NTNU
Norwegian University of
Science and Technology

Master's thesis

2022

Renate Askevold, Mikael Vågen

NTNU
Norwegian University of
Science and Technology
Faculty of Information Technology and Electrical
Engineering
Department of Electronic Systems

Master's thesis

Renate Askevold
Mikael Vågen

Automated Mapping and Change Detection of Rivers and Inland Water Bodies

by Semantic Segmentation of SAR Imagery using Deep Learning

May 2022

NTNU
Norwegian University of
Science and Technology

NTNU
Norwegian University of
Science and Technology

Automated Mapping and Change Detection of Rivers and Inland Water Bodies

by Semantic Segmentation of SAR Imagery using Deep Learning

Renate Askevold
Mikael Vågen

Signal Processing, Electronics Systems Design and Innovation
Submission date: May 2022
Supervisor: Tor Andre Myrvoll
Co-supervisor: Knut Alfredsen, Oddbjørn Bruland

Norwegian University of Science and Technology
Department of Electronic Systems

Renate Askevold, Mikael Vågen

Automated Mapping and Change Detection of Rivers and Inland Water Bodies by Semantic Segmentation of SAR Imagery using Deep Learning

Master thesis
for the degree Master of Science, M.Sc.
in Signal Processing, Electronics Systems Design and Innovation

Trondheim, May 2022

Norwegian University of Science and Technology
Faculty of Information Technology,
Mathematics and Electrical Engineering
Department of Electronics and Telecommunications

NTNU
Innovation and Creativity

NTNU

Norwegian University of Science and Technology

Master thesis
for the degree of Master of Science, M.Sc.
in Signal Processing, Electronics Systems Design and Innovation

Faculty of Information Technology,
Mathematics and Electrical Engineering
Department of Electronics and Telecommunications

Abstract

The aim of this thesis is to detect extents of water bodies and rivers in Norway by semantic segmentation of Synthetic Aperture Radar (SAR) satellite images. Two deep neural network architectures are implemented and trained to perform the segmentation task. The focus is to examine the accuracy and usability of automatic water mapping for change detection in rivers and water bodies.

The project is connected to the UN's fifteenth sustainable development goal: Life on Land, which aims to prevent consequences of climate change from causing land degradation and biodiversity loss. Automatic water mapping is an important tool in limiting potential consequences related to climate driven flood events, aiding disaster management and emergency response efforts.

Existing machine learning models obtain high accuracy if trained on highly accurate data, however, creation of such data is labor intensive, time consuming and requires manual work. To assess this challenge, the dataset created for this project is a collection of Sentinel-1 SAR images of different regions in Norway, with labels from the Norwegian Mapping Authorities. Images and labels are processed and tiled, then divided into training and validation data with an 85% to 15% split. The two chosen deep segmentation architectures are U-Net with a ResNet50 backbone, pre-trained on the ImageNet dataset, and DeepLabV3+ with an Xception backbone, pre-trained on the Cityscapes dataset.

U-Net achieves an overall class-weighted Jaccard index of 0.974, with a Jaccard index of 0.846 and accuracy of 0.886 specific to the water class. DeepLabV3+ yields a weighted Jaccard index of 0.976, detecting water with a class specific Jaccard index of 0.856 and an accuracy of 0.892.

Labels from the Norwegian Mapping Authorities are detailed, and includes narrow streams and rivers that are not detectable in SAR images with resolution 20x22m. As such, performance metrics are limited by differences between the original labels and images, and may be improved by providing labels with corresponding accuracy when testing. Obtaining SAR images with higher resolutions may increase model performance in terms of detecting very small changes in water extents. The current versions of the models are likely accurate enough to be used as part of a semi-automatic flood monitoring notification system. Yet, more accurate data and testing is needed to fully realize the potential of a reliable, fully automated system.

Sammendrag

Målet med denne oppgaven er å detektere utstrekninger av vannforekomster og elver i Norge ved semantisk segmentering av Syntetisk Apertur-Radar (SAR) satellittbilder. To dype nevrale nettverksarkitekturer er implementert og trent til å utføre segmenteringsoppgaven. Fokuset er å undersøke nøyaktigheten og nytteverdien til automatisk vannkartlegging for endringsdeteksjon i elver og vannforekomster.

Prosjektet er knyttet til FNs femtende bærekraftsmål: Livet på Land, som har som mål å forhindre at konsekvenser av klimaendringer forårsaker landforringelse og tap av biologisk mangfold. Automatisk vannkartlegging er et viktig verktøy for å begrense mulige konsekvenser knyttet til klimadrevne flomhendelser, samt hjelpe til med katastrofehåndtering og beredskapsarbeid.

Eksisterende maskinlæringsmodeller oppnår høy nøyaktighet hvis de trenes på svært nøyaktige data, men å skaffe og forberede slike data krever mye tid og manuelt arbeid. For å ta fatt på denne utfordringen er datasettet laget for dette prosjektet en samling Sentinel-1 SAR-bilder av ulike regioner i Norge, med etiketter fra Statens kartverk. Bilder og etiketter blir behandlet og delt i fliser, deretter delt inn i trenings- og valideringsdata med en fordeling på 85% til 15%. De to valgte dype segmenteringsarkitekturene er U-Net med en ResNet50-ryggrad forhåndsopplært på ImageNet-datasettet, og DeepLabV3+ med en Xception-ryggrad forhåndstrent på Cityscapes-datasettet.

U-Net oppnår en samlet klassevektet Jaccard-indeks på 0,974, med en Jaccard-indeks på 0,846 og nøyaktighet på 0,886 spesifikt for vannklassen. DeepLabV3+ gir en vektet Jaccard-indeks på 0,976, og oppdager vann med en klassespesifikk Jaccard-indeks på 0,856 og en nøyaktighet på 0,892.

Etiketter fra Statens kartverk er detaljerte, og inkluderer smale bekker og elver som ikke er mulig å detektere i SAR-bilder med oppløsning 20x22m. Derfor er ytelsesberegninger begrenset av forskjeller mellom de originale etikettene og bildene, og kan forbedres ved å bruke etiketter med tilsvarende nøyaktighet ved testing. Innhenting av SAR-bilder med høyere oppløsning kan øke modellens ytelse når det gjelder å oppdage svært små endringer i vannmengdene. De nåværende versjonene av modellene er sannsynligvis nøyaktige nok til å brukes som en del av et halvautomatisk varslingssystem for flomovervåking. Likevel er det nødvendig med mer nøyaktige data og testing for å fullt ut realisere potensialet til et pålitelig, helautomatisert system.

Preface

We would like to extend our utmost gratitude to our supervisor Tor Andre Myrvoll for ensuring that we have had access to the necessary resources for this thesis project, as well as providing valuable guidance throughout the duration of the project. We would also like to thank Knut Alfredsen and Oddbjørn Bruland for defining the thesis project statement, and for helping us acquire labels for the dataset. Finally, we wish to thank all those whom agreed to meet to discuss various theoretical topics which is the foundation of the work conducted here.

Contents

Appendices

List of Tables

List of Figures

List of abbreviations

ADC Analogue to Digital Converter

ASF Alaskan Satellite Facility

AUPRC Area Under Precision-recall Curve

EM Electromagnetic

EW Extra Wide

GAN Generative Adversarial Network

GIS Geographic Information System

GRD Ground Range Detected

IW Interferometric Wide

LUT Look-Up Table

MMSE Minimum Mean Square Error

PRC Precision-recall Curve

PRF Pulse Repetition Frequency

px Pixel

RAR Real Aperture Radar

ROI Region of Interest

SAR Synthetic Aperture Radar

SGD Stochastic Gradient Descent

SSH Secure Shell Protocol

Chapter 1

Introduction

Since the industrial revolution, greenhouse gas emissions have risen to cause increased surface temperatures on the planet. Recent studies suggest an increased likelihood of prolonged rainfall, drought, glacial melting and storms due to global climate change. As a direct consequence, the severity of natural disasters such as floods and hurricanes increases [Hey19]. In an attempt to reduce the costs associated with such disasters, new technological initiatives aim to be green and sustainable. Innovations also aim to aid in solving problems associated with global climate change.

“In 2015 the United Nations released the 17 sustainable development goals as an urgent call to action by all countries. They recognize that ending poverty and other deprivations must go hand-in-hand with strategies that improve health and education, reduce inequality, and spur economic growth - all while tackling climate change and working to preserve our oceans and forests.” [Nat21]

With climate change leading to increased flood risk in many areas, communities are at risk of losing homes and vital infrastructure. This has the potential to cause serious economic damage, and can even cause loss of life. It is therefore of utmost importance to reduce the consequences of these events. Preventative measures may include early warnings to enable evacuations, or barriers to prevent water flow in critical areas. Flood risk depends on weather and current water levels, and it is therefore beneficial to monitor rivers and water bodies in real-time to quickly be able to implement such preventative measures and help local emergency efforts with their work. Drones have become a popular way of gathering data for monitoring purposes, but are often banned in such conditions due to potential helicopter rescue missions and other forms of aid they may hinder. However, data used to monitor such events still needs to be accessible and frequently updated to reflect current water extents. Such data is available from various satellite platforms, which obtain close to real-time images due to their orbiting nature. Collecting data from satellites to monitor flooding is non-invasive and does not hinder any form of aid.

By the end of 2020, the number of Earth observing satellites approached 1000 [Moh21], all tasked with gathering data for research specific purposes. The Copernicus initiative funded by the European Space Agency is just one example of a satellite constellation that gathers landcover, atmospheric, and weather data on a global scale. The initiative aims to provide data for sustainable development and management in various sectors. Through its Sentinel missions, Copernicus provides high quality data to aid decision making before, during and after humanitarian crisis and natural disasters [ESAa].

Limited computational resources and restricted access has previously prevented satellite data from being leveraged to its full potential. However, global space initiatives now offer vasts amounts of free data to end users, hoping it will lead to more rapid scientific discoveries. Recent leaps in processing power has simultaneously fueled a breakthrough in the fields of artificial intelligence and machine learning. This development has resulted in a rapid increase in the use of machine learning for remote sensing applications. The field of study has seen a recent increase in research surrounding image segmentation, aiming to detect water, crops, ice, buildings and more. Competitions, such as SpaceNet [Spa22], yield various new object detection, image segmentation and real-time monitoring algorithms based on solutions provided by competitors through various challenges. Image segmentation research primarily focuses on optical images with unsupervised machine learning techniques. Notably, Possa and Maillard [PM17] and Obida et. al [Obi+] have detected rivers and water bodies with high accuracy using thresholding and support vector machine algorithms. More recent research by Garg et. al [Gar+21], however, shows that pre-trained deep learning models outperform more traditional machine learning algorithms such as support vector machines and random forests.

Optical imagery has successfully demonstrated the possibilities of Earth observation by remote sensing. Machine learning algorithms applied to optical images commonly achieve good results on landcover classification and feature identification tasks. However, optical sensors rely on reflected light in the visible and near infrared parts of the electromagnetic spectrum, which is blocked by the formation of clouds in the atmosphere. Since the sensors only rely on sunlight, they are unfortunately unable to gather data at night. Optical imaging is also limited to the color information on the top surface of objects. Active imaging sensors are not limited to imaging when there is an adequate supply of sunlight. One such sensor is the radar, or more specifically the airborne synthetic aperture radar (SAR). In addition to night vision, SAR penetrates cloud cover and is able to provide information on structural and electrical properties of objects by their scattering characteristics [Hal09].

The focus of this thesis is semantic segmentation of rivers and water bodies in SAR images, with the purpose of exploring the possibilities for change detection. Water has very distinct scattering characteristics at radar wavelengths. A trait that may be leveraged to distinguish water from other landcover types. The ability to detect change in water body extents from SAR images may aid in flood risk assessments, as well as real-time flood monitoring. These potential applications connects this thesis project to the UN's fifteenth sustainable development goal: Life on Land.

With this background, the project approaches the segmentation problem by testing the abilities of two deep learning architectures to distinguish water from other landscape types. One issue in SAR based remote sensing is the lack of large, labeled benchmark datasets [Gar+21]. For this reason, the task also encompasses the creation of such a dataset. Multiple models are developed by training the architectures on the labeled dataset that is produced. Their performance is then evaluated on a validation set similar to the training data, and unseen test data. The models are also tested on data with known changes in water extents, either due to human interference or flood events, and an assessment of their performance in terms of change detection is conducted.

The structure of this thesis follows the structure of the workflow, which starts by defining the problem and exploring the necessary theoretical background. This background is then used to create a dataset and implement, train and test models on this dataset. Finally, the performance of the models is discussed and analyzed. Chapter 2 presents the theoretical background on both data and deep learning. It starts with the basic theory on machine learning, image classification, image segmentation and two segmentation architectures. Further, it explains the basic theory of SAR data acquisition and processing. It concludes by explaining the processing needed to prepare SAR data for image segmentation. Following the theoretical background is Chapter 3, Methodology. This chapter explains the process of acquiring data from the Sentinel-1 mission, processing it and building a dataset. Then it covers the implementation, training and evaluation of two segmentation model architectures. Further it covers the structure of the code written for this thesis project. The results, including training performance and test performance of the models, are presented in Chapter 4, and is followed by a discussion in Chapter 5. In the discussion, different aspects of the results are discussed in light of the project goals and relevant theory. Chapter 6 contains some thoughts on further work and possible improvements. Finally, the conclusion in Chapter 7 summarizes the thesis, and some extra information on the dataset can be found Appendix A.

Chapter 2

Theoretical Background

This chapter aims to provide the theoretical background necessary to understand image segmentation of SAR imagery by deep learning. The first part presents the theory of different deep learning techniques and models. This includes the basics of Convolutional Neural Networks, image segmentation, and relevant model parameters. Furthermore, some image classification and segmentation architectures are presented in-depth. It will then cover background on satellite imaging with synthetic aperture radar (SAR), including the basics of SAR data acquisition and some background on scatterer characteristics. Necessary SAR image processing techniques is presented, followed by general pre-processing steps necessary to generate images compatible with deep learning algorithms.

Following a thorough review of the background material presented in the project preceding this thesis [VA21]. Some relevant material from the project is deemed necessary to provide a complete theoretical foundation for the work conducted here. This material is partially restated, indicated by footnotes.

2.1 Deep Learning

The field of deep learning is a sub-field of machine learning, which again is a part of the field of artificial intelligence (AI). There is no exact definition of what deep learning is, but a proposed definition is: *"DL is a class of machine learning algorithms that use multiple layers of computational units where each layer learns its own representation of the input data. These representations are combined by later layers in a hierarchical fashion."* [Ekm21]. This is an abstract definition, but is based on a part of deep learning referred to as deep neural networks. As the name might suggest, deep neural networks are inspired by neurons in the brain[1].

[1] Relevant material from project report [VA21].

2.1.1 Convolutional Neural Networks [2]

The Convolutional Neural Network (CNN) is a subclass of neural networks commonly used in image classification. Input images have a height, a width and a number of channels. A common case is three channels for RGB. When feeding an image to a CNN each neuron will see a number of pixels that is defined by the size of the kernel and the stride. As shown in Figure 2.1, a convolutional layer has a height, width and a number of channels or feature maps, just like the input image. The number of neurons needed to cover the image is usually less than the number of pixels which leads to a lower resolution for each layer down in the network. Each channel of neurons share weights and maps to one feature. They create feature maps by firing if they detect said feature. As the weights are shared it does not matter where the feature is located, this leads to CNNs being translation invariant.

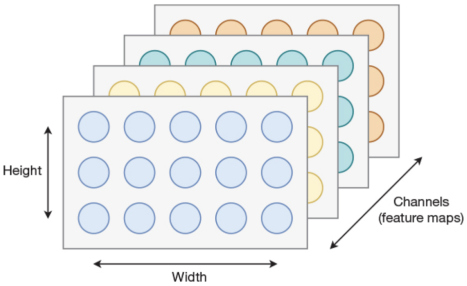


Figure 2.1: Illustration of a convolutional layer [Ekm21, Figure 7-4 page 178].

By connecting multiple convolutional layers where the output of one layer is the input of the next layer we get a network. Each layer will then represent a combination of features from previous layers, which makes it possible to detect more complex features. To avoid the dimensions of the layers getting smaller and smaller, one can use padding to keep the dimensions consistent. At the end of a CNN it is normal to have one or more fully connected layers. The fully connected layers are able to detect features the convolutional layers might not see. The very last layer often has the same number of neurons and outputs as there are classes. Upon classification the neuron with the highest activation in this last layer maps to a specific class, which is the model's prediction.

CNNs often also contain pooling layers. Pooling layers reduce feature map channel dimensions, which reduces the number of parameters that have to be learned by the model. In addition, pooling layers reduce the positional dependencies of features, which further aids the model in being translation invariant [Sav]. Max pooling is one of the most common pooling operations, which involves keeping the highest value from a group of neuron outputs.

2.1.2 Training a Neural Network

To train a neural network, a training dataset is needed. This dataset consists of features $X = \{x_n\}$ and labels $Y = \{y_n\}$, which represents the task the network is intended to do. The output of the network can be seen as an estimated value $\hat{y_n}$ from function f of the weights W, biases B, and the input x_n. This gives Equation (2.1).

[2] Relevant material from project report [VA21].

$$\hat{y_n} = f(x_n; W, B) \tag{2.1}$$

To ”train” the network means to optimize the weights and biases such that the predicted value $\hat{y_n}$ is as close as possible to the true value y_n. For the training algorithm to know how weights and biases should be updated, there has to be a defined function that gives a measure of how far the predicted value is from the true value. Such a function is called a loss function and can be defined as $l(y, \hat{y})$. The choice of function decides the objective of the training. The total loss of training is defined in Equation (2.2).

$$L(W, B, X, Y) = \frac{1}{|X|} \sum_n l(y_n, f(x_n; W, B)) \tag{2.2}$$

The goal of training can then be defined as finding the values $\hat{W}$ and $\hat{B}$ that minimizes the total loss for the training set. This is presented in mathematical form in Equation (2.3).

$$\hat{W}, \hat{B} = \underset{W,B}{\operatorname{argmin}}\, L(W, B, X, Y) \tag{2.3}$$

A way of performing this minimization is to use gradient descent. This technique is based on calculating the gradient of the loss function and using it to update W and B in the direction that gives a smaller loss. The model parameters are initialized with random values or Gaussian noise. The gradient of the loss is then computed based on the training set. A parameter called the learning rate, ϵ, decides how much W and B are changed in each iteration. Weights and biases will be updated according to Equation (2.4). These steps will be repeated until convergence in the minimum point.

$$\begin{aligned} W^{(i)} &= W^{(i-1)} - \epsilon \Delta_W L(W, B, X, Y) \\ B^{(i)} &= B^{(i-1)} - \epsilon \Delta_B L(W, B, X, Y) \end{aligned} \tag{2.4}$$

Traditional gradient descent is not the only way to train a network. The Adam algorithm is a popular choice for training as it is, according to D.P. Kingma and J.L. Ba Adam, “straightforward to implement, is computationally efficient, has little memory requirements, is invariant to diagonal rescaling of the gradients, and is well suited for problems that are large in terms of data and/or parameters” [BK15]. Adam is a method of stochastic optimization that only requires computation of first order gradients. It uses the estimation of the first and second moment of the gradients to compute individual and adaptive learning rates for each parameter. Adam gets its name from Adaptive Moment Estimation. It accelerates the gradient descent algorithm by using the exponentially weighted moving average of the gradients, combined with the adaptive learning rates, which results in faster convergence.

Loss functions define the objective of training, and different functions are suitable for different applications. The loss function is what decides if the prediction is good or bad. To get good results from training one needs to choose the function that best measures quality in the intended use case. Mean absolute error and mean square error are the most common loss functions in regression problems, while cross entropy loss is common in classification problems. In computer vision tasks such as image segmentation, DICE loss is one of the most used functions.

The formula for the categorical cross entropy loss function is presented in Equation (2.5). Categorical cross entropy is binary cross entropy extended to a multiclass problem. Entropy is a measure of randomness in data, and in the case of probability distributions, a smaller value of entropy indicates less uncertainty in the distribution. It works great as a loss function when the goal is to train the model to be more certain of its predictions, which is achieved by minimizing the cross entropy. Categorical cross entropy is often paired with a softmax activation function that outputs the probability of each class. A requirement of categorical cross entropy is that the labels must be one-hot encoded such that only the true values differ from zero.

$$\textit{cross entropy loss}: \quad e(y,\hat{y}) = -\sum_{c=1}^{N} y_c(\ln \hat{y}_c) \tag{2.5}$$

DICE loss, shown in Equation (2.6), is a region based loss that measures the overlap between the true label and the predicted label, where a value of zero indicates a perfect overlap. By design, DICE takes into account class imbalance in the dataset. Datasets for segmentation are naturally prone to class imbalances, which makes DICE loss a popular choice for segmentation tasks since it can be used to maximize overlap, a trait that prevents model inference based on the prominence of each class in the dataset.

$$DICE \ : \ e(y,\hat{y}) = 1 - \frac{2\sum_i^N y_i\hat{y}_i}{\sum_i^N y_i + \sum_i^N \hat{y}_i} \tag{2.6}$$

The activation function can be found in the hidden layers of a neural network. It is the part of the layer that decides if the neuron should be active or not. The activation function takes the weighted input and transforms it to a value to be given to the next layer. Each layer in the network is a linear combination of the input multiplied with the weights. Feeding this result to the next layer creates another linear combination. To be able to perform complicated operations the network must be able to model non-linear behavior. Activation functions are added to introduce the non-linearity needed for the model to perform such complex mappings.

There exists a lot of different activation functions, all suitable for different applications. These include the linear activation, Sigmoid, ReLU and softmax functions. Usually all the hidden layers in a network use the same activation function. The output layer normally uses another activation function that fits the intended application of the model. For example, the softmax function is good for multiclass classification, while sigmoid is better for multilabeling and the linear function for regression. In the context of CNNs, the two most commonly used activation functions are softmax for the output layer and ReLU for all hidden layers.

