ZHANG, Q., YUAN, Q., ZENG, C., LI, X. and WEI, Y. Missing data reconstruction in remote sensing image with a unified spatial–temporal–spectral deep convolutional neural network. *IEEE Transactions on Geoscience and Remote Sensing*. IEEE. 2018, vol. 56, no. 8, p. 4274–4288.

[49] ZHU, J.-Y., PARK, T., ISOLA, P. and EFROS, A. A. Unpaired image-to-image translation using cycle-consistent adversarial networks. In: *Proceedings of the IEEE international conference on computer vision*. 2017, p. 2223–2232.

Appendix A

The contents of the included storage media

- **doc** – LaTeX source files and the thesis in PDF format
- **dataset** – Datasets
- **models** – Pretrained models
- **src** – Source files
- **poster.pdf** – Poster from the assignment
- **README.md** – Provides additional description and manual to setup the source files.

Appendix B

Source code usage

Describes the usage of source files stored on included media. If you would like to run the code please go through `README.md` first. The separate modules that a user can use.

- `pix2pix.py` – main module – for training, evaluation and all architecture changes and modifications
- `training_graphs.py` – plots the training graphs based on the MetaCentrum output files
- `data_analysis.py` – analyzes the dataset
- `dataset_creation_pipeline.py` – creates the dataset

`pix2pix.py` and `data_analysis.py` have the shell alternatives to run the jobs on MetaCentrum.

Utilities and other files used by modules:

- `data_generator.py` – generates picture batches
- `models.py` – contains model architecture and sample generations
- `plotting.py` – plot logic implemented
- `utils.py` – utility methods
- `data_constants.py` – constants for dataset download
- `prety_confusion_matrix.py` – prints the confusion matrix. I am not an author, but I modified the code.

Help message output from `pix2pix.py`:

```
Usage: pix2pix.py [OPTIONS]

Options:
  --season TEXT            dataset you want to train on
  --dir TEXT               main directory, the directory structure is
                           required, for more, look at README.MD
  --version INTEGER        specifies which model is trained and evaluated
  --epochs INTEGER         number of epochs to train a model
  --patch INTEGER          only numbers 70 and 142 are supported
  --custom_mae BOOLEAN     uses custom MAE, tries to better redistribute
                           the dataset
  --evaluate_only BOOLEAN  the last generator model of the chosen version
                           is evaluated
  --deviation BOOLEAN      the deviance of the last generator model of the
                           chosen version is calculated
  --dry BOOLEAN            dry run, model graphs (gan, discriminator and
                           generator) are generated and saved in
                           model_graphs directory
  --help                   Show this message and exit.
```

Appendix C

Additional graphs.

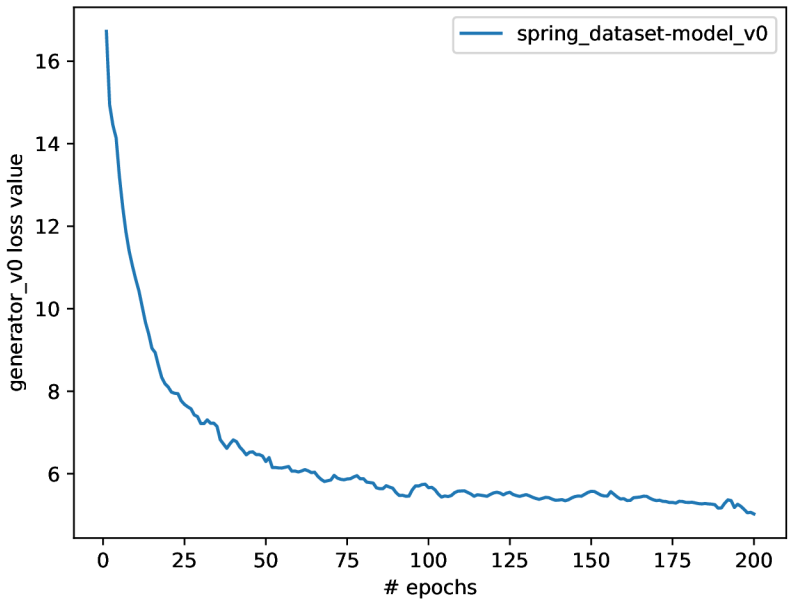

Figure C.1: Discriminator loss value for model 0. The loss value per epoch is the average of all losses in the epoch.