ReLU is short for Rectified Linear Unit and is defined in Equation (2.7). The advantage of using ReLU is that it is computationally efficient because it uses simple operations such as comparison, multiplication and addition, and because it does not activate all neurons at the same time. ReLU is also less prone to vanishing gradients and gives faster convergence since it is linear and non-saturating. On the other hand, ReLU has some disadvantages, such as the dying ReLU problem, which is due to all negative gradient values being zero. This may lead to the weights and biases of some neurons not being updated and thus not being activated.

$$f(x) = max(0, x) \tag{2.7}$$

The softmax activation function is the most common output unit to use in multiclass classification problems and is described by Equation (2.8). The output of the softmax function is a value between 0 and 1 for each class, and the outputs of all classes sum to 1. For these reasons, the output of the softmax function is considered the probability of the input belonging to a certain class. The class with the highest probability is the predicted class.

$$softmax(z)_f = \frac{e^{z_i}}{\sum_{j=1}^{n} e^{z_j}} \tag{2.8}$$

2.1.3 Evaluating a Neural Network

There are multiple metrics that may be used to evaluate model performance, depending on application. Some performance metrics that are relevant for image segmentation and classification models are presented.

The precision metric is used to evaluate the number of correct predictions out of all the predictions made by the model. Equation (2.9) shows the micro-averaged precision, which is the global precision where all samples are treated equally [SL09]. It sums the total number of true positives (TP), N, and divides this by the sum of total positive predictions ($TP + FP$). This yields the rate of correct, positive predictions. Depending on application, two other common weightings for the precision metric are macro-averaged and weighted average [Leu22]. Macro-averaged precision computes the precision for each class individually and returns the unweighted mean. This implies that macro-averaging treats all classes equally without considering possible class imbalances. If class imbalances are to be considered, weighted averaging is a common alternative to micro-averaging. Weighted average precision also computes class based precision, but returns the weighted mean of the class results with respect to class imbalances. Equation (2.10) shows the weighted average precision P_w in the case of a binary classification problem. Here, N_{c_1} and N_{c_2} are the total number of samples in class 1 and 2, α and β are the ratios of class occurrences to the total number of occurrences for class 1 and 2, respectively.

$$\text{Precision} \ = \ \frac{\sum_{i=1}^{N} TP_i}{\sum_{i=1}^{N} (TP_i \ + \ FP_i)} \tag{2.9}$$

$$P_w \ = \ \alpha \left(\frac{\sum_{i=1}^{N_{c_1}} TP_i}{\sum_{i=1}^{N_{c_1}} (TP_i \ + \ FP_i)} \right) + \beta \left(\frac{\sum_{i=1}^{N_{c_2}} TP_i}{\sum_{i=1}^{N_{c_2}} (TP_i \ + \ FP_i)} \right) \tag{2.10}$$

The recall metric is used to evaluate the amount of correct labels out of all the predicted labels. The micro-averaged recall metric is shown in Equation (2.11) [SL09], and is the ratio of true positives (TP) to all positive labels ($TP + FN$). Equation (2.12) shows the weighted averaged recall R_w in a two class case, just as for the precision.

$$\text{Recall} \; = \; \frac{\sum_{i=1}^{N} TP_i}{\sum_{i=1}^{N} (TP_i \; + \; FN_i)} \tag{2.11}$$

$$P_w \; = \; \alpha \left(\frac{\sum_{i=1}^{N_{c1}} TP_i}{\sum_{i=1}^{N_{c1}} (TP_i \; + \; FN_i)} \right) + \beta \left(\frac{\sum_{i=1}^{N_{c2}} TP_i}{\sum_{i=1}^{N_{c2}} (TP_i \; + \; FN_i)} \right) \tag{2.12}$$

Precision and recall are inversely proportional metrics, that is, increasing one will decrease the other and vice versa. A balanced classification model should maximize both metrics with equal weighting. The $F1$-score, shown in Equation (2.13), combines precision and recall by their harmonic mean, and is therefore a common performance indicator for classification problems.

$$F_1 \; = \; \frac{2 \; \cdot \; \text{Precision} \; \cdot \; \text{Recall}}{\text{Precision} \; + \; \text{Recall}} \tag{2.13}$$

Binary classification models output a value that may be interpreted as the probability of a predicted value belonging to a certain class. In this case, a threshold is commonly chosen such that all predicted probabilities above this threshold are grouped in the *positive class*, and all that fall below this threshold are grouped into the *negative class.* The $F1$-score indicates the precision-recall performance given a specific threshold. Further analysis of model performance may require *precision-recall curves (PRCs)* [Bro18], which plots precision against recall across thresholds. Figure 2.2 shows the PRC of a perfect classifier which a good binary classifier should approach. The area under the PRC (AUPRC) may be calculated to give an indication of model performance in terms of precision and recall, a perfect classifier will have AUPRC = 1.0.

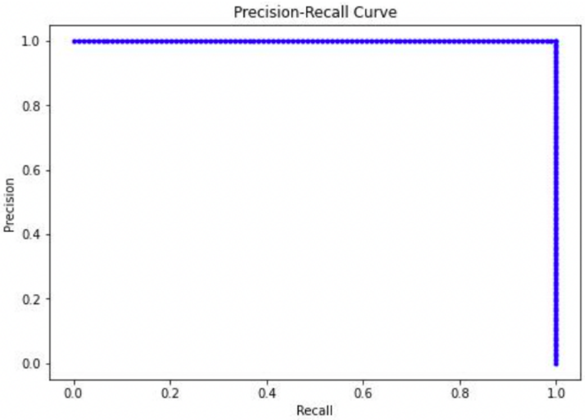


Figure 2.2: Precision-recall curve of a perfect classifier with AUPRC = 1.0.

Segmentation accuracy is simply the ratio of correctly classified pixels to all pixels, shown in Equation (2.14). Although accuracy is highly intuitive, it may fail to consider class representations. To account for this, it may also be calculated separately for each class to give model accuracy on specific classes [3].

$$\text{Segmentation Accuracy} \;=\; \frac{\text{Correct Pixels}}{\text{All Pixels}} \tag{2.14}$$

Segmentation models commonly use the Jaccard index to evaluate model performance on each individual class. In the case of semantic segmentation, the Jaccard index yields the ratio of correctly predicted pixels to all predicted- and labeled pixels. This ratio is a similarity coefficient between the set predicted by the model and the set of actual labels for each class, and is given by Equation (2.15) [Ste16]. X and Y are the two sets to be compared, for example predicted and actual sets [4].

$$J(X, Y) \;=\; \frac{|X \cap Y|}{|X \cup Y|} \tag{2.15}$$

2.1.4 Residual Neural Networks[5]

Residual neural networks (ResNet) are a type of CNN introduced in 2015, outperforming both GoogLeNet and VGGNet with its 152 layer deep architecture. Previous attempts to train very deep convolutional networks resulted in higher training error due to vanishing gradients and degradation. ResNet was introduced to address the issue and solved it by introducing skip connections, illustrated in Figure 2.3. Skip connections make it easier for the network to learn the identity function. The idea behind this is that the best solution might be close to the identity function as only small variations are needed to improve the accuracy, thus the learning algorithm will start to look for a solution in a space that is likely to have one [Ekm21, p. 216].

The structure of ResNet consists of groups of convolutional layers with a kernel of 3x3 and a stride of 1. Periodically, it uses convolutional layers with stride 2 to halve the dimensions and double the number of channels. After each convolutional layer ResNet uses batch normalization. The purpose of batch normalization is to normalize the values inside a network, and to stabilize and accelerate the learning process. It may be applied before or after the activation function. In ResNet it is applied before.

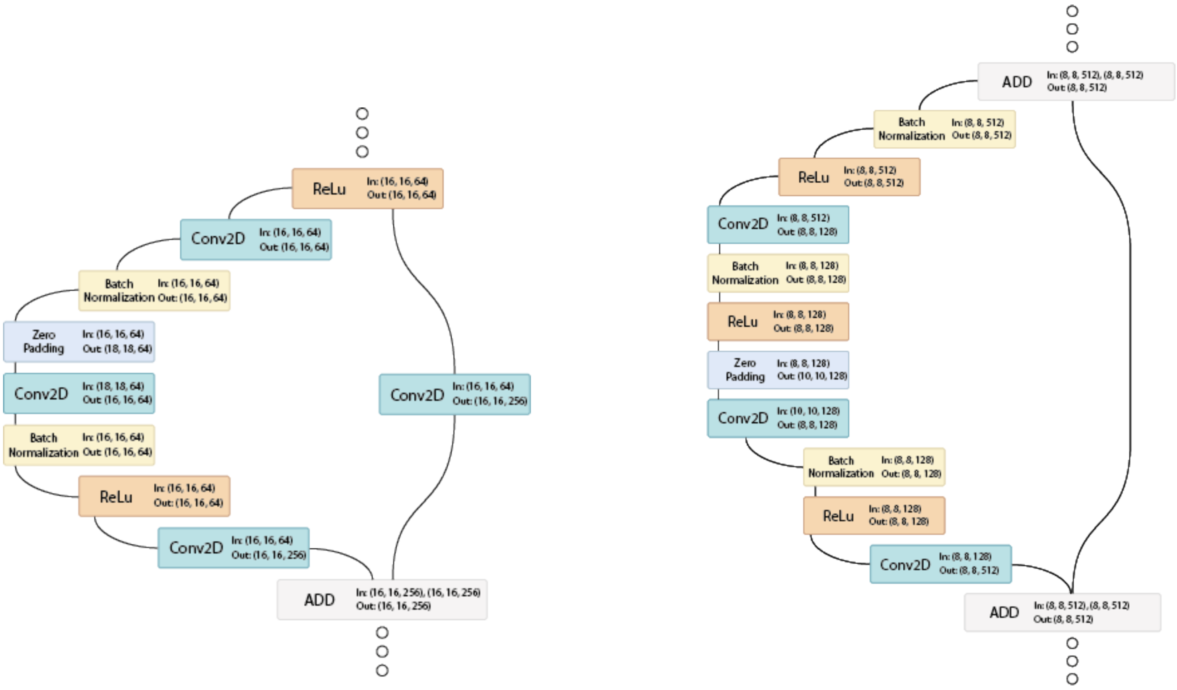
Skip connections are implemented by adding the output of a previous layer to the output of a later layer before passing it through the activation function. Mathematically this is explained in Equation (2.16), where F is used in place of the function for the layer to simplify. This type of skip connection is illustrated in Figure 2.3b, and is valid as long as the dimensions of the layers are the same. In ResNet, where dimensions are halved and number of channels doubled, a 1x1 convolution is applied to $x_{i,j,k}$ to get the right shape, as illustrated in Figure 2.3a.

$$y_{i,j,k} = ReLu(x_{i,j,k} + F_{i,j,k}(x)), \qquad i = 1...w, j = 1...h, k = 1...c \tag{2.16}$$

[3] Segmentation accuracy description from project report [VA21].

[4] Jaccard index description is partially from the project report [VA21].

[5] Relevant material from project report [VA21].


ReLu In: (16, 16, 64) Out: (16, 16, 64)
Conv2D In: (16, 16, 64) Out: (16, 16, 64)
Batch Normalization In: (16, 16, 64) Out: (16, 16, 64)
Zero Padding In: (16, 16, 64) Out: (18, 18, 64)
Conv2D In: (18, 18, 64) Out: (16, 16, 64)
Batch Normalization In: (16, 16, 64) Out: (16, 16, 64)
ReLu In: (16, 16, 64) Out: (16, 16, 64)
Conv2D In: (16, 16, 64) Out: (16, 16, 256)
Conv2D In: (16, 16, 64) Out: (16, 16, 256)
ADD In: (16, 16, 256), (16, 16, 256) Out: (16, 16, 256)
ADD In: (8, 8, 512), (8, 8, 512) Out: (8, 8, 512)
Batch Normalization In: (8, 8, 512) Out: (8, 8, 512)
ReLu In: (8, 8, 512) Out: (8, 8, 512)
Conv2D In: (8, 8, 512) Out: (8, 8, 128)
Batch Normalization In: (8, 8, 128) Out: (8, 8, 128)
ReLu In: (8, 8, 128) Out: (8, 8, 128)
Zero Padding In: (8, 8, 128) Out: (10, 10, 128)
Conv2D In: (10, 10, 128) Out: (8, 8, 128)
Batch Normalization In: (8, 8, 128) Out: (8, 8, 128)
ReLu In: (8, 8, 128) Out: (8, 8, 128)
Conv2D In: (8, 8, 128) Out: (8, 8, 512)
ADD In: (8, 8, 512), (8, 8, 512) Out: (8, 8, 512)

(a) Skip connection where the two layers do not have the same number of features.

(b) Skip connection where the two layers have the same number of features. Figure also illustrates channel reduction between layers to reduce the number of parameters.

Figure 2.3: Illustration of two different skip connections found in ResNet50.

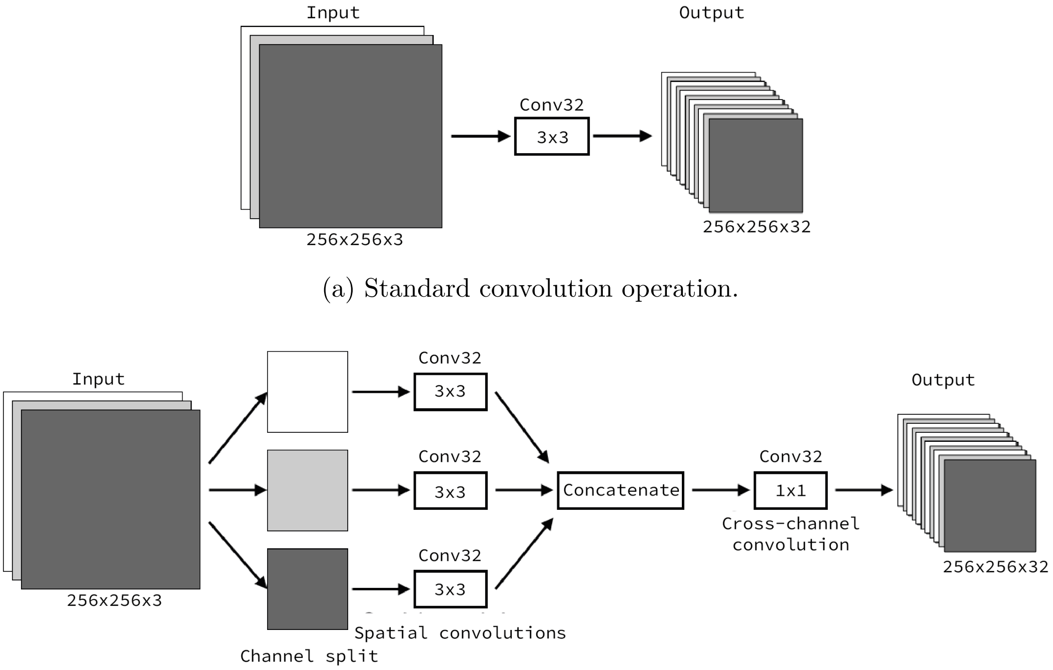
To reduce the total number of weights it has to learn, ResNet uses a trick when it gets deep into the network. It adds a 1x1 convolutional layer to temporarily reduce the number of channels in the input of the 3x3 convolutional layer, it then applies another 1x1 convolutional layer to increase the number of channels afterwards. This is illustrated together with a skip connection in Figure 2.3b.

2.1.5 Xception

Google first proposed the Xception architecture in 2017 as an improvement to their Inception V3 architecture. Inception V3 was the runner-up to the ResNet architecture in the 2015 ImageNet Large Scale Visual Recognition Challenge [Liu+15]. Xception is shown to achieve better results on the ImageNet dataset than Inception V3, and outperforms both Inception and ResNet on other datasets. [Cho17, p. 5]

Xception assumes that spatial (height and width) and cross-channel data are entirely uncorrelated. With this assumption it is possible to build more efficient networks by use of depthwise separable convolutions. Consider a standard convolution operation which maps one feature space consisting of all channels to a new feature space using a single kernel. Since only one kernel is responsible for extracting spatial and cross-channel features, it is likely that the resulting feature map has lost substantial amounts of information. Depthwise separable convolutions address this issue by splitting the convolution into multiple "sub-convolutions".

Spatial features are first extracted by NxN-convolutions performed on each channel separately, and then cross-channel features are extracted by a single 1x1-convolution. A block diagram of conventional convolution is shown in Figure 2.4a, while Figure 2.4b shows the same for depthwise separable convolutions. Separately extracting features in the spatial and cross-channel dimensions results in more efficient parameter estimation, as well as richer feature maps. The latter is, however, only valid if the assumption that there is little to no correlation across channels is true. Google has proved this to be empirically true, as Xception outperforms most recent architectures that do not make this assumption [Cho17].



(a) Standard convolution operation.

(b) Depthwise separable convolution as implemented in the Xception architecture.

Figure 2.4: Comparison of standard convolution with the depthwise separable convolutions used in the Xception architecture to provide more efficient calculation of parameters and richer feature maps.

The complete architecture of the Xception network is shown in Figure 2.5. It mainly consists of blocks of separable convolutions with ReLU activation. The entry flow increases the number of channels and reduces spatial resolution by a combination of separable convolutions and max pooling layers. The middle flow performs feature extraction without changing the dimensions of the feature map, while the exit flow increases the channels further before a logistic regression layer performs the actual classification. Xception also utilises residual skip connections to enable deep training without vanishing gradients and degradation, as introduced in Section 2.1.4.

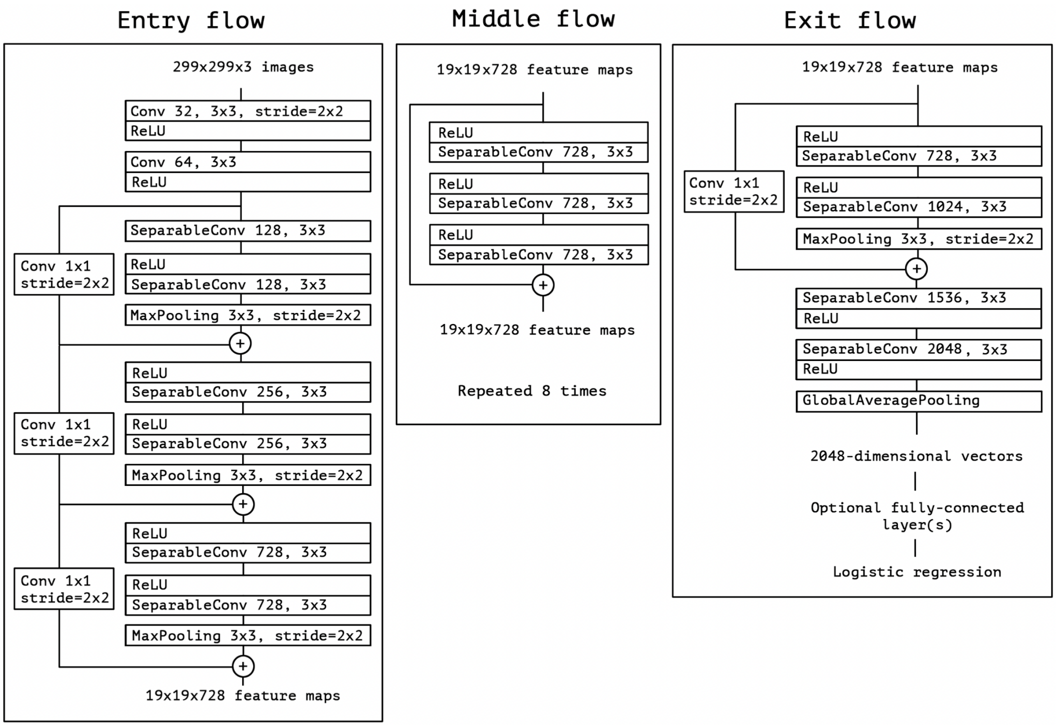


Figure 2.5: Architecture of the original Xception neural network. Diagram from [Cho17, p. 5].

2.1.6 Transfer Learning

Big networks are trained on enormous amounts of data to achieve high performance. This kind of training is very resource intensive, requiring large amounts of computational power and storage. Gathering large, high quality data sets to train such networks is also often a challenge. Transfer learning is a technique used in machine learning to build well performing neural networks with little available data and computational resources. Transfer Learning is realized in one of two ways: either by using a pre-trained network and training it further with a specialized dataset for a similar task as it was intended for, or by taking part of a pre-trained network and adding it to a network to be specialized for another but similar task. The idea behind transfer learning is that even if the pre-trained network is not trained for the same task, it might have some abilities that can easily transfer to another task. For example if you have a huge network that can classify images of 1000 different animals and objects, and you want to classify ten different dog breeds, the network is already good at recognizing animals so it will not need to train as much to recognize the dog breeds. To change from 1000 to 10 classes one would have to exclude the last few layers of the model and add new ones that divides the output into 10 instead of 1000. When using transfer learning and adding new layers, the new layers will have random weights so the training process will be faster if the transferred layers are frozen in the beginning of training to not ruin the pre-trained weights. The pre-trained layers can then be unfrozen and fine tuned afterwards to get a better result[6].

[6] Relevant material from project report [VA21]

2.2 Image Segmentation

In an image classification task a neural network learns to recognize a set of objects based on a whole image. After passing an image through the network the image is assigned one or multiple labels based on what it is trained to do. In object detection the network is also taught to highlight an object's location by drawing a box where it thinks the object is. To obtain the exact shape of the object the network must be trained for segmentation, which is a combination of detection and classification. When performing segmentation of an image, each pixel is assigned a label based on which class it belongs to. The network will then return an annotated image. A model trained for segmentation will return more detailed information on the image than a classifier or a detector.

There are two types of segmentation; instance segmentation and semantic segmentation. In instance segmentation the network is segmenting out each instance of an object in the image. If there are three dogs in an image, an instance segmentation network will recognize each dog as an individual and segment out the pixels belonging to each dog. In the resulting image each dog will be annotated with its own label. When doing semantic segmentation each pixel is assigned a class and all pixels with the same class is segmented together. A semantic segmented image of three dogs will just tell which pixels are dog-pixels and not distinguish the individual dogs. Hence all dogs will be annotated with the same label. This project focuses on semantic segmentation [7].

2.2.1 Semantic Segmentation[8]

In semantic image segmentation the goal is to classify all the pixels of the image into a set of different classes. A key aspect needed for this to be possible is to keep the dimensions of the image intact through the network. If the network input is a three dimensional RGB-image, the output should be the same height and width as the input, but the number of channels should now be the number of classes, rather than the three RGB-channels. Figure 2.6 shows a simple network that fulfills the requirements of semantic segmentation. The input of the network is an RGB image, and the output is a five dimensional array which equals the image in height and width, but where all pixels in the same class have the same color.

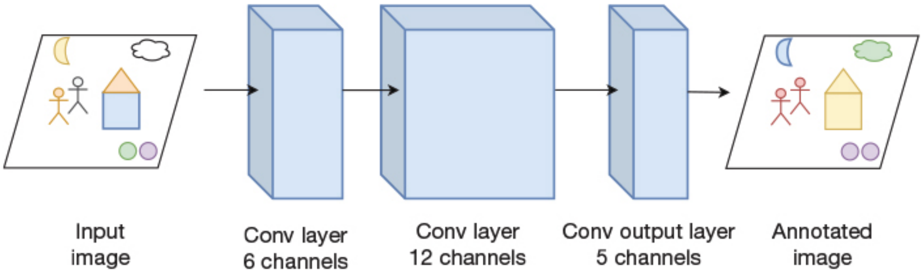


Figure 2.6: Illustration of a simple architecture for semantic segmentation [Ekm21, Figure B-8 appendix B].

[7] Relevant material from project report [VA21].
[8] Relevant material from project report [VA21].

As explained in the book by Magnus Ekman [Ekm21, Semantic Segmentation, appendix B], the network in Figure 2.6 will not be sufficient. It consists of a convolutional network without pooling, where all layers have a stride of one. All layers in such a network will have the same height and width. This is done so that the output will have the right shape. Due to an increasing number of channels with constant height and width, the cost is that the total number of features to calculate and store per layer increases with depth. This is the opposite of what happens in a typical CNN where the resolution decreases and the number of features remain constant or decreases with depth.

Traditional CNNs work very well for image classification, but to utilize these networks for semantic segmentation there must be a way to ensure that the output has the right dimensions, and that the classes are mapped to the right pixels. Since the resolution decreases with depth, the network must upsample the image to its original resolution. Some common techniques are nearest neighbor interpolation and bilinear interpolation, described in "*Learning Deep Learning*" [Ekm21, Upsamling tecniques, appendix B]. These techniques can be implemented by spacing the pixels with zero-valued pixels and applying convolution with different convolutional kernels depending on technique. Instead of using such specialized kernels the weights can be learned in the same fashion as a normal convolutional layer. This results in a deconvolutional layer and is explained in greater detail in [Ekm21]. Deconvolutional layers should not be confused with the mathematical deconvolution operation. As previously mentioned, resolution may also be reduced by max pooling. Pooling, like convolution, can be undone. When undoing a max pooling operation the pixel value is placed in the same location as the maximum value and the space in between pixels is filled with zero-valued pixels, similar to what is done in deconvolution. Max pooling and unpooling is illustrated in Figure 2.7.

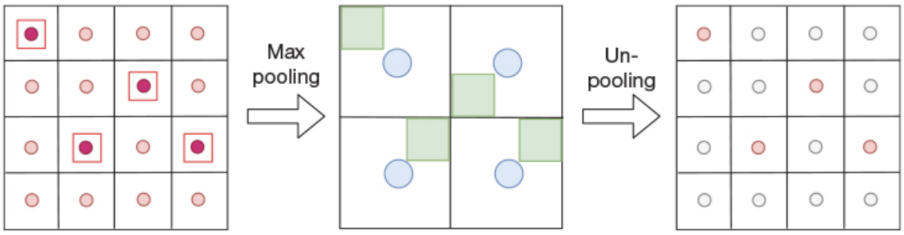


Figure 2.7: Illustration of max pooling and unpooling. To the left the pixel with the highest value in each group of four is marked with a darker color. In the middle the green square is the past location of the max value and the blue circle is the new location. To the right the non-zero pixels are marked red [Ekm21, Figure B-11 appendix B].

Using deconvolution and unpooling, a learning deconvolution network (DeconvNet) was proposed by Noh, Hong and Han in 2015 [NHH15]. Their network is illustrated in Figure 2.8. In their network they have mirrored the VGGNet for image classification, discarding the output layer. By using the opposite operations going from 224x224x3 to 1x1x4096 at its smallest and back up to a segmentation pixel map with a resolution of 224x224x21 identifying 20 different objects leaving one class for things that are not classified.

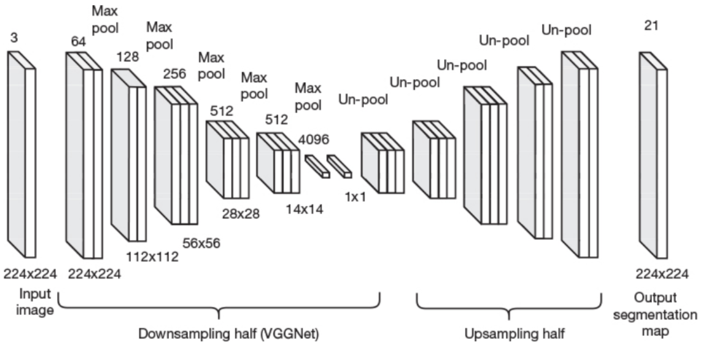


Figure 2.8: Illustration of a learning deconvolutional neural network (DeconvNet). Each block represents a convolutional layer [Ekm21, Figure B-13 appendix B].

2.2.2 U-Net

A popular network architecture for image segmentation is the U-Net. U-Net was introduced in 2015 by Ronneberg, Fischer and Bronx and was intended for biomedical image segmentation [RFB]. The idea behind the architecture is that the deconvolutional part of the network would improve with access to more data. A U-Net model mainly consists of two parts, the encoder or the downsampling half and the decoder or the upsampling half. The U-Net illustrated in Figure 2.9 is based on VGGNet just like the DeconvNet in Figure 2.8. The encoder part of the network is equal to the downsampling part of DeconvNet and VGGNet. The decoder is where U-Net is different. It mirrors the encoder layers with deconvolution and unpooling, but uses a different input. U-net concatenates the output of the previous layer with the output of the matching convolutional layer of the encoder, using the concatenated layers as the input to the next layer in the decoder. This allows the decoder to have more information available when upsampling, resulting in improved accuracy. The output of the decoder is then a segmentation map like that of DeconvNet. U-Net gets the name and the U-shaped architecture from the way it uses copies of output from the encoder in the decoder, otherwise it would have the same hourglass shape as DeconvNet.[9]

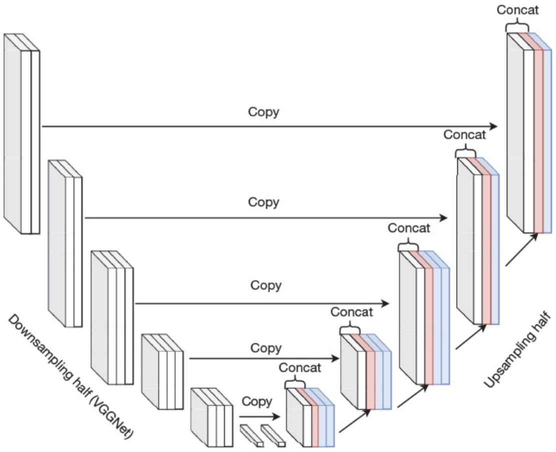


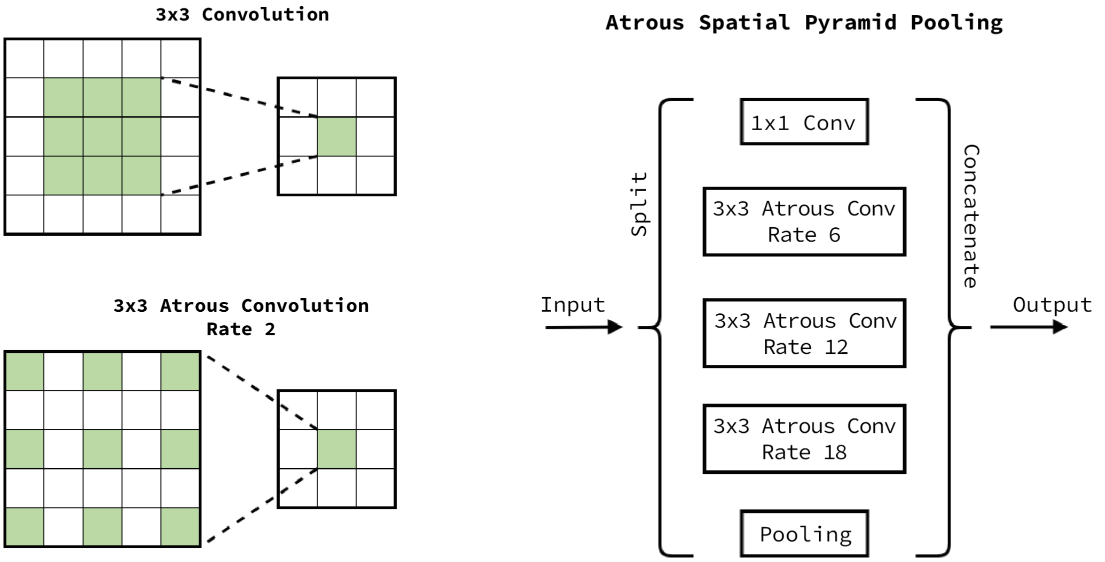
Figure 2.9: Illustration of U-Net architecture [Ekm21, Figure B-14 appendix B]. White layers are copied from the encoder to the decoder. Red layers are output from the previous layer that is concatenated with the white layer. Blue layers illustrate convolutional layers.

[9] Relevant material from project report [VA21].

2.2.3 DeepLabV3+

DeepLabV3+ is a state-of-the-art fully convolutional segmentation architecture introduced in 2018. It employs an encoder and decoder similar to U-Net, but leverages Atrous Spatial Pyramid Pooling (ASPP) to extract richer feature maps, as well as only utilising a single concatenation of encoder output to the decoder for improved efficiency. Encoder-decoder structures such as U-Net are excellent at boundary recognition, while richer feature maps provided by ASPP enable recognition of image context at multiple levels. DeepLabV3+ combines both of these methods to provide the mutual benefits. The architecture performed best on the PASCAL VOC 2012 and Cityscapes datasets upon release [Che+].

The encoder of DeepLabV3+ expands the U-Net encoder with ASPP by use of multiple atrous convolutions at different rates. Atrous convolutions are similar to standard convolution operations, but expand the field-of-view available to a convolutional kernel without increasing kernel size, as shown in Figure 2.10a. This enables feature extraction with greater context without severely increasing the computational complexity as much as if the kernel size is increased. A complete ASPP setup is shown in Figure 2.10b, and is simply a stack of atrous convolutions with different rates. This provides feature extraction with different levels of context and localization properties.



(a) Comparison of standard and atrous convolution.

(b) ASPP used by DeepLabV3+ to extract features at multiple fields of view.

Figure 2.10: DeepLabV3+ uses atrous spatial convolutions at multiple rates to extract features with multiple fields of view without increasing computational complexity.

To accurately obtain object boundaries, the feature maps from the encoder are passed through a decoder which uses a combination of upsampling and convolution to obtain an output at the same resolution as the input. The output of the encoder is upsampled and concatenated with the output of a layer with matching resolution in the backbone network (ie. ResNet). This ensures that the low-level features provided by the image classification encoder is not lost. Figure 2.11 shows the complete architecture of DeepLabV3+ where Xception is the chosen network backbone for classification purposes. While any backbone may be used, Xception proves to provide the best performing segmentation model [Che+, p. 13].

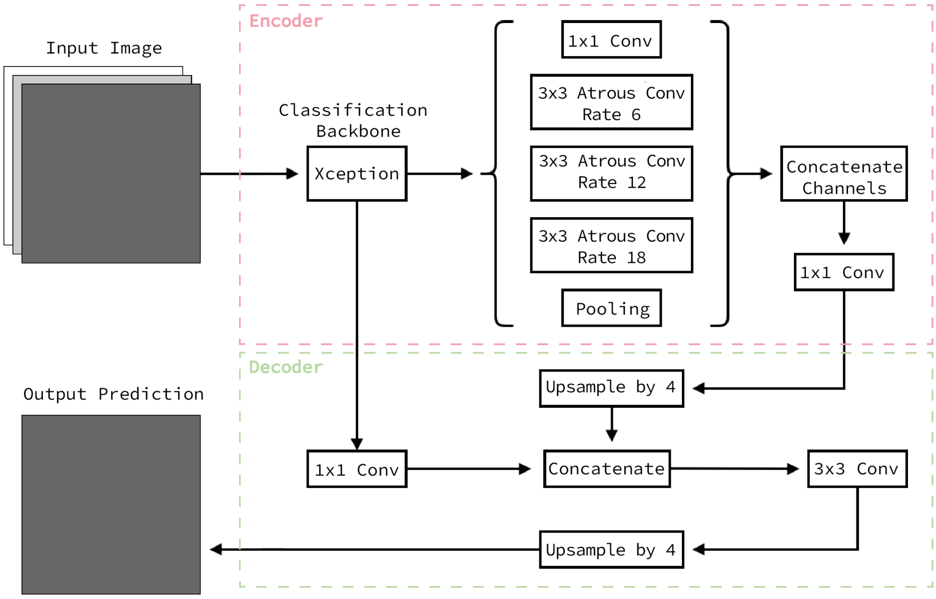


Figure 2.11: Architecture of DeepLabV3+. The encoder performs ASPP on the output of the feature extracting backbone, which is the Xception network introduced in section 2.1.5. The decoder combines the output of the encoder with the output of a layer with equal resolution from Xception. To achieve output resolution equal to the input resolution, the decoder upsamples the acquired features multiple times.

2.3 Synthetic Aperture Radar Imaging

Advancements in radar technology provides the means to acquire high-resolution images which complement the more common thermal and optical images. Synthetic Aperture Radars (SAR) address various shortcomings associated with optical imagery. Such shortcomings include the inability to deal with obstructing weather, and limited atmospheric penetration [FL99, p. 3]. To gain a fundamental understanding of SAR imagery provided by airborne sensors, the basic principles and benefits of pulsed radar systems and SAR is presented. This section further covers various acquisition parameters related to SAR images acquired by satellite remote sensing. Finally, SAR-products acquired by Sentinel-1 as part of ESA's Copernicus program is briefly presented.

2.3.1 Principles of Pulsed Radar Systems

Radar systems emit electromagnetic (EM) radio-frequency (RF) pulse trains toward a target commonly referred to as a scatterer. Upon interaction with the scatterer, parts of the transmitted signal is reflected back to the radar. The backscattered signal is received and processed by the radar system [RSH10, p. 4]. Analysis of backscatter may provide information about the distance to the target, relative target velocity, target structure, or the dielectric properties of the target [Pod18][Wan08, p. xii]. Figure 2.12 shows a simple, ground based radar system. The radar antenna emits pulses in the *range* direction, and the cross-section of the emitted beam pattern at any point is defined by the *azimuth*- and *elevation* directions. The beam is scattered by the airplane upon interaction, and a small part of the beam is reflected back to the radar antenna.

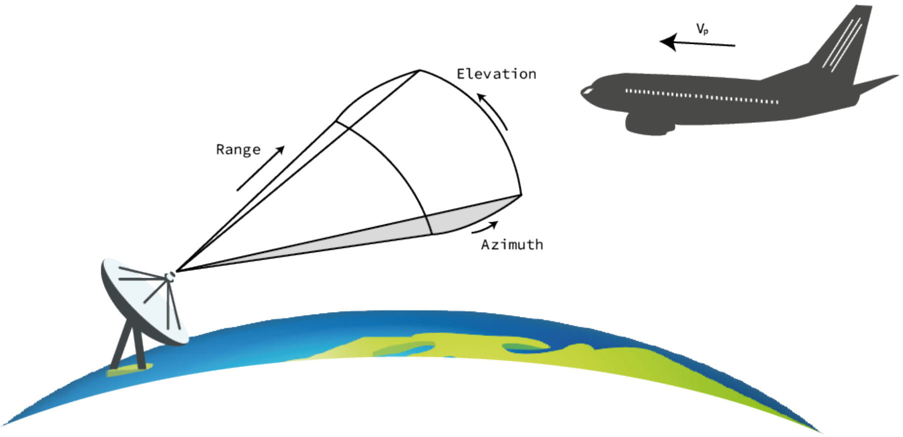


Figure 2.12: Basic illustration of a ground based radar detecting air traffic.

Distance to the target, known as range R, can be determined by measuring the travel time τ of the pulse, that is, the time from transmit to receive. The range to the target is then given by Equation (2.17) [Jan21, p. 23]. The constant c is the speed of light at which all electromagnetic signals travel. The distance is divided by two to account for two-way propagation.

$$R = \frac{c\tau}{2} \tag{2.17}$$

Since the airplane is moving relative to the radar, the return signal will be frequency-shifted relative to the transmitted signal. This shift in frequency is known as a Doppler shift, f_d, and may be used to determine the aircraft's velocity, v, as shown in Equation (2.18) [RSH10, p. 275]. The Doppler shift may be negative or positive, indicating that a target is moving away from, or towards, the radar antenna. The wavelength λ is related to the frequency, f, of the transmitted wave as in Equation (2.19), and ϕ is the angle between the range direction and velocity vector of the airplane.

$$f_d = \frac{2v}{\lambda}\cos\phi \tag{2.18}$$

$$\lambda = \frac{c}{f} \tag{2.19}$$

Received power, P_{rx}, from a scatterer depends on the range to the scatterer, as well as various characteristics of the transmitted pulse and target. Equation (2.20) is the radar equation describing the relationship between the received power, radar cross section and range. P_{tx} is the power of the transmitted pulse, G_{tx} and G_{rx} are antenna gains at transmit and receive, respectively [RSH10, p. 64].

$$P_{\text{rx}} = \frac{P_{\text{tx}}G_{\text{tx}}G_{\text{rx}}\lambda^2\sigma}{(4\pi)^3R^4} \tag{2.20}$$

As is apparent from Equation (2.20), received backscatter is a measure of a target's radar cross section, σ. When an incident wave interacts with a scatterer, the radar cross section is defined as the power scattered by this scatterer in a specific direction. Cross section thus characterizes the effects of the target on incident waves, and is defined in Equation (2.21), where E refers to power per unit area [RSH10, p. 219]. This information is directly related to reflectivity and is commonly used to infer something about target structure, dielectric properties, materials, etc., depending on the radar beam's incidence angle.

$$\sigma = \lim_{R\to\infty} 4\pi R^2 \frac{|E_{scattered}|^2}{|E_{incident}|^2} \tag{2.21}$$

Radar systems most commonly utilize radio frequencies in the microwave range of the electromagnetic spectrum, ranging from 300MHz to 300GHz [You18]. This corresponds to wavelengths between 1mm and 1m. These wavelengths propagate with minimal atmospheric interference, as shown in Figure 2.13. As such, radars are commonly leveraged for earth observation purposes to minimize atmospheric effects on the resulting data. Microwave wavelengths also provide the benefit of penetrating cloud-cover and other weather related phenomena, which allows data collection independent of environmental conditions. In addition, radars are active sensors that do not rely on sunlight to operate, allowing data collection at nighttime [Wan08]. These characteristics favor radar over optical and thermal sensing methods for applications such as flood monitoring.

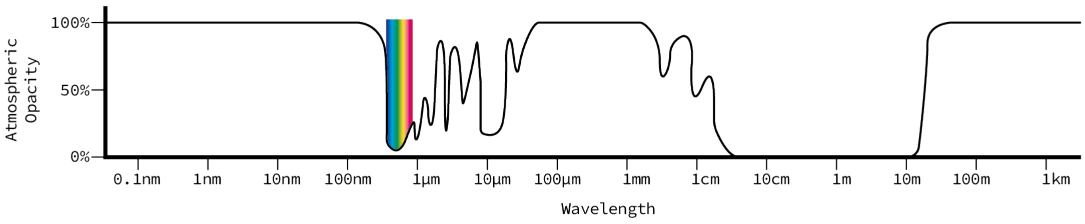


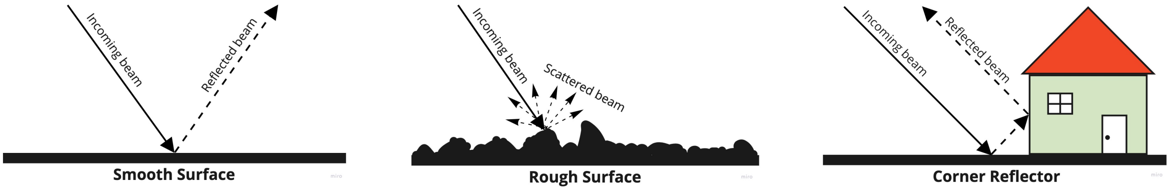
Figure 2.13: Atmospheric opacity/absorption at various wavelengths of the EM-spectrum [NASa]. Atmospheric windows are portions of the spectrum where atmospheric absorption is low, allowing EM-waves to travel with little interference.

2.3.2 Properties of Target Scatterers

As mentioned in Section 2.3.1, the contribution of the target scatterer(s) on the incident radar beam yields information about target reflectivity. High reflectivity of radar wavelengths implies a higher value of σ, and thus more received power, as shown by the radar Equation (2.20). Incidence angle, θ, and pulse wavelength, λ, are controllable parameters that affect reflectivity. Accounting for these parameters, reflectivity also depends on the structure of the scatterer, its dielectric properties, as well as its contents [Lil+19, p. 10].

Scatterer structure determines the direction and power of the backscattered signal. Rough surfaces generally produce more backscatter than smooth surfaces at non-zero incidence angles. Smooth surfaces act as *specular reflectors*, implying that the incident beam is reflected in a single direction [Sta18]. If the incidence angle of the radar beam is non-zero, the beam will in its entirety be reflected at an angle equal to the incidence angle, but in the opposite direction. This results in no backscatter returning to the radar. Specular reflection is illustrated in Figure 2.14a. Rough surfaces are *diffuse reflectors*, shown in Figure 2.14b. Diffuse reflectors will reflect the incoming beam in multiple directions, resulting in some backscatter to the radar antenna. A surface is considered rough if the height of the surface variations, h, meets the Rayleigh-criterion in Equation (2.22) [Lil+19, p. 10]. According to the Rayleigh-criterion, wavelength plays a key role in whether or not reflection will be specular or diffuse. Strong backscatter occurs when structures reflect most of the beam back to the radar. Corner reflectors are common examples of this, consisting of multiple specular reflectors arranged in such a way that the beam is returned to the radar in its entirety. Figure 2.14c shows a typical corner reflector that will result in strong backscatter [Lil+19, p. 11].

$$h > \frac{\lambda}{8 \cos \theta} \tag{2.22}$$



(a) Specular Reflection (b) Diffuse Reflection (c) Corner Reflection

Figure 2.14: Surface roughness and geometry of structures affects the amount of backscatter returning to the radar.