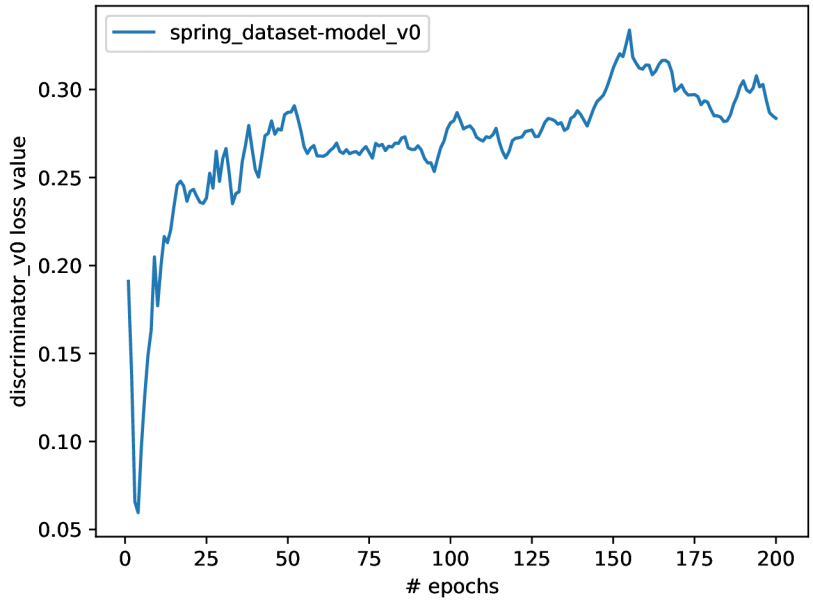

Figure C.2: Generator loss value for model 0. The loss value per epoch is the average of all losses in the epoch.

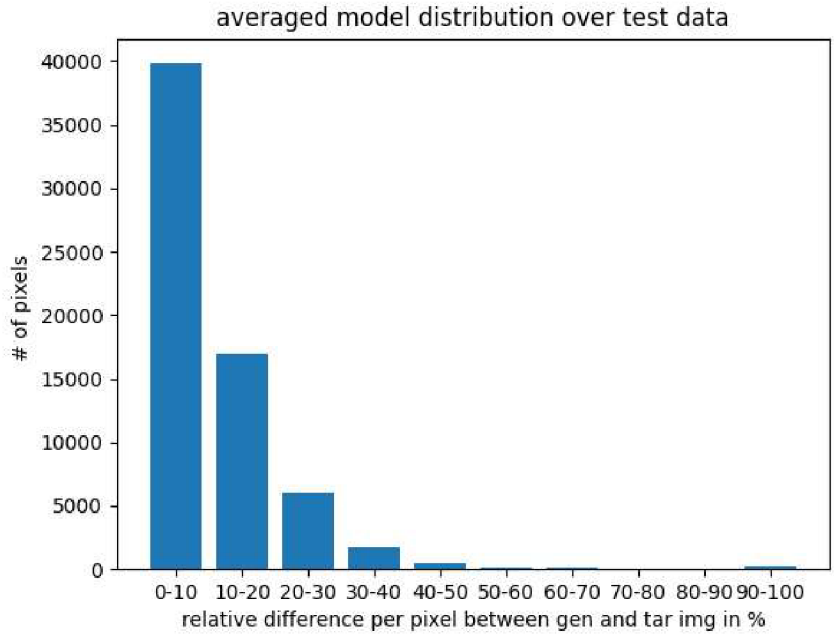

Figure C.3: Relative difference between generated and targeted image of model 2. Calculated for each pixel, over the whole test data of the All dataset. The sum is averaged over images and shown on the graph.