Different materials exhibit different dielectric properties which directly relate to their reflectivity. In general, materials with high dielectric constants, such as water (dielectric constant of 80), are strong reflectors [Hal09]. Such materials will provide stronger backscatter than those with lower dielectric constants. Moisture content within materials with low dielectric constants will thus affect backscatter, assuming the wavelength is large enough to penetrate the initial surface/boundary and reach the moisture. With careful selection of wavelengths, one may therefore examine the contents of the material. Resulting backscatter is then known as *volume reflections* [Lil+19, p. 11].

Backscatter will contain information about these properties. It is therefore possible to distinguish different types of scatterers and materials using radar. Radar images obtained by satellites may therefore be leveraged to classify landcover- and surface properties, among a multitude of other applications.

2.3.3 Synthetic Aperture Radar

Synthetic Aperture Radar (SAR) is a high resolution radar imaging system. Radar imaging systems combine measurements from multiple beams to form an image of an area of interest known as a scene. Figure 2.15 shows a typical imaging setup for remote sensing radar imaging systems. The radar is side-looking, which implies that it does not image along the nadir line directly beneath itself. This enables the radar to distinguish all returns in the range (R) direction. Beams are transmitted in slant range (R_s), forming a beam pattern on the ground with an extent in range and azimuth (X). The extent of the beam pattern in the range direction is commonly referred to as the swath of the radar.

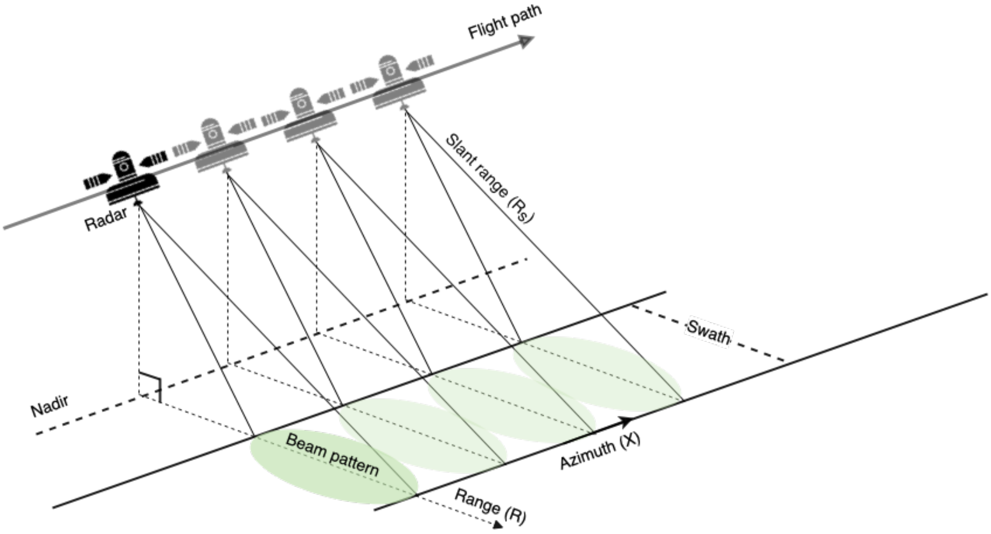


Figure 2.15: Illustration of side-looking airborne imaging radar.

The radar measures backscatter from a transmitted beam, then it moves along the flight path to the next azimuth position and repeats. Combining measurements of all the beams results in a range vs. azimuth image. Resulting 2D-images inherently have two resolutions, range-resolution and azimuth-resolution. Range-resolution ΔR is dependent on the bandwidth, B, of the transmitted pulse and is shown in Equation (2.23) [RSH10, p. 845]. Imaging radars with conventional radar apertures, referred to as real aperture radars (RARs), yield an azimuth-resolution ΔX equal to the half-power (3dB) azimuth beamwidth Θ_{3dB}, given in Equation (2.24). D_{x} is the aperture size of the antenna in the azimuth direction.

$$\Delta R = \frac{c}{2B} \tag{2.23}$$

$$\Delta X = \Theta_{\text{3dB}} = R_s \frac{\lambda}{D_{\text{x}}} \tag{2.24}$$

In remote sensing, obtaining high azimuth resolutions using RAR systems require very large antennas due to orbit height. For example, consider an RAR operating onboard a satellite with a center frequency of $f_0 = 10\text{GHz}$. The satellite is in low Earth orbit and the slant range distance to the scene is 650km. The mission is tasked with acquiring radar imagery of the surface with an azimuth resolution of $\Delta X = 10\text{m}$. The required aperture size will then be 1.95km, as shown in Equations (2.25), where c is the speed of light at which microwaves propagate. Launching a satellite with an antenna of that size is clearly not viable.

$$\begin{gathered}\lambda = \frac{c}{f_0} = \frac{3 \cdot 10^8 \frac{\text{m}}{\text{s}}}{10\text{GHz}} = 30\text{mm} \\ D_{\text{azimuth}} = R_s \frac{\lambda}{\Theta_{\text{azimuth}}} = 650\text{km} \cdot \frac{30\text{mm}}{10\text{m}} = 1.95\text{km}\end{gathered} \tag{2.25}$$

To achieve high azimuth resolutions without unrealistically large, physical antennas, synthetic apertures are used. Synthetic aperture is the technique of leveraging the movement of a radar mounted on a moving platform to simulate a larger aperture by signal processing. Backscatter acquired from each pulse is sampled and stored in an *azimuth vs. range* matrix, shown in Figure 2.16. Each row corresponds with a single received pulse. The azimuth direction is acquired by multiple beams and is sampled at the pulse repetition frequency (PRF), which is the frequency at which the radar emits pulses. This matrix represents an unfocused, raw SAR image.

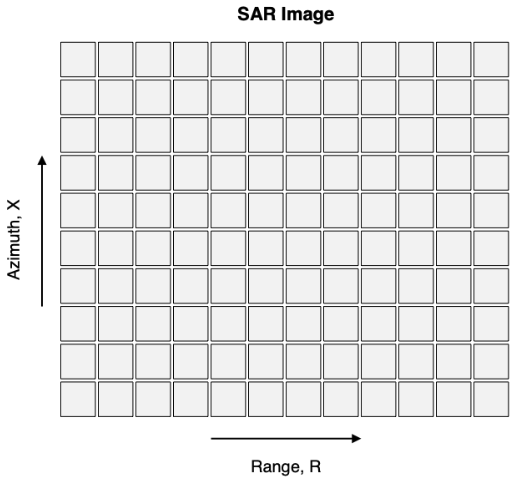


Figure 2.16: Unfocused SAR image, each beam has an extent in the range direction, while the azimuth direction is created by multiple pulses. Each box represents a resolution cell with dimensions depending on the range- and azimuth resolution.

Since the radar is moving relative to the target, received frequencies are doppler shifted in the azimuth dimension. The doppler phenomenon is what SAR leverages to synthetically increase azimuth resolution. Similar to how range-resolution is dependent on the bandwidth, B, of the transmitted pulse, doppler processing ensures azimuth-resolution is dependent on the doppler bandwidth, B_d. The formation of SAR images is called *focusing*, and is most commonly done by use of the Range-Doppler Algorithm, described by D. E. Jansing [Jan21, p. 68]. Assuming a synthetic aperture (flight/imaging distance) large enough to obtain correct interference patterns [Sci97], a simplified equation for the azimuth resolution is given in Equation (2.26) [Jan21, p. 44-45]. The velocities, v_g and v_a, are the ground- and air velocities of the satellite, respectively. The squint angle, ψ, is the angle between range, R, and the nadir line. In Figure 2.15, the radar is imaging broadside with no squint, $\psi = 0$.

$$\Delta X = \frac{0.886 v_g \cos\psi}{B_d} = \frac{D_{\mathrm{x}} v_g}{2 v_a} \tag{2.26}$$

It is worth noting that the given azimuth resolution, as well as the general theory of SAR presented here, is valid for the stripmap imaging mode which is introduced in Section 2.3.4. Stripmap mode is thought to provide the simplest and most intuitive introduction to the basics of SAR imaging. Other imaging modes may leverage the ability to coherently combine beams of the exact same regions to increase resolution, in which case Equation (2.26) may not be accurate. Without going into detail; coherent integration over multiple pulses is a common technique in radar to increase the signal-to-noise ratio [RSH10, p. 66]. Integrating over n beams will yield a processing gain of n, which implies the resulting SNR will be n times greater than if using a single pulse.

2.3.4 SAR Acquisition Parameters

Remote sensing SAR imaging systems operate with a set of parameters that determine the available data product type. SAR satellite missions will typically vary these parameters depending on customer orders and/or scattering characteristics of the regions being imaged. This ensures application-specific data products are available to end users. The acquisition parameters include radar center frequency, beam polarization, image acquisition mode and swath width.

Center frequency is predetermined by the mission, and is not altered while the SAR satellite is in operation. Center frequency is directly related to wavelength, and determines the penetration capabilities of the radar beam. The choice of operating frequency is application dependent. Longer wavelengths will penetrate deeper into vegetation and soil, and is thus more viable for studying volume scatter, as discussed in Section 2.3.2. Shorter wavelengths are optimal for high-resolution landcover mapping. Commonly utilized SAR frequencies are divided into a set of bands shown in Table 2.1.

Transmitted- and received SAR beams may be horizontally- (H) or vertically (V) polarized. SAR polarization is “like”-polarized if transmitted and received beam polarizations are the same (HH or VV), otherwise the acquired product is “cross”-polarized (HV or VH). Some SAR-platforms also have dual-polarization capabilities (ie. HH + HV) [Sar09, p. 19]. Dual-polarization generally provides more scatterer information than single-polarization beams

Table 2.1: Most common SAR frequency bands with some common applications, information from [Her+20].

Band	Frequency [GHz]	Applications
X	8 – 12	Urban monitoring, landcover mapping
C	4 – 8	Landcover mapping, ocean ice monitoring
S	2 – 4	Agriculture monitoring
L	1 – 2	Geophysical monitoring by interferometry

[Liu16], while flatter terrain features such as water are more distinguishable from other landcover features using horizontal polarization [Rao+].

Modern remote sensing SAR missions may acquire images in three different acquisition modes; stripmap, spotlight, and scanSAR, shown in Figure 2.17. When stripmap imaging, the squint angle of the radar is fixed and the beam is moved along the swath by the movement of the platform. In spotlight mode, the beam is steered to focus on a specific area for multiple beams while the platform is moving. Stripmap imaging can thus cover large areas, but the azimuth resolution is limited to the Doppler beamwidth. Coherently combining beams acquired by spotlight mode will provide higher azimuth resolutions of a smaller area. ScanSAR combines steering and platform movement to provide a compromise between azimuth resolution and coverage [Jan21, p. 12-13]. For stripmap and scanSAR imaging, the swath width may be altered to meet coverage requirements.

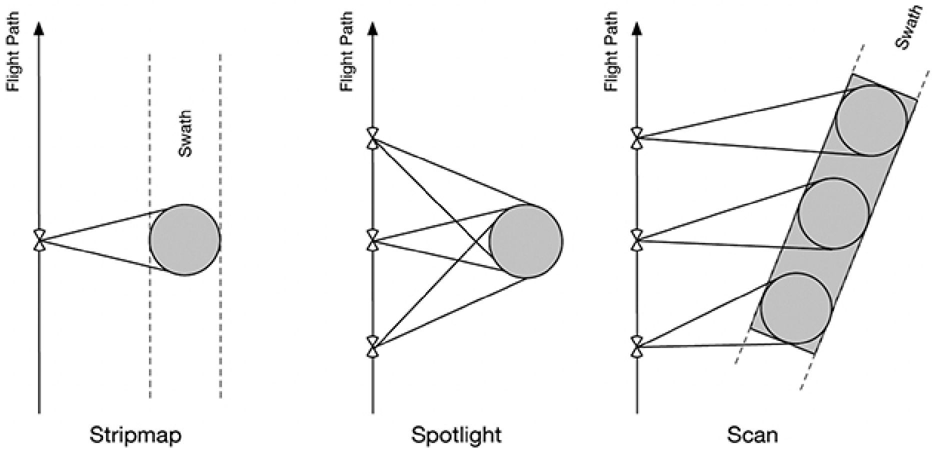


Figure 2.17: The three available acquisition modes for SAR imaging, image from [Jan21, p. 12].

2.3.5 Sentinel-1

Sentinel-1 is a SAR satellite mission conducted as part of the European Space Agency's Copernicus initiative. The mission constellation consists of two polar orbiting satellites, Sentinel-1A and Sentinel-1B, each equipped with a C-band SAR instrument. Sentinel-1 has a revisit time of six days if both satellites are operating, allowing near real time monitoring. Data products from the mission are freely available through Sentinel Hub's EOBrowser, the Copernicus Open Access Hub, or the Alaska Satellite Facility. Sentinel-1 data products provide the highest SAR resolutions that are open access and freely available [Sky21]. This makes Sentinel-1 a very common data source for research- and application specific purposes.

The satellites image with both stripmap and scanSAR configurations, and operate with both single- and dual polarized radar beams. The highest available resolution is 5 meters in both range- and azimuth, acquired by stripmap imaging. As for scanSAR the mission aims to cover a larger swath, and provides products from two scanSAR modes; Interferometric Wide Swath (IW) mode and Extra Wide Swath (EW) Mode. The swath widths for IW- and EW-modes are 250km and 400km, respectively. Availability of data products with specific acquisition parameters vary with region. For example, stripmap images with HH + HV polarization is only acquired for very specific regions.

Copernicus provides three different data products for each imaging mode. These products are a result of varying degrees of processing. Level-0 products are RAW, unfocused satellite data, and are generally not available to the public. Level-1 data products are focused and geo-referenced, and are available as either Single Look Complex (SLC) or Ground Range Detected (GRD) products. Level-1 SLC products preserve phase information in addition to amplitude, and are commonly used for radar interferometry applications. Level-1 GRD products have also been detected, multilooked and projected to ground range with a WGS84 Earth ellipsoid model. This process eliminates the phase information and provides a geo-referenced image with less speckle and noise, at the cost of lowered spatial resolution in the resulting image. Finally, Level-2 OCN products are available for specific needs over the oceans. For non-interferometry use, Level-1 GRD products are often the best option. These products are available in different resolutions as shown in Table 2.2 [ESAc].

Table 2.2: Level-1 GRD products from the Sentinel-1 mission are available in three different resolutions and all vary depending on image acquisition mode. Table from [ESAb].

	Imaging Mode	**Resolution**	**Pixel Spacing**
Full-resolution	Stripmap	9x9m	3.5x3.5m
High-resolution	Stripmap	23x23m	10x10m
	IW	20x22m	10x10m
	EW	50x50m	25x25m
Medium-resolution	Stripmap	84x84m	40x40m
	IW	88x87m	40x40m
	EW	93x87m	40x40m

2.4 Processing of SAR images

The processing steps needed to use a SAR image is heavily dependent on the application of the image and the level of the start product. Level-1 SAR products are most commonly used since Level-0 RAW data products are unfocused and compressed, requiring large amounts of processing to be usable. In this section the steps deemed necessary to use Level-1 GRD images to train deep neural networks to recognize water is explained. These steps can be used for SLC products as well.

2.4.1 Orbit Correction

Satellite orbit data is recorded by multiple sensors including gyros, GPS and ground observations. The recorded data can be used to calculate the precise orbit of the satellite. This calculation takes time and is not always included in SAR product download bundles, but may be uploaded at a later date. Orbit correction involves updating the metadata of the image by downloading and applying corrected orbit information, including satellite velocity and position. This may improve the accuracy of the image by improved geocoding and resolution.

2.4.2 Thermal Noise Removal

Amplitude images are affected by additive thermal noise. It is often visible as a color difference between sub-swaths in the image. This noise affects the signal to noise ratio and can be removed by normalizing the backscatter of the entire image. This step helps reduce the difference between swaths in multi-swath acquisition modes. Thermal noise exists in all electronic equipment and occurs due to the movement of charge carriers in conducting material. As an increase in temperature increases the amount of particle movement, the noise level increases with temperature. The noise power of thermal noise in the radar receiver is described by Equation (2.27) where N is the noise power, k_B is Boltzmanns constant, T_s is the system temperature, T_0 is the standard temperature of $290K$, F is the receiver subsystem noise figure and B is the receiver bandwidth in Hz.

$$N = k_B T_s B = k_B T_0 F B \tag{2.27}$$

For Sentinel-1 products, ESA provides noise vectors in look-up tables (LUTs). The denoising procedure is done by creating a denoising matrix η from the LUT as described in [ESA17], and applying the formula in Equation (2.28). Here $DN(i,j)$ is the digital number of the intensity value, $\eta(i,j)$ is the matching value from the denoising matrix. Negative values are set to zero.

$$DN(i,j)^2_{denoised} = |DN(i,j)|^2 - \eta(i,j) \ \ < \ 0 \tag{2.28}$$

2.4.3 Radiometric Calibration

Radiometric calibration must be applied for the pixel values of the image to be directly related to the radar backscatter of the scene. Each Level 1 product comes with four calibration LUTs. These can be used to produce the values for σ_i^0, β_i^0 and γ_i^0, or to return to the original Digital Number (DN). β^0 is the brightness coefficient and represents reflectivity per unit area in slant range. γ^0 is reflectivity in a plane perpendicular to the line of sight of the radar and an ellipsoidal model of the ground surface [Sma11]. The normalized radar cross section, σ^0, is a common measurement of the strength of a radar signal that is reflected by a target. It is related to ground range and is generally used to interpret surface properties [Arca]. The tables are used to apply a range dependent gain including the absolute calibration constant and an offset for the GRD products. The formula used to calculate the normalized radar cross section σ^0, and thus calibrate the image, is shown in Equation (2.29). DN_i is the digital number for the pixel and σ_i^0 is the matching value from the LUT.

$$\sigma^0(i) = \frac{{DN_i}^2}{(\sigma_i^0)^2} \tag{2.29}$$

2.4.4 Speckle Filtering

Speckle filtering is used to remove an artifact called speckle in radar images. Speckle occurs due to constructive and destructive interference of the electromagnetic waves scattered from multiple surfaces adding up in the image, which creates a granulated texture. Different filters can be used to reduce the amount of speckle. Some of the most common speckle filters are the Lee filter and Frost filter. Both of these filters are based on the assumption that the speckle noise can be modeled as multiplicative noise [OA10] that is independent of the backscattered signal. This is modeled as shown in Equation (2.30) where $I(t)$ is the image with speckle noise, $R(t)$ is the noise-free image or the radar backscatter and $v(t)$ is the independent speckle noise.

$$I(t) = R(t) \cdot v(t) \tag{2.30}$$

The Lee filter is based on minimum mean square error (MMSE) and works by using local statistics from a square window to calculate a new value for the center pixel [Arcb]. In more detail, this is done by computing the linear combination of the center pixel intensity in a filter window and an average intensity of the window. This is shown in Equation (2.31) where $\hat{R(t)}$ is an estimate of $R(t)$, $\bar{I}(t)$ is the mean of $I(t)$ and $W(t)$ is a weighting function given in Equation (2.32). Here C_v is the variance coefficient of the speckle noise and C_I is the variance of an affected pixel [San+16].

$$\hat{R(t)} = \bar{I}(t) + W(t)(I(t) - \bar{I}(t)) \tag{2.31}$$

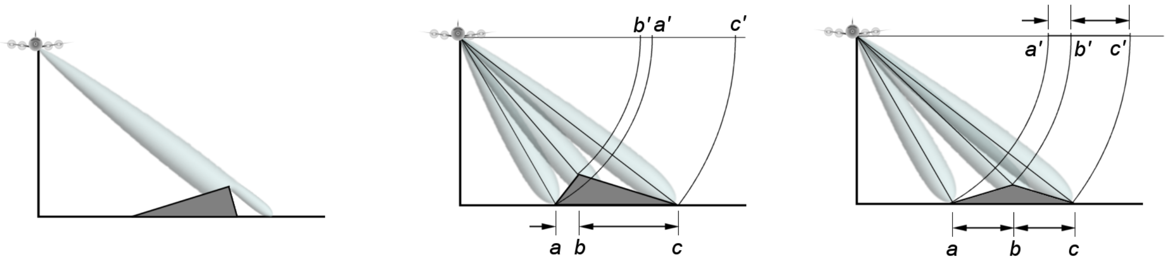
$$W(t) = 1 - \frac{C_v}{C_I} \tag{2.32}$$

The Frost filter is an adaptive filter that has the abilities of a mean filter in uniform areas and behaves like a high pass filter in areas with high contrast, thus preserving edges while smoothing speckle noise. The Frost filter, like the Lee filter, is based on MMSE. The filter response is adaptive and varies locally with C, which is the coefficient of variation. C is defined as the ratio of the local standard deviation, σ, to the local mean. The Frost filter is implemented based on the formula in Equation (2.33). Here k is a normalization constant, $\bar{I}$ is the local mean, $|t| = |X - X_0| + |Y - Y_0|$ and n is the size of the moving window [San+16].

$$DN = \sum_{nXn} k\alpha e^{\alpha|t|}, \quad \alpha = \frac{4}{nC^2} \cdot \frac{\sigma^2}{\bar{I}^2} \tag{2.33}$$

2.4.5 Terrain Correction

Since SAR-images are captured in slant range, effects such as shadowing, layover and foreshortening can occur. These effects are shown in Figure 2.18. The shadowing effect is shown in Figure 2.18a and occurs when the radars line of sight is obstructed by mountains and other tall objects, much like when the sun goes down in the evening. Layover is shown in Figure 2.18b. It occurs when the backscatter from the top of an object reaches the receiver before the backscatter from the bottom and as a result the top of the object will be misplaced toward the bottom and laid on top of that signal. The last effect is foreshortening shown in Figure 2.18c. This effect is a misrepresentation of distance and occurs when there is an incline towards the radar. As shown in the figure the distance between point a and b appear shorter due to the slant range view [Lil+19].



(a) shadowing

(b) layover

(c) foreshortening

Figure 2.18: Illustration of different effects that can occur in SAR-images due to slant range [Wol].

Terrain correction is a procedure that aims to correct these effects as they can lead to errors when geocoding the image. When applying terrain correction the image is combined with a digital elevation model (DEM) to shift the pixels such that the correct spatial relationship is achieved.

The terrain correction procedure starts by loading, reading and initializing the DEM, the state vectors for velocity and position of the satellite, and the look-up tables (LUTs). These steps are done to get ready for orthorectification, which is the process of converting the SAR-image into a cartographic reference system such as WGS84 EPSG:4326. The process of orthorectification starts by defining the corner coordinates of the area to be decoded. This is either decided by the user or the edges of the scene. Once there are coordinates that unambiguously defines the area to be geocoded the algorithm will loop through all points in the DEM. It starts moving through the latitudes in the DEM from a minimum to a maximum value and for each of these lines it will go through the longitudes from minimum to maximum. The raster of the DEM is indexed such that it matches the resolution of the SAR-image.

Orthorectification uses “backward geocoding” methodology, which means that it calculates the range and azimuth indices of the SAR-images from each point in the DEM. The process of deriving these indices is called geolocation.

The most accurate way to perform geolocation is to solve the Range-Doppler equations to determine the corresponding range and azimuth positions to a location $\vec{P} = (P_x, P_y, P_z)$ on the earth’s surface. The Range equation is shown in Equation (2.34) where $\vec{S}(t)$ is the position of the satellite and $R(t)$ is the range at time t. The Doppler equation is shown in Equation (2.35) where $f_D(t)$ is the Doppler frequency at time t, λ is the wavelength of the radar carrier frequency, $\vec{v_s}(t)$ is the velocity of the satellite, $\vec{v_p}$ is the velocity of the position P on the Earth’s surface and t is the azimuth time of the satellite [SS19].

$$R(t) = |\vec{P} - \vec{S}(t)| \tag{2.34}$$

$$f_D(t) = -\frac{2}{\lambda} \cdot (\vec{v_p} - \vec{v_s}(t)) \cdot \frac{\vec{P} - \vec{S}(t)}{R(t)} \tag{2.35}$$

The solution of the Doppler equation will give an azimuth time that satisfies the Doppler condition [SS19, p.26], making it possible to use Equation (2.36) to calculate the azimuth index. Here $I_a(t_D)$ is the azimuth image index for time t_D, t_D is the time satisfying the Doppler condition, and t_0 and δ_t is the azimuth start time and sample interval time and can be found in the scenes metadata.

$$I_a(t_D) = \frac{(t_D - t_0)}{\delta_t} \tag{2.36}$$

For slant range images the range index $I_r(P)$ for a point P is calculated using Equation (2.37), where $R(t_D)$ is the slant range from the range equation that satisfies the Doppler condition, r_0 and δ_r are near range and range pixel spacing and can be found in the metadata. For ground range images the range parameter $R(t_D)$ must be transformed to ground range. This procedure is described in this reference [SS19, p. 32].

$$I_r(P) = \frac{R(t_D) - r_0}{\delta_r} \tag{2.37}$$

When both the azimuth and range indices corresponding to the input radar geometry are calculated, the SAR-image can be re-sampled into the chosen map geometry. This is done using methods such as bilinear, clipped cubic convolution or nearest-neighbor re-sampling.

2.5 Pre-processing for Deep Learning

Processed SAR images from Sentinel-1 are commonly saved as rasters with GeoTIFF formats. This format encodes metadata alongside the image. To fully prepare the images for Deep Learning algorithms, a series of pre-processing steps can be implemented to improve model performance and reduce computational complexity. These pre-processing steps include contrast stretching, normalization, tiling and data augmentation.

Contrast stretching aims to improve contrast between individual pixels by spreading the pixel values over a desired range. In the case of landcover segmentation, enhanced contrast will often improve the ability to distinguish landcover types with similar characteristics. One common contrast stretching method is *histogram stretch*, which improves contrast without producing unnatural looking images due to exaggerated alterations. Histogram stretch creates a histogram of all pixel values, and stretches the image linearly within a chosen range $[p_{\min}, p_{\max}]$, clipping all values outside this range to $p_{\min}$ or $p_{\max}$. In SAR images, some pixels have unrealistically high values due to constructive interference of backscatter and incidence angles. It is therefore beneficial to choose $p_{\min}$ and $p_{\max}$ to reduce the effects of such pixels on image contrast and quality. The new pixel value, p_{out}, is given by Equation (2.38), where a and b are the desired lower and upper bounds of the resulting image, respectively [Fis+03]. The image is thus stretched within $[p_{\min}, p_{\max}]$ and normalized to $[a, b]$. For images to be compatible with most deep learning algorithms, the normalization interval is typically chosen so $a = 0$ and $b = 1$.

$$p_{\text{out}} = (p_{in} - p_{\min}) \left(\frac{b-a}{p_{\max} - p_{\min}} \right) + a \tag{2.38}$$

As mentioned in Section 2.3.5, high resolution IW-GRD products from Sentinel-1 cover large regions with a pixel size of 10x10m. Training deep networks with images of that size requires large amounts of available hardware memory. Tiling the input image into several smaller images is commonly performed before training to reduce memory requirements. An additional benefit of tiling is increased training speed, which allows more frequent testing in time-limited projects. Reviewed research conducted before this thesis project suggests a possibility of more accurate segmentation results when utilizing larger tiles, and that tile size should be chosen as large as possible with given time and hardware limitations [VA21, p. 12].

Deep learning models need to generalize well to be useful. In terms of remote sensing this means the model should be able to accurately predict data from multiple regions. To achieve a generalized model, a large and varied dataset is preferable to train with. Although satellite data exists in abundance, captured images over specific regions do not show a lot of variance since changes in landcover are mostly caused by slow processes that occur over multiple decades. There is also a lack of accurately labeled data, which currently consists of landcover maps that do not adapt to recent changes in landcover. As such, data augmentation may be applied to artificially increase the variance within the dataset [Tom20]. With greater variance, the model will not overfit (ie. memorize training data), leading to better results on new, unseen data. Data augmentation increases variance by transforming images before training, and may include image rotation, translation, cropping, flipping, etc. In case of training with a smaller dataset, augmentation may be used to increase the size of the dataset by keeping the original images and adding the augmented images.

Chapter 3

Methodology

This chapter presents the implementation methods utilized in this thesis project. All methods are implemented in Python, and readers interested in the complete code-base are referred to `https://github.com/masteroppgave2022/Project`. The chapter first covers data acquisition and processing to give an understanding of how the datasets used to train and test the models are produced. Model implementation and testing is then presented, including the chosen set of evaluation metrics. The final section briefly summarizes how the code is organized to aid those interested in the Python implementation. All implementation methods are based on the theory presented in Chapter 2, and produced results are documented and discussed in Chapters 4 and 5, respectively.

3.1 Dataset

Multiple datasets are produced from SAR-images captured by Sentinel-1 as part of ESA's Copernicus program. The method of acquisition and image processing is the same for all datasets, but the datasets are of varying sizes with slight differences in imaged regions. Multiple datasets enables the possibility of training on smaller datasets when testing small changes to save time. Masks are processed from inland water labels acquired from The Norwegian Mapping Authority as part of the Geovekst initiative [KG22]. This section elaborates the process of acquiring images and labels, and the processing of these to produce complete datasets ready for training segmentation models. Appendix A provides additional details on the specific images acquired from Sentinel-1.

3.1.1 Data Acquisition

SAR images from Sentinel-1 are accessed and downloaded through the Alaskan Satellite Facility's (ASF) Search API. The API is accessed using the `asf_search` Python wrapper [Adm21]. Images are requested with a set of search parameters as shown in Table 3.1. Requested images are checked for duplicates and appended to a download queue, and the API download method is invoked on the queue. This implementation allows parallel downloads of large amounts of SAR images by specifying search parameters in separate configuration files.

Table 3.1: Search parameters specified when requesting SAR images from ASF. The right column contains the possible values of each parameter.

Parameter	**Possible Values**
Platform	Sentinel-1A/1B/2A/2B/3
Processing level	SLC, GRDH, GRDM, OCN
Region-of-Interest	Coordinate bounding box
Beam mode	SM, IW, EW
Polarization	HH, HV, VH, VV, VV+VH, HH+HV
Flight direction	Ascending, Descending
Start	Valid date
End	Valid date
Max results	Positive integer

Available acquisition parameters and resolutions are predetermined by the Sentinel-1 mission, as elaborated on in Section 2.3.5. Over central parts of Norway, Sentinel-1 captures images in IW-mode with VV+VH polarized waves. The products of interest are GRDH-images. Flight direction should not affect the ability of models to detect water, so this variable is set to "None" to include both ascending and descending directions. Start and end dates, and maximum results, is set to acquire a time series of images within certain periods. Maximum results is also altered to regulate the size of the produced dataset. SAR-products that are requested with these parameters have the file format *.SAFE*, which stands for "Standard Archive Format for Europe". A product includes XML files containing metadata, some preview images, as well as the measured and generated SAR-image bands. Images captured using multiple polarizations have one band per polarization.

The API requires the region of interest in WKT-string format, which is extracted from shapefiles created with Geographic Information System (GIS) software. For the purpose of this project, the 25 regions shown in Figure 3.1 are chosen to constitute the dataset. Acquired images consist of the entire scene in which the Region-of-Interest (ROI) is contained, with a spatial resolution of 20x22m and pixel size of 10x10m. Multiple images are acquired of each ROI for different periods of time. Additionally, Sentinel-1 scenes containing one ROI will likely also contain neighboring ROIs. This limits the number of scenes that need to be acquired to produce a dataset of desired size. This is efficient to limit the amount of necessary downloads.

All regions are chosen based on the landcover characteristics they contain. To mitigate the effects of layover and shadowing in the images, very steep mountainous areas are avoided. For the purpose of mapping water extents it is also beneficial that the images contain sufficient amounts of various water bodies. The chosen regions therefore include both larger rivers and lakes, and the proportion of water to dry land is attempted maximized without generating shapefiles for each water body within a region. This is done to ensure a balance of dry land and water features for the model to train on, which is previously found to yield performance gains in segmentation models [VA21, p. 30]. Finally, image labels are available by county, and regions are thus chosen within three Norwegian counties to reduce the total amount of data to be acquired.

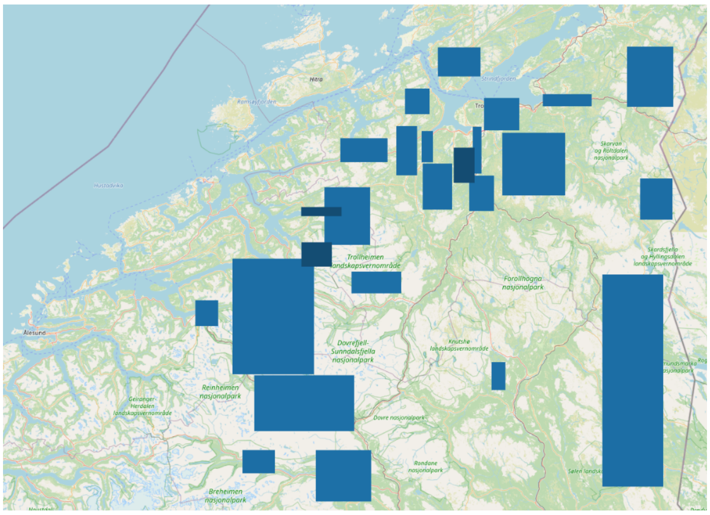

Hitra
Strindfjorden
Ramsøyfjorden
Hustadvika
Skarvan og Roltdalen nasjonalpark
Trollheimen landskapsvernområde
Forollhogna nasjonalpark
Skardsfjella og Hyllingsdalen landskapsvernområde
Ålesund
Dovrefjell-Sunndalsfjella nasjonalpark
Knutshø landskapsvernområde
Reinheimen nasjonalpark
Geiranger-Herdalen landskapsvernområde
Dovre nasjonalpark
Rondane nasjonalpark
Breheimen nasjonalpark

Figure 3.1: Polygon shapefiles showing regions of interest in central Norway. Datasets are produced with images of these regions.

Unprocessed water labels are acquired from The Norwegian Mapping Authority, who provides the “FKB-Vann” dataset which consists of geo-referenced water shapefiles. The dataset is produced by manual analysis of high resolution aerial imagery, and the resulting shapefiles have a resolution of 7cm to 25cm, and an uncertainty of 0.1 to 1.0m depending on location [Geo22]. “FKB-Vann” is requested from GeoNorge [KG22] by county and downloaded when the request is approved. Downloaded data consists of one shapefile for each of the three requested counties: Innlandet, Trøndelag, and Møre og Romsdal, which are merged to one shapefile as shown in Figure 3.2. The final shapefile consists of polygons highlighting water bodies in central Norway, and is used to produce masks for the ROIs.

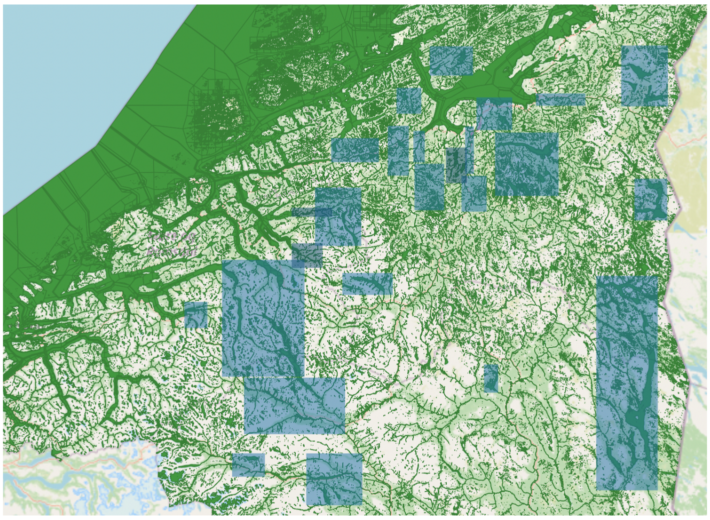
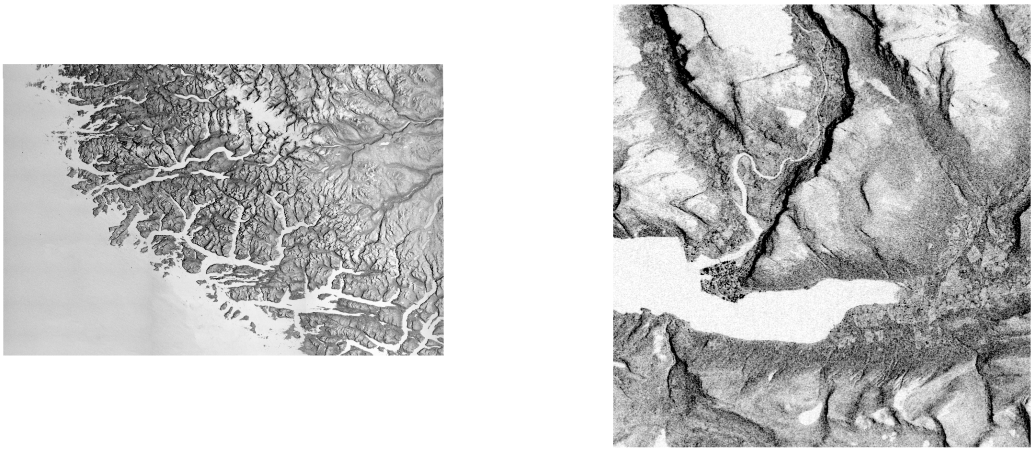


Figure 3.2: Shapefile containing polygons of water bodies in central Norway in green, with ROIs overlayed in blue.

3.1.2 Data Processing

Acquired images are processed to achieve accurate representations of the captured SAR-data that corresponds with the regional landcover. Furthermore, the water labels from “FKB-Vann” are processed to generate masks for each image. The resulting images and masks from these processes are further processed to produce tiled datasets suitable for training machine learning models. All correctional processing of the images from Sentinel-1 is implemented using the `snappy` library, which is the Python adaptation of the Snap software provided by the European Space Agency [ESA20]. Snap [ESAd] provides a toolbox specifically available to process SAR-imagery obtained from Sentinel-1. All processing of masks, as described here, is implemented using the Python `GDAL` and `RasterIO` raster processing libraries.

In raw format the downloaded scenes are very large, occupying anywhere between 1.5 and 5 gigabytes of hard drive space each. To reduce processing times the scenes are cropped to the ROIs presented in Section 3.1.1 before processing, an example is shown in Figure 3.3. Each resulting subset is then processed in a series of steps, as shown in Figure 3.4.



(a) A full scene from Sentinel-1 covering parts of Norway.

(b) A subset of the full scene.

Figure 3.3: One full scene and a subset of given scene.

The metadata is updated from an accurate orbit file downloaded from ESA to ensure accurate position of the satellite and scene. This step is necessary to ensure all subsequent processing steps involving these parameters are accurate. Thermal noise is then removed by Snap’s integrated thermal noise removal method, which automatically accesses and subtracts noise vectors contained in the metadata. The subset image is then radiometrically calibrated, transforming the measured backscatter into values of normalized radar cross-section, σ_0. Calibration is the single most important processing step as cross-section is a measure of influence of the target on the wave received by the radar, and thus gives information about target properties. The calibrated image is speckle filtered with a Frost filter of size 5x5px, to remove unwanted artifacts caused by interference patterns. Frost is preferred to the Lee filter due to its edge preserving characteristics, which is beneficial when distinguishing water edges from non-water edges. Finally, the subset is geometrically corrected and accurately geo-referenced by terrain correction with a Digital Elevation Model (DEM). The utilized DEM is acquired from The Norwegian Mapping Authority with a resolution and pixel spacing of 10x10 meters. This DEM is chosen due to its high resolution as more common global DEMs that are freely available are limited to a 30x30 meter resolution at the latitudes of interest.

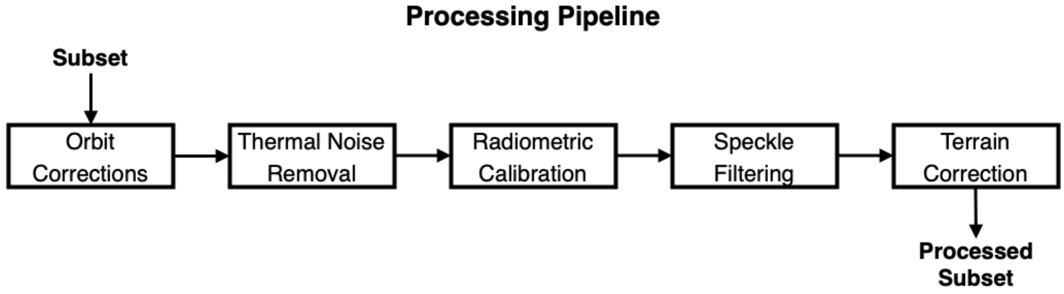
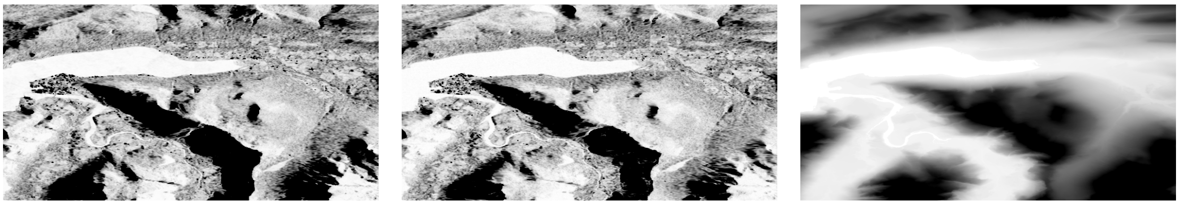


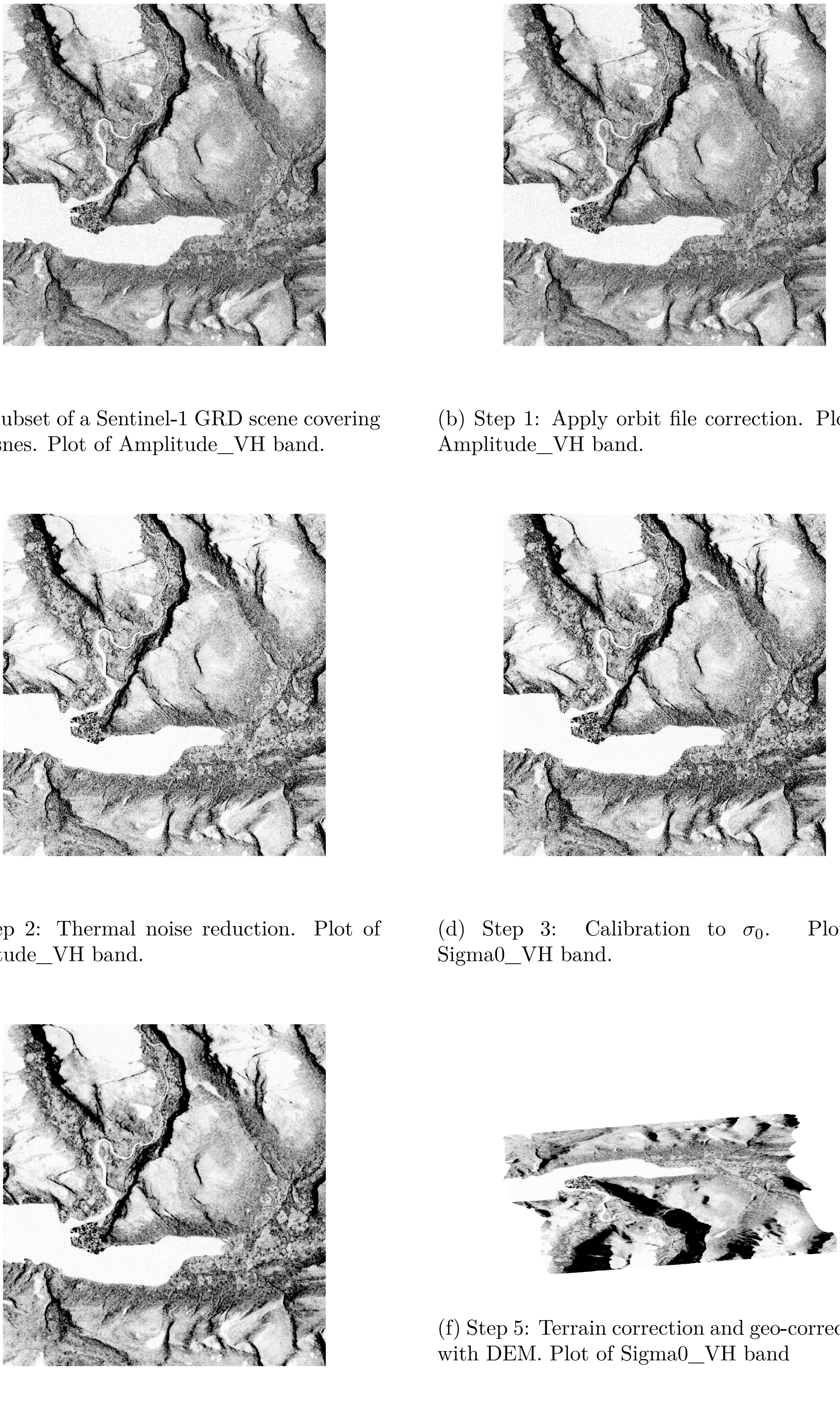
Figure 3.4: Each image subset is processed to produce a high-quality geo-referenced image with suppressed noise and speckle artifacts.