Figure C.4: Confusion matrix for model 1 - All dataset. Compares predicted pixels with present NDVI.

Figure C.5: Confusion matrix for model 2. Compares predicted pixels with present NDVI.

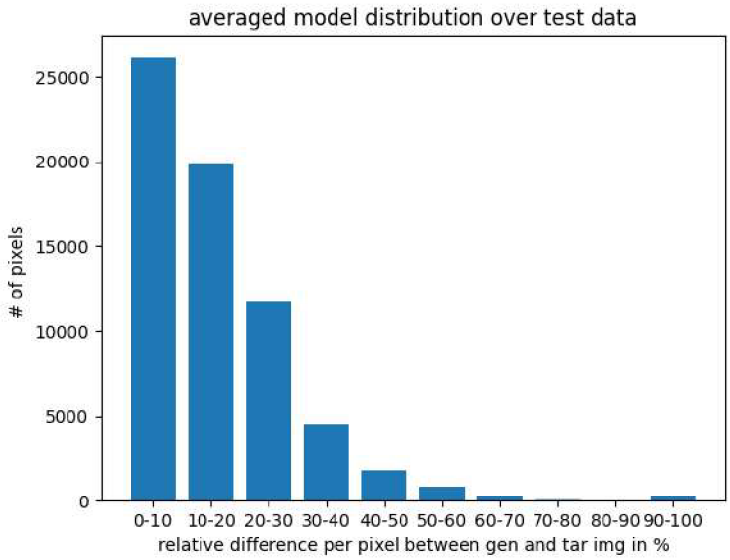

Figure C.6: Relative difference between generated and targeted image of model 3.

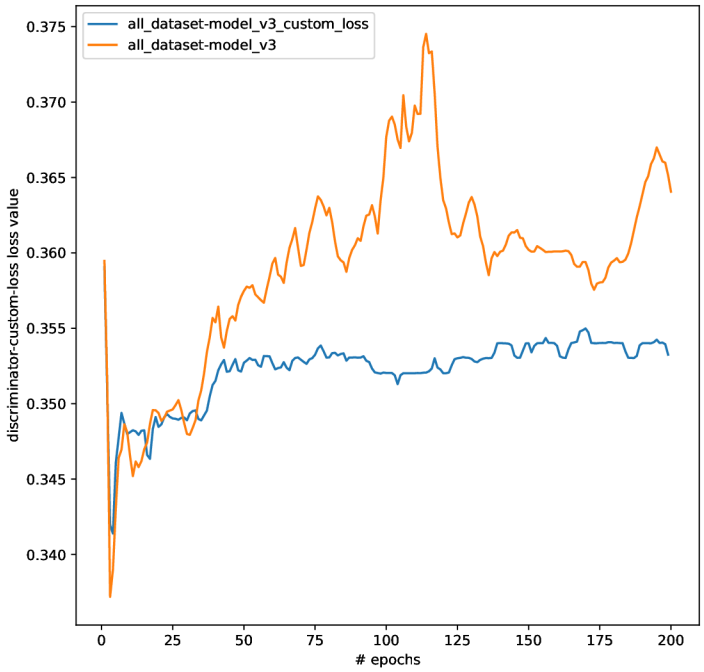

Figure C.7: Discriminator loss value for model 3 with custom loss. The loss value per epoch is the average of all losses in the epoch.