The processed images consist of two SAR-image bands, one for each polarization (VV and VH), and each showing the normalized backscatter the radar has obtained from the scene. The DEM of the same region is appended as a third band. This is beneficial since most existing Deep Learning algorithms are optimized to train with optical images with three bands (Red, Green, Blue), negating the need to modify or merge input bands before training. Antropov et al. [Ant+21] empirically proves that using a DEM as a third band results in a model that performs better than when using a combination of the two existing bands to make a third band. The three bands the processed images consist of are shown in Figure 3.5. Figure 3.6 shows the VH polarized band of the same image during processing, illustrating how the processing steps affect the contents. Although it is hard to distinguish the first five processing steps by visual inspection, noise and speckle is suppressed in Figure 3.6c and Figure 3.6e. Calibration, shown in Figure 3.6d, ensures the image represents the radar cross section, σ^0, rather than the backscatter intensity values received by the radar. Careful examination of the images shows improved contrast and less speckle throughout the first 5 processing steps. The most notable visual change occurs after terrain correction since the image is not correctly geo-referenced prior to this step. The processed three band image is saved in GeoTIFF format.



(a) Processed VV band. (b) Processed VH band. (c) Elevation band.

Figure 3.5: All bands given to the segmentation model for training.



(a) A subset of a Sentinel-1 GRD scene covering Andalsnes. Plot of Amplitude_VH band.

(b) Step 1: Apply orbit file correction. Plot of Amplitude_VH band.

(c) Step 2: Thermal noise reduction. Plot of Amplitude_VH band.

(d) Step 3: Calibration to σ_0. Plot of Sigma0_VH band.

(e) Step 4: Speckle filtering with Frost-filter. Plot of Sigma0_VH band.

(f) Step 5: Terrain correction and geo-correction with DEM. Plot of Sigma0_VH band

Figure 3.6: Processing steps performed on the dataset.

As the time and resources required to hand-label the data is not available, the chosen compromise is to produce masks from the acquired "FKB-Vann" shapefile. The shapefile is first re-projected to the chosen project CRS (EPSG:4326) and map projection (WGS84) using GIS-software. The shapefile is then cropped to the ROIs to obtain shapefiles covering the extents of the processed images. These shapefiles are then converted to masks with pixel value 1 corresponding with water, and pixel value 0 corresponding with everything else. The pixel size of the mask is fetched from a processed image of the same region to ensure the generated mask is a pixel-perfect match to its corresponding image. After the conversion the mask is saved as a GeoTIFF raster. Figure 3.7 shows each of the processing steps implemented to produce a mask for one of the ROIs.

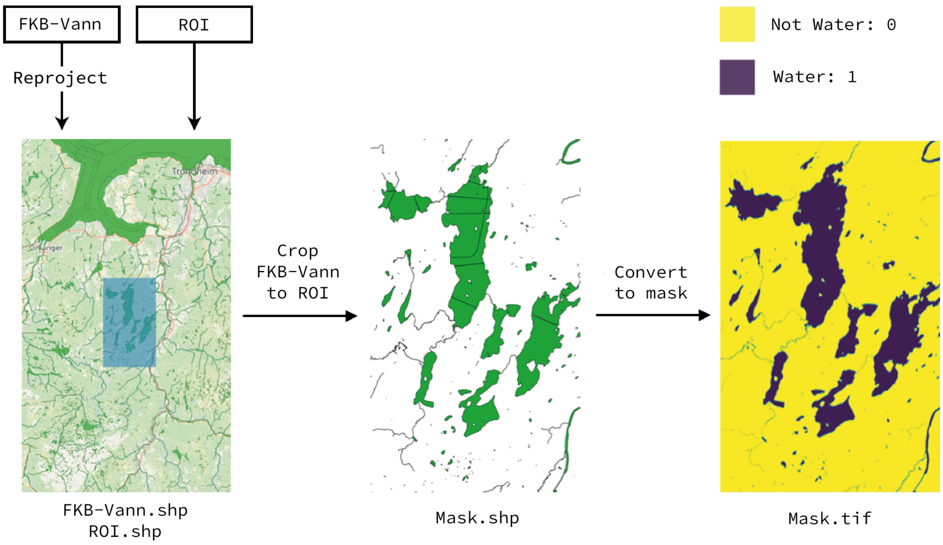


Figure 3.7: Process of generating masks from the "FKB-Vann" dataset provided by The Norwegian Mapping Authority.

3.1.3 Building a dataset

Processed images and corresponding masks are tiled into 256x256 pixel blocks. This size is chosen to maximize the amount of content available in each tile without limiting the desirable batch size loaded into memory during model training. Larger tiles with lower batch sizes may also prolong the time it takes to train various models, which is not prioritized given the thesis project's time constraints. All image and mask tiles are shuffled to a random order and split into training and validation data. The chosen validation split is 85%-15%, such that 85% of the tiles are used for training and 15% for validation. It is also desirable to test the models on unseen data. A number of regions are therefore withheld from the training and validation datasets. Images from these regions constitute a test set consisting of full sized images along with their tiles. The purpose of this is to compare model performance on full-sized images to that of tiled images. In total, a large dataset is produced with the specifications shown in Table 3.2. A subset of this dataset is copied to a smaller dataset as shown in Table 3.3. The intended purpose of the smaller dataset is to enable quicker training to test implemented changes to the models before training with the larger dataset to save time.

Before the dataset is passed to the model, all the image tiles are contrast enhanced by

Table 3.2: Specifications of the generated dataset for model training and testing.

Main Dataset			
Dataset	**Number of Images**	**Image Size**	**Water percentage**
Train	100 933	256x256px	10.83 %
Validation	17 812	256x256px	10.79 %
Test	258	Full size	7.85 %
Test	14 296	256x256px	7.85 %

Table 3.3: Subset of the main dataset. This enables faster model training when testing smaller changes to model parameters before training on the large dataset.

Subset Dataset		
Dataset	**Number of Images**	**Image Size**
Train	17 000	256x256px
Validation	3 000	256x256px

a simple percentile based histogram stretch, presented in Section 2.5. Since SAR-images often contain some unnaturally high or low pixel values, all images are scaled and stretched between the 2nd and 98th percentiles contained in the pixel value histogram of each image. This suppresses outliers, as well as providing a slight increase in image contrast. The images are then normalized to the interval [0,1]. Since all tiles are generated from multiple images of the ROIs, it is highly likely that many of the tiles are very similar. To avoid the model memorizing these repeating patterns, images and corresponding masks are randomly augmented to synthetically increase the variation in the dataset. The augmentations and their frequencies are shown in Table 3.4, and have been implemented with a Python library called `Albumentations` [Bus+20].

Table 3.4: Augmentations used on the training data to increase data variance and prevent models overfitting. The probability of an image being subject to an augmentation is also given.

Augmentation	**Probability**
Random Rotate 90 degrees	0.5
Channel Shuffle	0.2
Horizontal Flip	0.5
Vertical Flip	0.5

3.2 Segmentation

This section aims to describe model implementation, the training process, and the process of evaluating model performance. The implementation of the chosen model architectures is described first, followed by the training environment and methods of training. Finally, chosen methods of model evaluation is briefly presented. The section does not include in-depth theory and diagrams relating to model architectures, as this is covered in Chapter 2, but briefly summarizes development choices and the general workflow used during implementation.

3.2.1 Models

This thesis project implements two model architectures, U-Net and DeepLabV3+. Both architectures are based on existing implementations. This subsection briefly describes the source of the architectures and modifications made to the models to accommodate the intended purpose of the project. It also gives a brief, general background as to why these specific architectures are chosen. All models and related methods are written using the `Tensorflow` 2.7.0 and Keras APIs.

U-Net with ResNet50 Backbone

The U-Net architecture was first proposed for segmentation of medical images and is designed to learn from few training samples. It is familiar from previous work, which shows that U-Net is capable of highly accurate image segmentation. The architecture is described in detail in Section 2.2.2, which shows frequent concatenations of encoder layers to decoder layers. This trait implies that U-Net constantly considers low level features, such as lines and shapes, extracted by the encoder, making U-Net an exceptional edge detector. This characteristic is beneficial for water detection since radar images show distinct edges separating water and land. To maximize the potential performance of U-Net, the high performing ResNet50 image classification architecture is the chosen encoder backbone. The decoder of the U-Net is a deconstruction of the ResNet50 encoder, mirroring all layers.

This thesis project utilizes the Tensorflow Keras implementation of U-Net provided by the `segmentation_models` library [Yak19]. The model is pre-configured to train with input shape (None, None, 3), and is thus flexible in terms of spatial dimensions, but expects three channels. A necessary benefit of this is the ability to train with images of any shape, which enables testing on datasets with varying image sizes. The project time span also favors the use of transfer learning, and the ResNet50 encoder is therefore initialized with weights that are pre-trained on the ImageNet dataset. The decoder weights are randomly initialized, and the last layer uses softmax activation. The output of the network is an image with height and width equal to the input image, but the softmax activation function implies two output channels, one for each class.

DeepLabV3+ with Xception Backbone

The proposed U-Net model performs well with distinct edges separating the various classes. In the specific case of SAR imagery, water exhibits vastly different scattering characteristics than other landcover types, which clearly separates it from other classes. Despite this, the resolution of the acquired images is limited, which leads to a misrepresentation of edges along slightly narrower parts of the larger rivers. Since rivers in Norway commonly reside in valleys and show similar characteristics, it is thought to be beneficial to train a second architecture that is able to infer based on image context. The chosen architecture is Google's DeepLabV3+, described in Section 2.2.3, and the chosen encoder backbone is Xception.

The DeepLabV3+ architecture is cloned from Zakirov's implementation [Zak21]. Some modifications are made to this implementation to accommodate the intended use case. Firstly, the backbone of interest is Xception, and all code relating to other backbone architectures is removed to simplify the implementation. The architecture is then modified to enable training and testing with images of varying sizes, as the original implementation needs a specific image shape to compile. This modification is necessary to allow model inference on the test set, which consists of images of varying sizes. The network is configured to receive three channel images as input. The output is equal to the input in spatial dimensions, but with two channels. Each channel corresponds with a class, either "water" or "not water", respectively. The weights of the encoder are pre-trained on the Cityscapes dataset, while the decoder weights are randomly initialized, and the chosen activation in the last layer is the softmax function.

3.2.2 Model Training

All models are trained on a Linux server running Ubuntu 20.04. The server runs an AMD Ryzen Threadripper 3960X 24-Core Processor, and an NVIDIA GeForce RTX 3090 GPU with 24 GB memory. All code implementations are hosted on this server, and the server is accessed by Secure Shell Protocol (SSH). Visual Studio Code is the editor used to remotely edit and test the implemented code. Cuda version 11.4 is used to fully leverage the GPU acceleration available on the server.

Prior to training the models on the main dataset, test runs are conducted on the smaller dataset. These models are trained to quickly test training parameters such as batch size, learning rate, and optimizer. The factors that are considered during testing are resource limitations, training time, model performance and project duration. Results acquired from these models set the baseline for the parameters used to train models on the main dataset. As such, empirical evidence serves as the method of parameter tuning during this thesis. Test runs indicate that model performance may perhaps be improved by lowering the batch size from the chosen size of 32 used here. This is not deemed necessary since the performance gains are minimal while training time is nearly doubled. Larger batch sizes are not experimented with since both implemented architectures require so much memory that even a batch size of 64 is impossible when limited to 24 GB of GPU memory. The learning rate is chosen as small as possible while still showing satisfactory convergence of the loss function, based on the test runs, and the Adam optimizer is chosen since it shows satisfactory convergence and training speed on the smaller dataset.

One primary model is trained for each of the chosen architectures, U-Net and DeepLabV3+. Each model is trained for 30 + 15 epochs, using the Adam optimizer with a DICE loss function. In addition to DICE, accuracy is tracked during training. The first 30 epochs are trained with the pre-trained encoder frozen, and the last 15 epochs with all layers unfrozen. An initial learning rate of 10^{-4} is used while the encoder is frozen. To increase the likelihood of reaching an actual minimum when training gets deeper, the learning rate is reduced to 10^{-5} for the last 15 epochs. In accordance with the test runs on the smaller dataset, the chosen batch size is 32.

After testing the two primary models, the DeepLabV3+ model does not achieve the expected performance. Thus, an additional DeepLabV3+ model is trained without freezing the encoder weights, simply fine tuning the pre-trained encoder throughout all epochs. This further tunes DeepLabV3+ to the dataset without having to train deeper. Learning rate, optimizer and batch size is not altered for this model.

Binary classification with a deep neural network gives a choice of two classification approaches in terms of network output. With regards to this project, the most efficient approach would be to have a single network output with the probabilities of samples belonging to the "water" class. By setting a probability threshold, all values above this threshold are classified as "water", while all values below are not. Another option is to have two network outputs, where each output calculates the probability of "water" and "not water", respectively. The models simply classify samples into the class with the highest probability. Despite being slightly less efficient, this approach is chosen to avoid having to fine tune a threshold to achieve the best possible results. It is worth noting that an output layer with softmax activation will give a number of outputs that correspond with the number of classes, as is used here. Realization of the first approach is commonly done with other activation functions, such as the Sigmoid function.

3.2.3 Evaluation

The three models are evaluated on a test set consisting of full-sized images of regions the models have not seen before. To examine the impact of image size on model performance, the models are also tested on a test set consisting of 256x256-tiles of the same images. Predictions made by the model on these tiles are stitched together to their original images, allowing easier visual inspection of the results. Some typical machine learning metrics such as precision, recall and F1-score are calculated to indicate the performance of the model. In addition, some more segmentation oriented metrics are calculated. These include pixel accuracy and the Jaccard index, as covered in Section 2.1.3. Since the primary purpose is to detect water against all other landcover classes, the Jaccard index is calculated per class, specifically indicating model accuracy on the water class. PRCs are plotted for the predicted probabilities of pixels belonging to the water class, and the AUPRC is calculated as a performance measure of a model's ability to detect water. This metric does not consider the predicted probability of the pixel not being part of the water class.

3.3 Code Structure

The implemented methods and modules are available at the thesis GitHub page, `https://github.com/masteroppgave2022/Project`. All of the chosen solutions are implemented in Python, except for the acquisition of unprocessed image labels and creation of shapefiles covering the ROIs. The code consists of four modules, with one `Main` module that acts as a control unit. Users specify paths to datasets and module configurations in a configuration file that is read by `Main`. This file ultimately decides which of the four modules to run, which parts of these to skip, as well as providing necessary user information for the requested modules. Figure 3.8 is a high-level block diagram that shows how the code is organized.

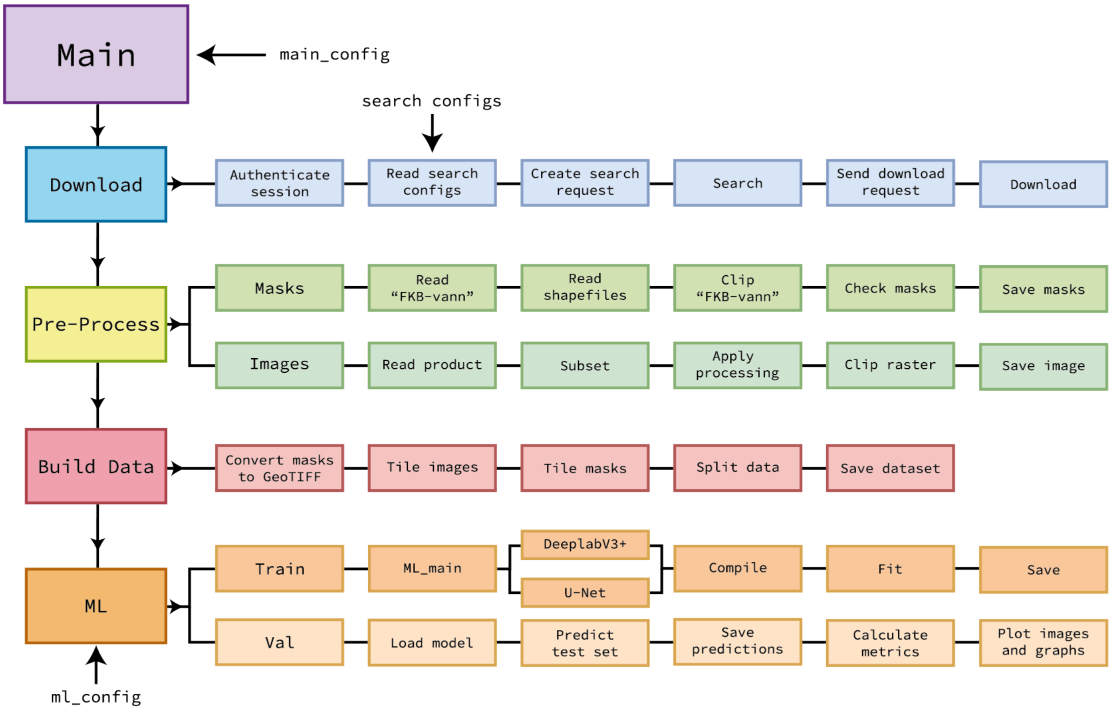


Figure 3.8: Flowchart showing how the code-base is organized. The implementation consists of four modules; `Download`, `Pre-Process`, `Build Data`, `Machine Learning (ML)`, and one `Main` module that controls these modules.

The `Download`-module is tasked with requesting and acquiring Sentinel-1 images from ASF. Credentials required to authenticate a session at ASF is provided by the `main_config` in `Main`. Desired search parameters are provided by users in configuration files, referenced as `search_configs`, one for each ROI and time period. The search returns images that satisfy the parameters, and the download location specified by the user is checked for duplicates before all search results are downloaded. The downloaded results are fully sized Sentinel-1 scenes with minimal processing performed by ESA, and are saved to the specified download location as a compressed .zip-file.

`Pre-Process` accesses the downloaded images and applies the processing pipeline to all images in the folder provided by `Main` and the `Download`-process. It iterates through all scenes checking for overlap with all ROIs, and saves a subset of the scene for each ROI that is fully contained within the scene. Each of the subsets are processed through the pipeline presented in Figure 3.4. Before the subsets are saved as GeoTIFF files, they are again cropped to ensure that they accurately describe the same area as the shapefiles. In a separate process, the masks from "FKB-Vann" are cropped according to the same shapefiles as the SAR data. They are then checked and cleaned for loose points or lines not included in any river or water body polygon, before they are saved as shapefiles matching each ROI.

All datasets are produced by `Build Data`, which parses the processed data provided by `Pre-Process`. This module takes the provided data and turns it into a finished dataset. The first step is to convert the masks from shapefiles to rasters, and these are saved as GeoTIFF files. This function uses `GDAL` to sample the masks to the same resolution as their corresponding SAR images, and gives all water labels a pixel value of 1. All images and masks are then tiled into squares of 256x256px. The module then shuffles the data and splits it into a train and a validation set. The train-validation split ratio is given by `main_config` in `Main`. Which regions to use for testing is also given in the configuration file. Images and masks of these regions are both saved as tiles and fully sized images in a separate test image folder.

Model training and testing is handled by the `ML` module. Everything relating to the machine learning part of the thesis is in this module. The chosen architecture is specified by `Main`, and users specify training parameters and datasets in a configuration file defined as `ml_config`, read by `ML`. Everything in the `Train` sub-module is handled by an `ML_main`-function, which accesses all necessary objects and methods used to train the models. A custom data generator is created such that images and masks are opened correctly. Here, the images are contrast enhanced by histogram stretch, and the masks are one-hot encoded before both are augmented and fed to the Tensorflow "fit" function. In addition, a DICE loss function is written as an extension to the "tensorflow.keras.losses.loss" class, as Keras does not have an implementation of said function. In the `Val` sub-module of the `ML` module, the images from the test and validation datasets are predicted, metrics are calculated and different figures and graphs are plotted. Separate Jupyter Notebooks are also created to enable testing without having to reinitialize variables for every run.

Chapter 4

Results

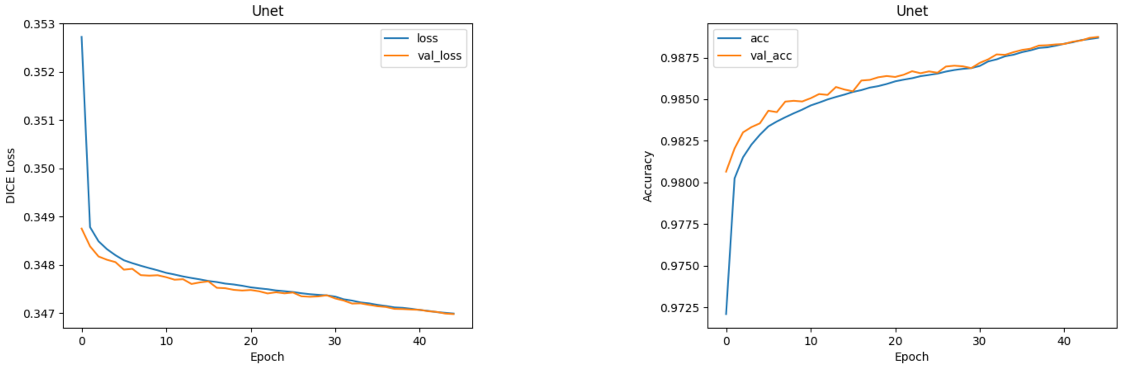
This chapter presents all results obtained by the implemented models on the various datasets. Training metrics are presented to give an understanding of model convergence and behavior during the training phase. Metrics used to measure model performance on validation data and unseen test data is given. Furthermore, some predicted images are shown to give a visual indication of performance, including some images of regions with temporally varying water extents. The latter aims to indicate model performance in regards to detecting changes in water bodies and rivers.

4.1 Training Performance

This sections presents the training results acquired by the best performing U-Net and DeepLabV3+ architectures. The results are presented as loss and accuracy plots, along with an accompanying description that further elaborates on the training performance of the models.

Figure 4.1 presents the results from training a U-Net architecture with a ResNet50 backbone for image segmentation. Figure 4.1a shows DICE loss per epoch. One can see that there is a drop in loss between the first and second epochs before it slowly decreases through the rest of the epochs. Even though the graph shows a reduction in the DICE loss, the validation loss, referenced as val_loss, is in the interval $[0.348, 0.346]$ for all epochs. Figure 4.1b shows the accuracy per epoch during training. The graph of the accuracy mirrors the DICE loss as it begins with a large step from the first to the second epoch and then a slow increase while keeping the validation accuracy in the interval $[0.9806, 0.9887]$.

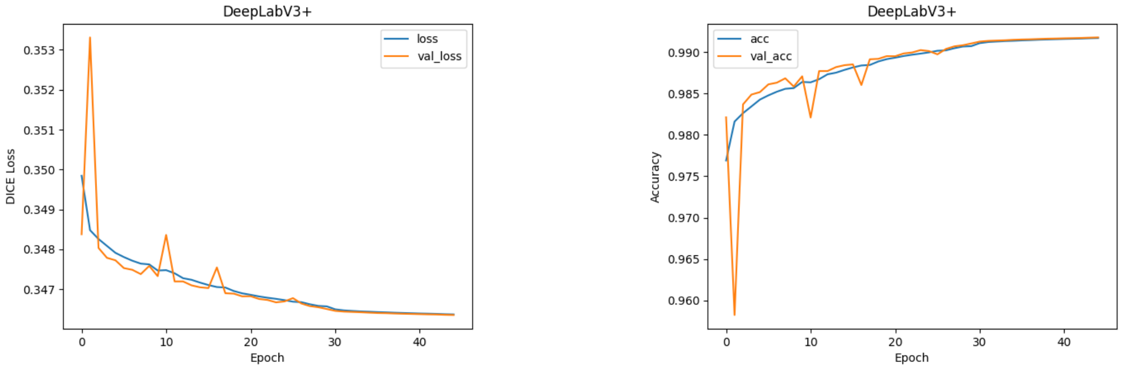
In Figure 4.2, the results from training a DeepLabV3+ architecture with an Xception backbone is presented. The graphs from the DeepLabV3+ training resembles those of the U-Net architecture. Figure 4.2a shows DICE loss per epoch. The loss has a few spikes, but slowly decreases and converges to 0.346 during the last 15 epochs. Figure 4.2b shows a graph of the accuracy. Similar to the loss graph, it has a few drops before it converges to 0.9917.



(a) DICE loss per epoch.

(b) Accuracy per epoch.

Figure 4.1: U-Net training results.



(a) DICE loss per epoch.

(b) Accuracy per epoch.

Figure 4.2: DeepLabV3+ training results.

4.2 Model Performance

Both of the implemented and trained model architectures are tested on a validation dataset and a test dataset. The tested U-Net model is trained with pre-trained encoder weights from the ImageNet dataset. The first 30 epochs are trained with frozen encoder weights, while 15 further epochs are trained with no frozen weights. Two DeepLabV3+ models with pre-trained weights from the Cityscapes dataset are tested, one with frozen encoder weights for the first 30 epochs, and one with no frozen weights. The numerical and visual results obtained by these models on the validation and test sets are given here.

4.2.1 Performance Metrics

Model performance is evaluated with the performance metrics presented in Section 2.1.3. The primary evaluation metric is the class-based Jaccard index, which indicates similarity between the predicted class and true class. A global, class weighted Jaccard index is also calculated to give a single number indicating performance. Table 4.1 shows the performance of the three models on the validation dataset. DeepLabV3+ trained without frozen layers achieves the highest scores in all categories, with a Jaccard index of 0.929 on the water class, and a water pixel accuracy of 0.945. U-Net is the best performing model partially trained with frozen encoder weights, with a water class Jaccard index of 0.903 and water pixel accuracy of 0.924.

Table 4.1: Model performance on validation data.

Model	**Weighted Jaccard**	**Jaccard by Class**	**Precision**	**Recall**	**F1**	**Water Accuracy**
U-Net	0.979	0.903/0.988	0.989	0.989	0.989	0.924
DeepLabV3+*	0.985	**0.929**/0.991	0.992	0.992	0.992	**0.945**
DeepLabV3+	0.972	0.871/0.984	0.985	0.986	0.986	0.895

* Trained with pre-trained weights NOT frozen during first 30 epochs.

The models are tested on unseen test data to give an indication of how well the models generalize. Table 4.2 shows the calculated performance metrics for model inference on test data that is tiled to the same size as the training and validation data. DeepLabV3+ trained without frozen encoder weights achieves the highest scores with a Jaccard index of 0.856 on the water class, correctly predicting 89.2% of the labeled water pixels. As for the models partially trained with frozen encoder weights, U-Net performs the best with a water class Jaccard index of 0.846, and 88.6% accuracy in predicting water. Class weighted precision, recall and F1-scores are also very high with little variance, ranging from 0.986 to 0.988 for all models.

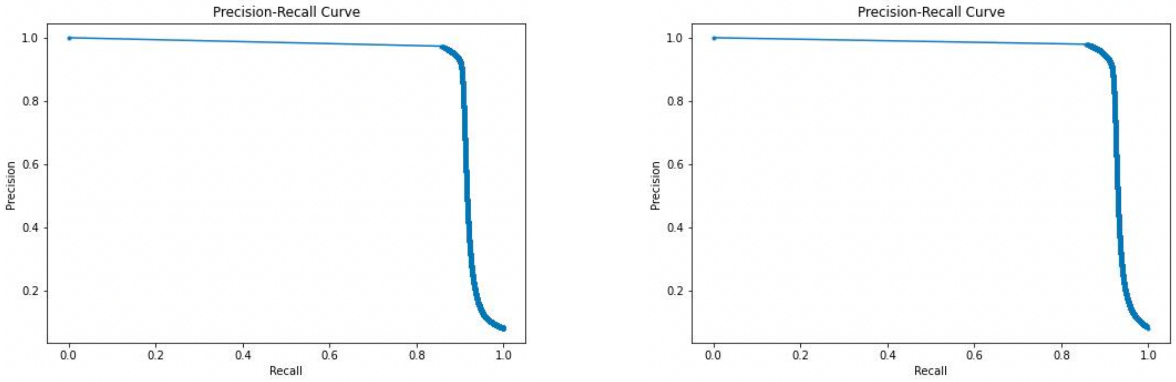
Table 4.2: Model performance on unseen test data consisting of 256x256px size images.

Model	**Weighted Jaccard**	**Jaccard by Class**	**Precision**	**Recall**	**F1**	**Water Accuracy**
U-Net	0.974	0.846/0.986	0.987	0.987	0.987	0.886
DeepLabV3+*	0.976	**0.856**/0.987	0.988	0.988	0.988	**0.892**
DeepLabV3+	0.974	0.842/0.986	0.986	0.987	0.986	0.881

* Trained with pre-trained weights NOT frozen during first 30 epochs.

Figure 4.3a shows the PRC achieved by U-Net when tested on 256x256px tiles. This curve only considers the predicted probabilities of samples belonging to the water class, and does not consider the predicted probabilities of these same samples not belonging to the class. By observation the PRC shows that the U-Net architecture approaches the PRC of a perfect classifier, as presented and discussed in Section 2.1.3. In terms of precision and recall, this indicates that U-Net mostly predicts probabilities of pixels being water correctly across all thresholds. The AUPRC is 0.912 for U-Net.

The PRC achieved by DeepLabV3+ on the same data as U-Net is shown in Figure 4.3b. Visual observation of the plot shows that it is very similar to the PRC by U-Net, but with slightly lesser slopes on both the horizontal and vertical lines. This indicates that DeepLabV3+ is slightly closer to the perfect classifier, and the AUPRC of 0.927 proves this observation. In terms of PRC, both models perform well, with DeepLabV3+ slightly ahead of U-Net on 256x256 tiles, which is also validated by the other performance metrics.



(a) PRC of U-Net inferred on 256x256 image tiles, AUPRC = 0.912.

(b) PRC of DeepLabV3+ inferred on 256x256 image tiles, AUPRC = 0.927.

Figure 4.3: PRC of U-Net and DeepLabV3+

Table 4.3 shows performance on the same test set, but model inference is performed on fully sized images. The size of the images the model predicts varies, but is always a lot larger than the size of the training tiles. U-Net is the only model that preserves some of the performance achieved when testing on 256x256px tiles. U-Net achieves a Jaccard index of 0.822 on the water class, and a water accuracy of 87.0%. Precision, recall, and F1-scores are all 0.985. Both DeepLabV3+ models perform poorly when predicting images larger than the training images. The model trained without frozen encoder weights performs worst, with a water class Jaccard index of 0.250 and 25.8% accuracy on prediction of water.

Table 4.3: Model performance on unseen test data consisting of full-sized images.

Model	**Weighted Jaccard**	**Jaccard by Class**	**Precision**	**Recall**	**F1**	**Water Accuracy**
U-Net	0.971	**0.822**/0.984	0.985	0.985	0.985	**0.870**
DeepLabV3+[*]	0.884	0.250/0.938	0.936	0.939	0.938	0.258
DeepLabV3+	0.947	0.681/0.970	0.971	0.971	0.971	0.779

[*] Trained with pre-trained weights NOT frozen during first 30 epochs.

4.2.2 Visual Results

To give a visual indication of model performance, some predicted images from the test dataset is shown. Visual results presented here are limited to the two models that achieves the highest numerical results, U-Net, and DeepLabV3+ trained without frozen encoder weights. Acquired predictions result from both models predicting 256x256px tiles, and these predictions are stitched together to a complete image for visualization.

Figures 4.4, 4.5, 4.6, 4.7, 4.8 and 4.9 shows U-Net predictions of six images consisting of rivers and lakes in different parts of Norway. Each figure consists of the SAR image (a), prediction (b), and the ground truth mask (c) from the "FKB-Vann"-dataset. For comparison, Figures 4.4 and 4.5 show predictions of the same region captured during ascending and descending satellite orbits, respectively. U-Net shows slightly increased accuracy when predicting the descending orbit image. Furthermore, one image of Storvatnet that is captured in March winter conditions is shown alongside the same region in autumn. The SAR images of this region shows that the lake seems slightly more noisy in March, which may indicate that it is covered by surface ice. U-Net still accurately maps Storvatnet in winter, but it does not detect parts of the river running through the image. These images are included to showcase the model's ability to map water bodies during winter. In general, U-Net shows high accuracy for most water body extents, but fails to predict smaller tributaries, compared with ground truth. These tributaries are hard to visually detect in the SAR images.

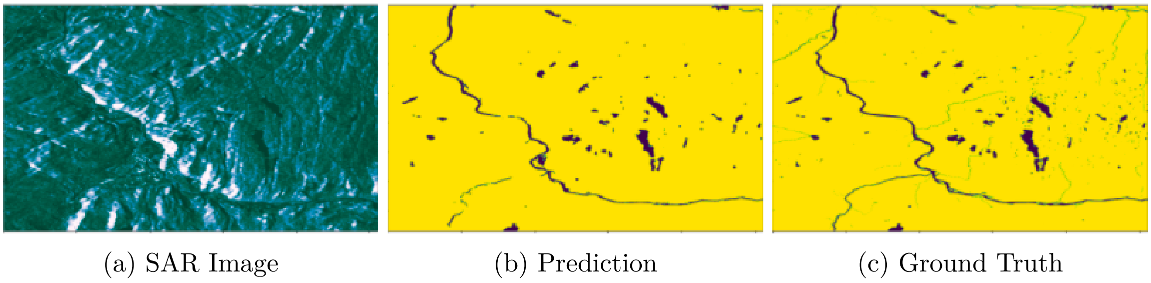

(a) SAR Image (b) Prediction (c) Ground Truth

Figure 4.4: The river Gaula running through Hovin, captured on a descending orbit, September 2020.

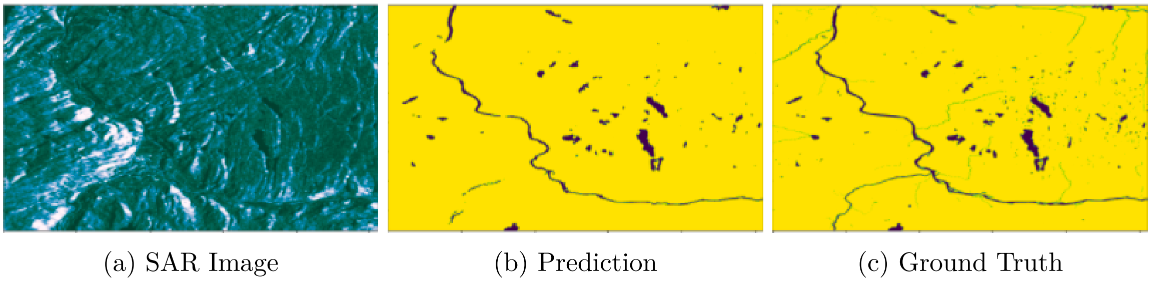

(a) SAR Image (b) Prediction (c) Ground Truth

Figure 4.5: The river Gaula running through Hovin, captured on an ascending orbit, September 2019.

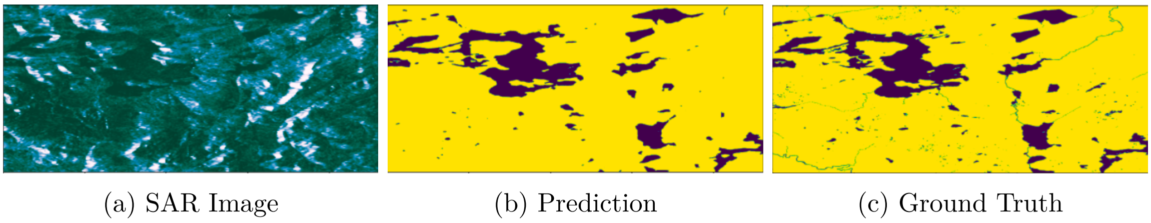

(a) SAR Image (b) Prediction (c) Ground Truth

Figure 4.6: Various lakes in Grytdalen nature reserve, March 2020.

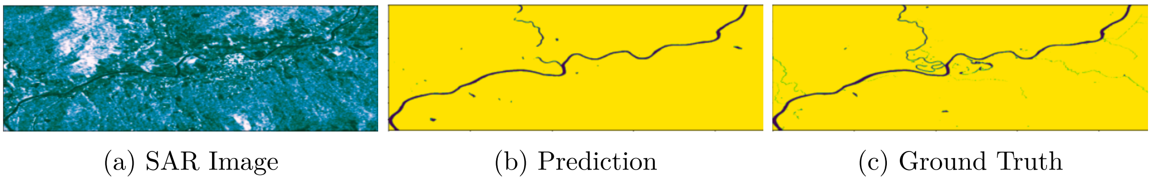
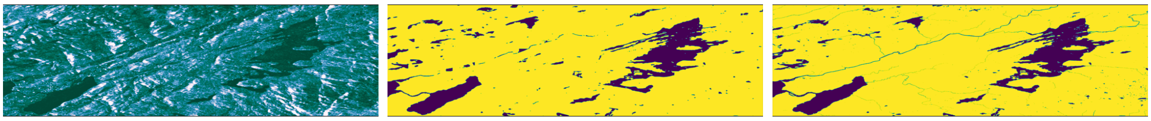

(a) SAR Image (b) Prediction (c) Ground Truth

Figure 4.7: The river Glomma running through Tynset, September 2019.

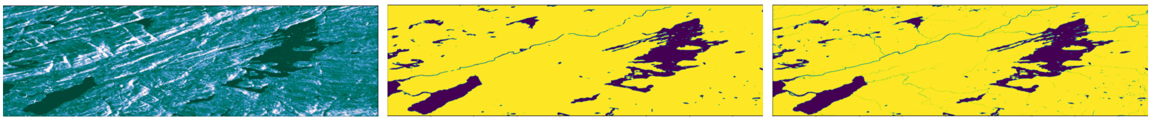


(a) SAR Image

(b) Prediction

(c) Ground Truth

Figure 4.8: Lake Storvatnet and surroundings on the Fosen peninsula, March 2020.



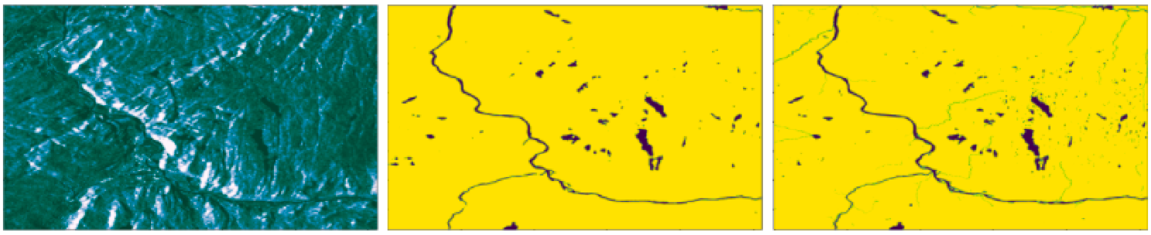
(a) SAR Image

(b) Prediction

(c) Ground Truth

Figure 4.9: Lake Storvatnet and surroundings on the Fosen peninsula, September 2020.

Figures 4.10, 4.11, 4.12, 4.13, 4.14 and 4.15 shows predictions made by the DeepLabV3+ model on the same images as those predicted by U-Net. DeepLabV3+ detects some smaller tributaries to a greater extent than U-Net. This is especially apparent when comparing the main tributary in Figure 4.10 with the U-Net prediction in Figure 4.4. Figures 4.13 and 4.15 also shows that DeepLabV3+ yields slightly more accurate river predictions than U-Net. Compared with ground truth, DeepLabV3+ also seems slightly more accurate than U-Net in predicting the river and small water bodies in Figure 4.14, which is an image captured during winter conditions. DeepLabV3+ generally detects water extents with high accuracy, but also misses details on very narrow rivers and streams compared with ground truth.

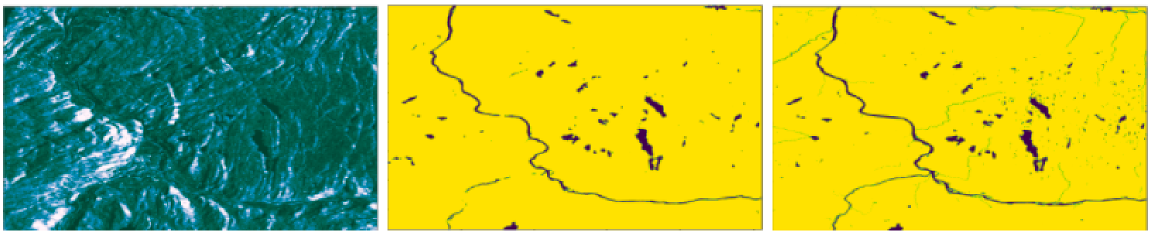


(a) SAR Image

(b) Prediction

(c) Ground Truth

Figure 4.10: The river Gaula running through Hovin, captured on a descending orbit, September 2020.

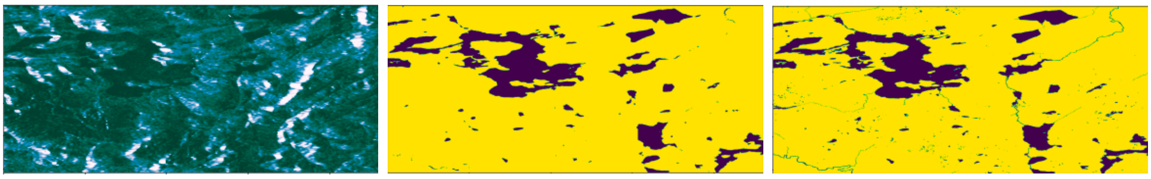


(a) SAR Image

(b) Prediction

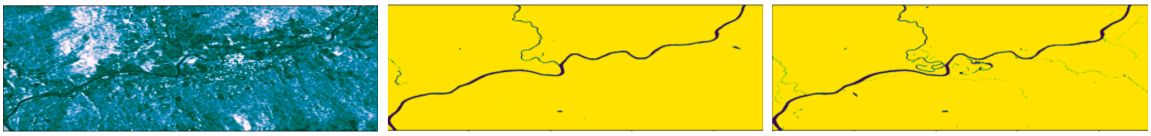
(c) Ground Truth

Figure 4.11: The river Gaula running through Hovin, captured on an ascending orbit, September 2019.



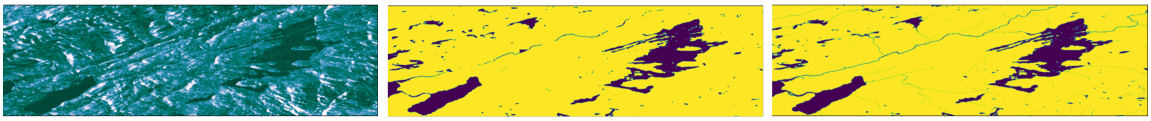
(a) SAR Image (b) Prediction (c) Ground Truth

Figure 4.12: Various lakes in Grytdalen nature reserve, March 2020.



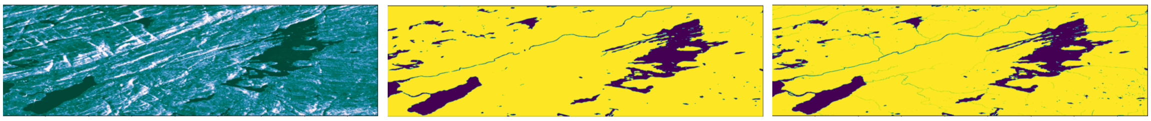
(a) SAR Image (b) Prediction (c) Ground Truth

Figure 4.13: The river Glomma running through Tynset, September 2019.



(a) SAR Image (b) Prediction (c) Ground Truth

Figure 4.14: Lake Storvatnet and surroundings on the Fosen peninsula, March 2020.