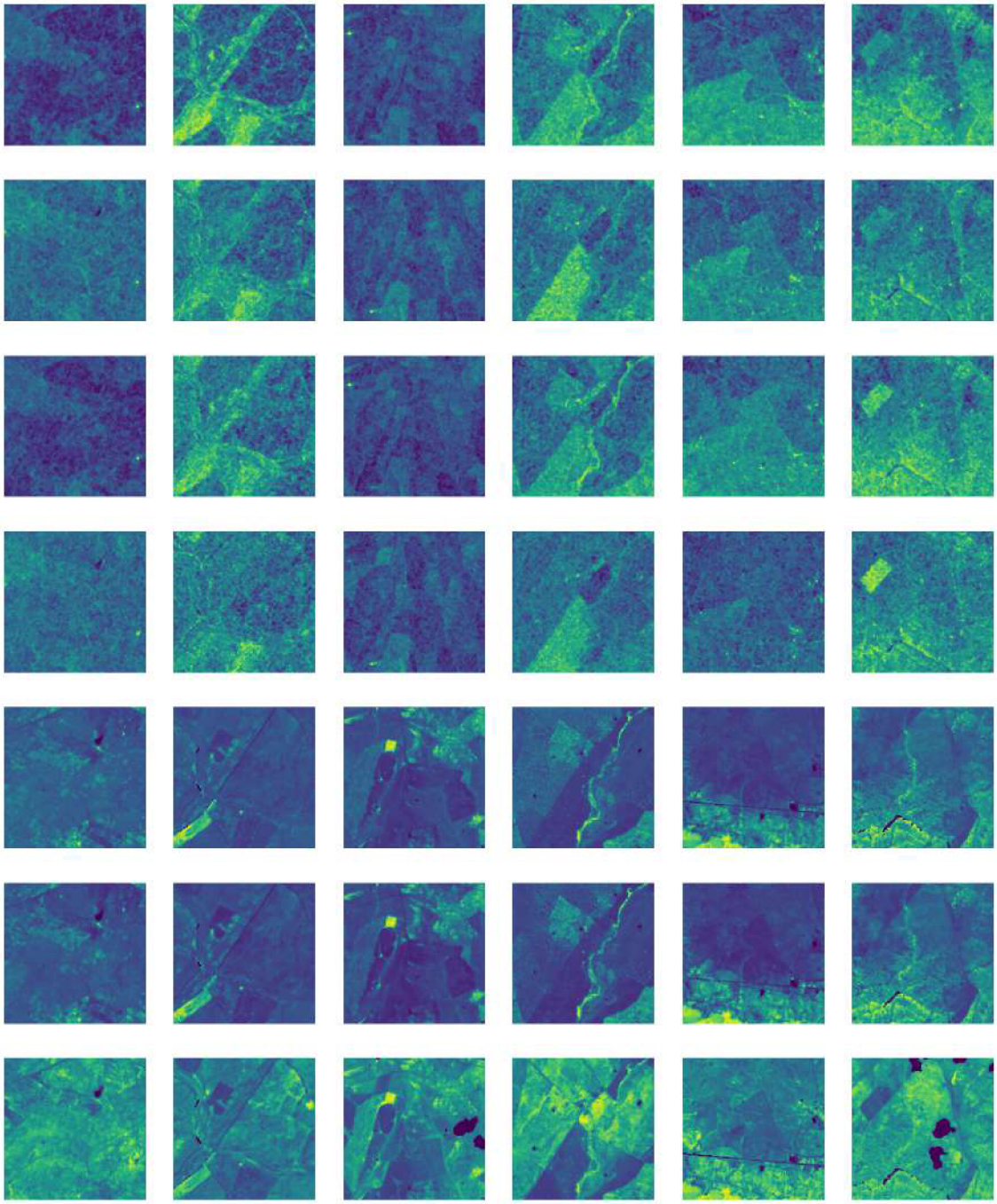

Figure C.8: Worst 6 pictures generated by model 3 fixed from test data of the All dataset. Each column represents the picture. The rows from the top: last revisit time radar VH, last revisit time radar VV, actual radar VH, actual radar VV, last revisit time NDVI, generated NDVI, and ground truth.