(a) SAR Image (b) Prediction (c) Ground Truth

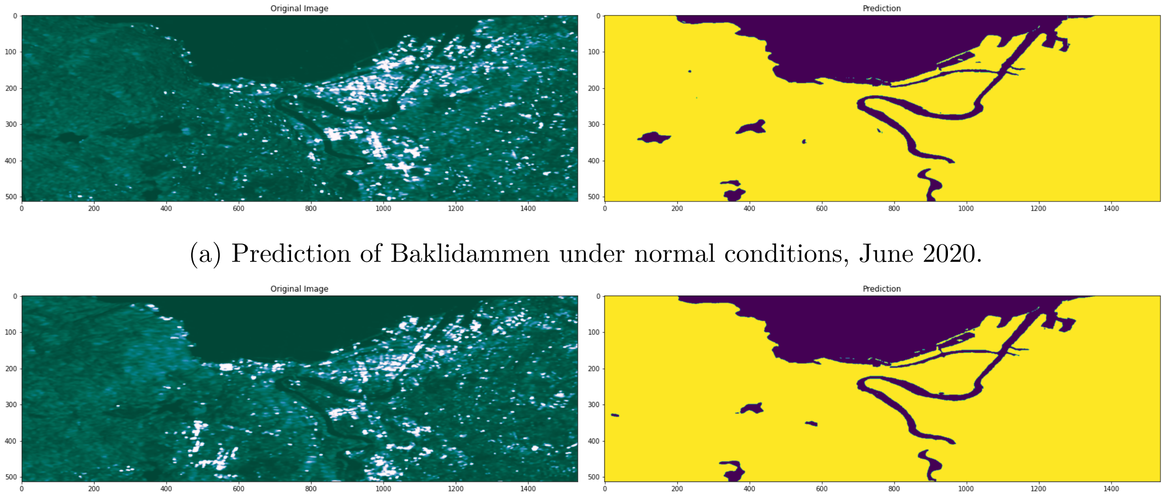
Figure 4.15: Lake Storvatnet and surroundings on the Fosen peninsula, September 2020.

4.3 Change Detection

The main objective of this thesis project is to explore the possibilities for change detection of rivers and water body extents using deep learning. To test the abilities of the implemented models on such a task, the models are tasked with predicting three regions with known variance in water extents. The resulting predictions are shown below, along with a short description of each region.

4.3.1 Baklidammen

Figure 4.16 shows a SAR image of Trondheim and its surrounding areas to the left, and a prediction made by the U-Net model on this region to the right. The area of interest in this image is a small body of water known as Baklidammen, which is located around image coordinates $[175, 320]$. In June 2020, Baklidammen was full of water, as shown in Figure 4.16a. The water body was drained in June 2021, shown in Figure 4.16b. Observing the figures shows that U-Net correctly identifies this change by detecting the water body in June 2020, but not in June 2021.

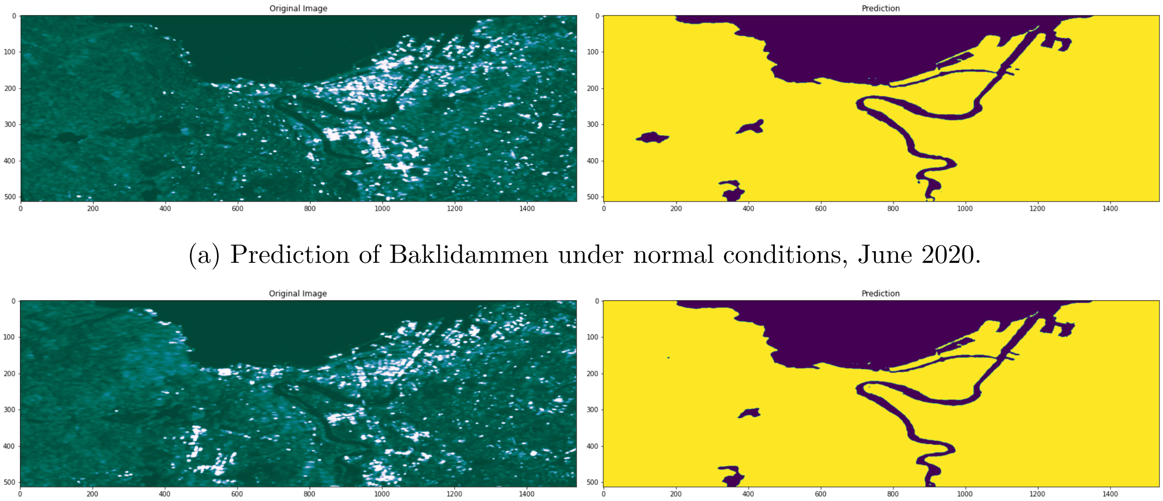


(a) Prediction of Baklidammen under normal conditions, June 2020.

(b) Prediction of Baklidammen after it has been drained empty, June 2021.

Figure 4.16: U-Net prediction of Baklidammen.

Figure 4.17 shows predictions performed by DeepLabV3+ on the same images of Baklidammen. This model also detects that the water has been drained, and does not predict the presence of water after the dam has been emptied. Additionally, DeepLabV3+ correctly identifies the entire Nidelven river running south from Trondheim, which is partially missed by U-Net.

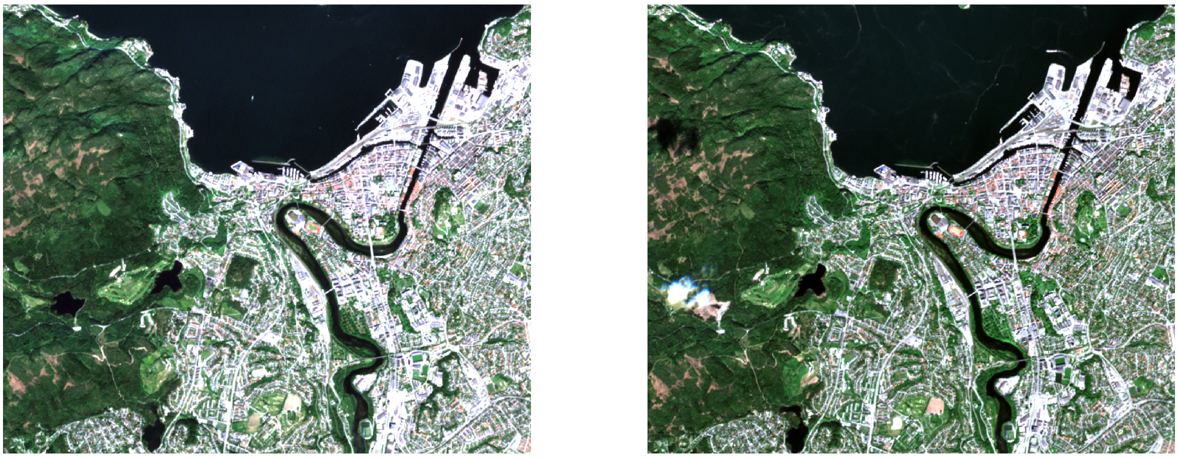


(a) Prediction of Baklidammen under normal conditions, June 2020.

(b) Prediction of Baklidammen after it has been drained empty, June 2021.

Figure 4.17: DeeplabV3+ prediction of Baklidammen.

Visually, Baklidammen might be difficult to spot in the SAR-image. For this reason, optical images from Sentinel-2 showing Baklidammen before and after draining is shown in Figure 4.18. Figure 4.18a shows Baklidammen to the far left, while this area is replaced by the bright lake bed with some clouds covering parts of it in Figure 4.18b. Note that the images from Sentinel-2 are not in the same coordinate reference system and map projection as the predicted SAR images, and are therefore not directly comparable in terms of scale and positioning.



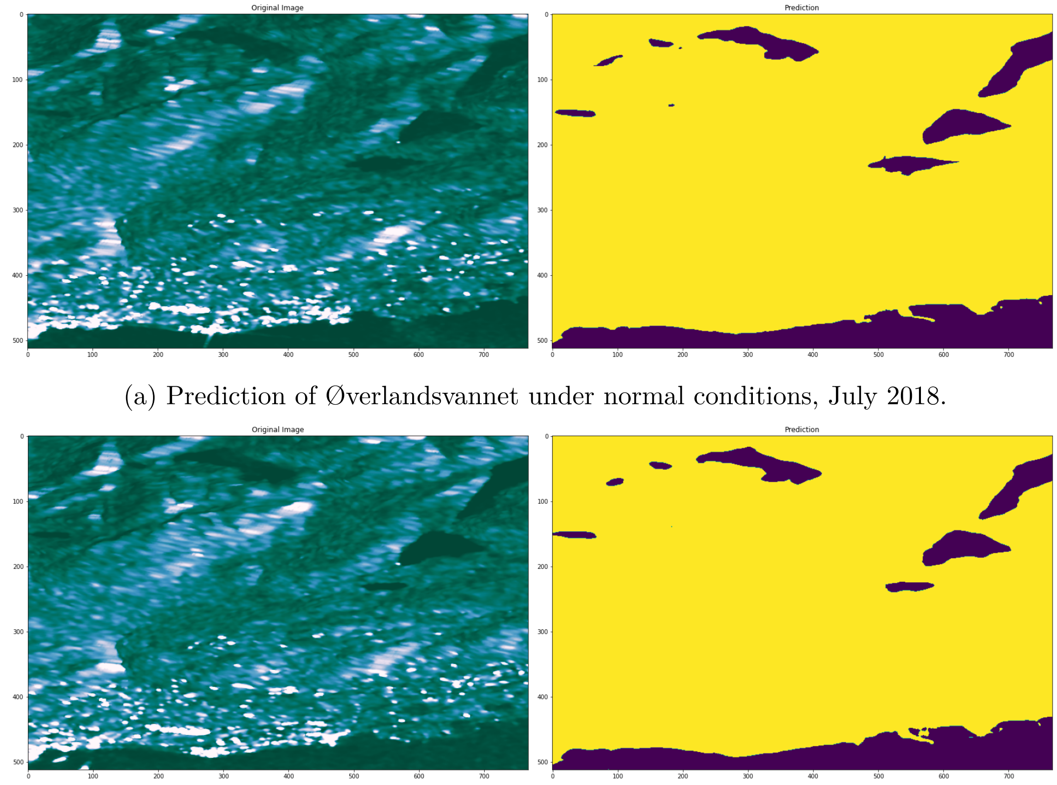
(a) Sentinel-2 image of Baklidammen under normal conditions, June 2020.

(b) Sentinel-2 image of Baklidammen after it has been drained empty, June 2021.

Figure 4.18: Sentinel-2 images of Baklidammen.

4.3.2 Øverlandsvannet

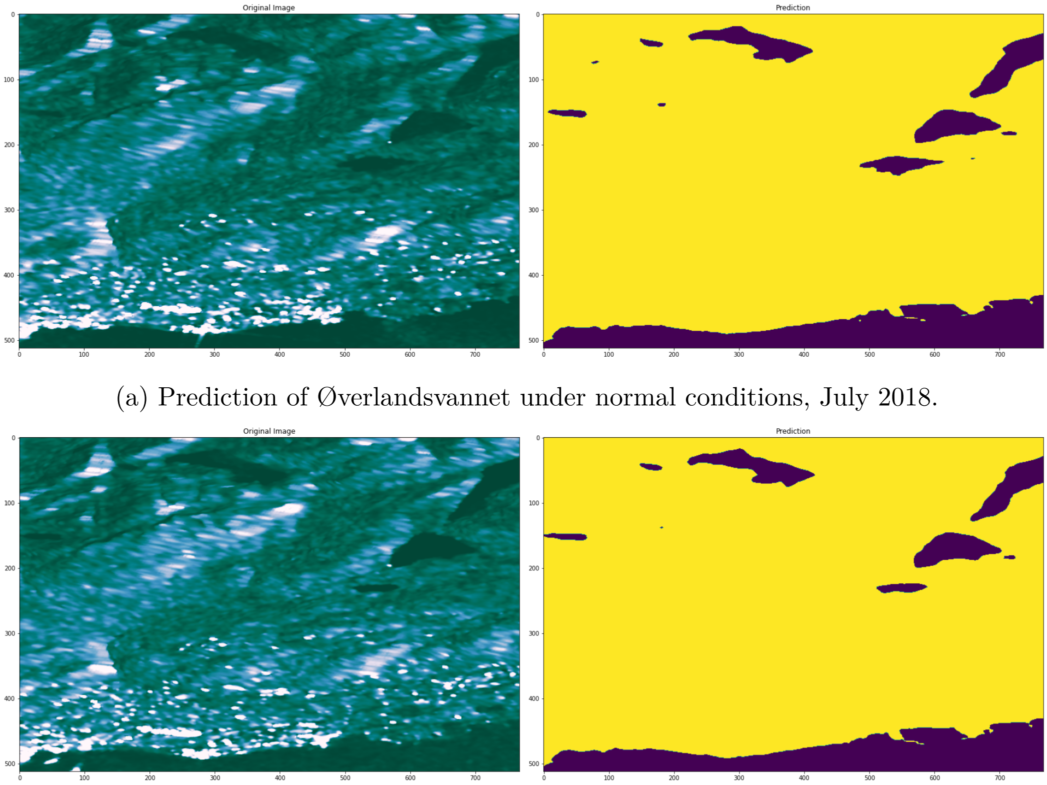
Figures 4.19 and 4.20 shows U-Net and DeepLabV3+ predictions of Øverlandsvannet before and after partial draining. Øverlandsvannet is located at coordinate $[550, 220]$ in the right part of the image. In July 2018, Øverlandsvannet was full, correctly identified by both models in Figures 4.19a and 4.20a. The dam was partially drained in July 2019, and both models correctly detect a smaller water body in Figures 4.19b and 4.20b. Sentinel-2 images of the same time period is shown in Figure 4.21 for reference. Again, this image differs in map coordinate system and is not directly comparable to the SAR images and predictions in positioning and scale.



(a) Prediction of Øverlandsvannet under normal conditions, July 2018.

(b) Prediction of Øverlandsvannet after partial draining, July 2019.

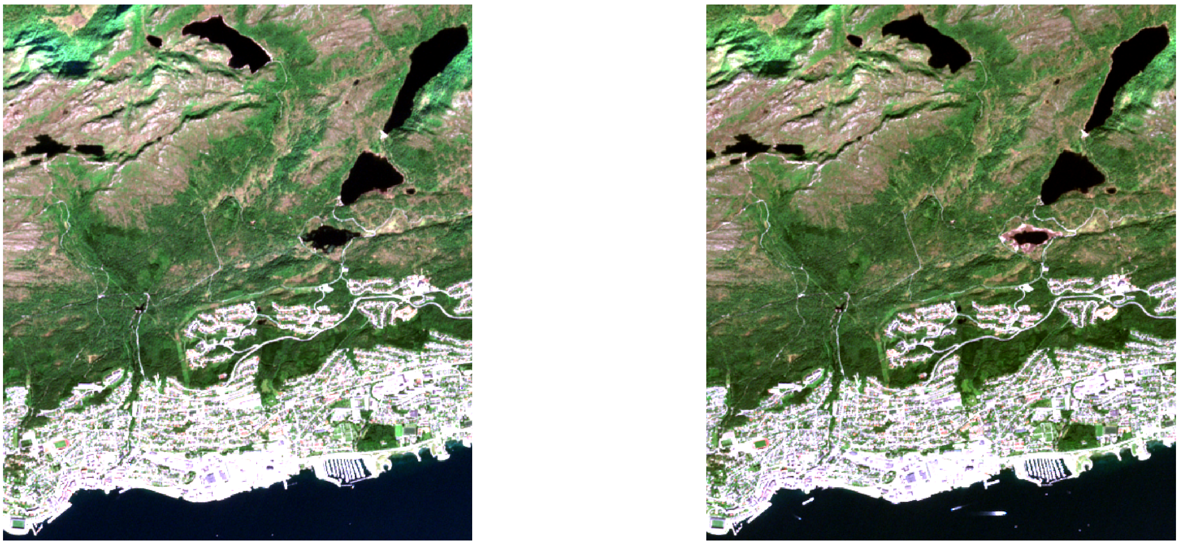
Figure 4.19: U-Net predictions of Øverlandsvannet.



(a) Prediction of Øverlandsvannet under normal conditions, July 2018.

(b) Prediction of Øverlandsvannet after partial draining, July 2019.

Figure 4.20: DeeplabV3+ predictions of Øverlandsvannet.



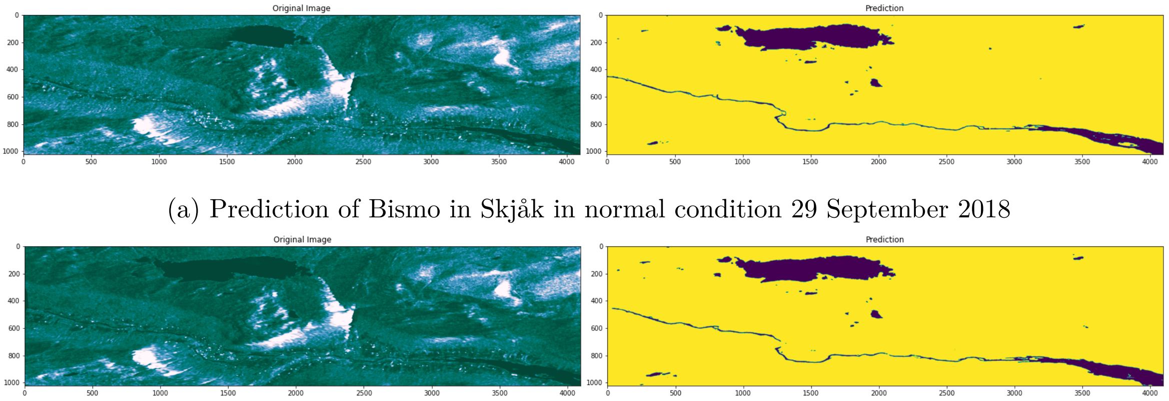
(a) Sentinel-2 image of Øverlandsvannet under normal conditions, July 2018.

(b) Sentinel-2 image of Øverlandsvannet after partial draining, July 2019.

Figure 4.21: Sentinel-2 image of Øverlandsvannet

4.3.3 Bismo, Skjåk

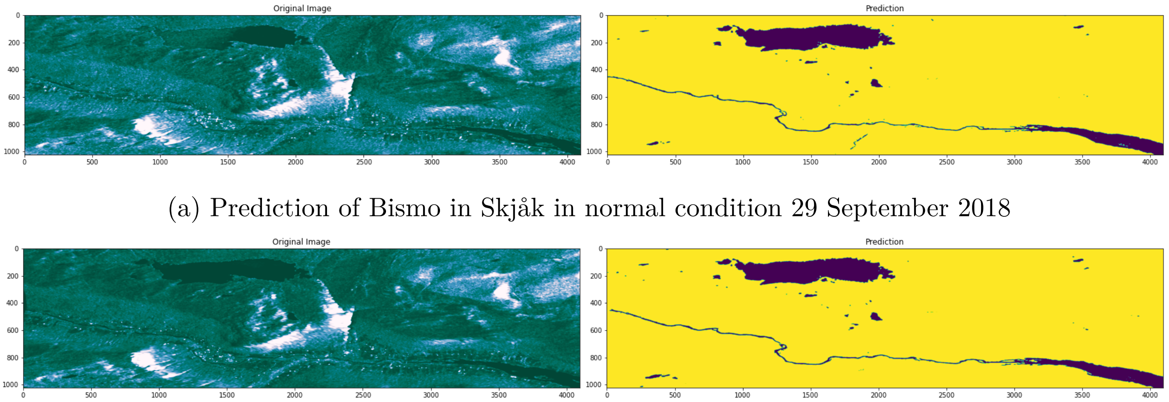
During October 2018, Skjåk experienced a flood event due to rain and large amounts of melting snow, which caused the river Otta to overspill its banks. Figure 4.22 shows SAR images of Bismo, a town that was especially exposed during the flood, before and shortly after the event. The predictions are acquired by the U-Net model, while Figure 4.23 shows the same images predicted by DeepLabV3+. Ideally, the SAR image should have been from the peak of the flood the 15. October, but this was not available from Sentinel-1. Thus, the image presented here does not represent the full extent of the flood, but water levels are still known to be slightly higher than normal.



(a) Prediction of Bismo in Skjåk in normal condition 29 September 2018

(b) Prediction of Bismo in Skjåk just after a flood 16 October 2018

Figure 4.22: U-Net prediction of Bismo in Skjåk, before and after a flood



(a) Prediction of Bismo in Skjåk in normal condition 29 September 2018

(b) Prediction of Bismo in Skjåk just after a flood 16 October 2018

Figure 4.23: DeeplabV3+ prediction of Bismo in Skjåk, before and after a flood

Figure 4.24 and Figure 4.25 are comparisons of the September and the October images of Bismo as predicted by DeeplabV3+ and U-Net. Both are created by subtracting the September prediction from the October prediction and plotting the difference over the October prediction. Thus the dark violet colored areas were detected in the October image, but not in the September image, while the orange areas were detected in the September image, but not in the October image. The lighter colored violet areas were detected in both images. A comparison of the two figures shows that U-Net predicts more water shortly after the flood than DeepLabV3+, while it predicts less water prior to the flood. U-Net detects a larger area of water in the river and lakes in the October image, but it has missed parts of the river on the left side of Bismo. Bismo is located in the semicircle created by a large bend in the river, straight below the largest lake in the prediction. The difference between the two predictions made by DeepLabV3+ is smaller than the corresponding U-Net predictions. Both models agree that the river is a couple of pixels wider on average along each bank. Both models seem to agree on where the river has become a lot bigger and have detected more water in the same areas, but they differ in how much larger they think it has become, and how large the river was prior to flooding.

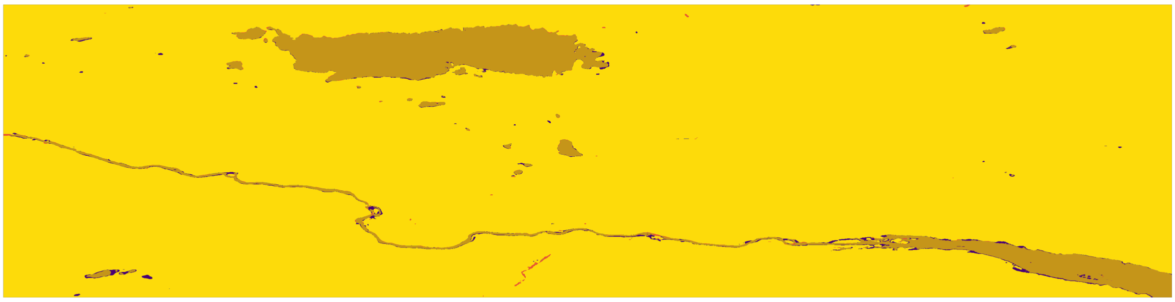


Figure 4.24: Comparison of the two Bismo images predicted by DeepLabV3+. The darkest violet colored areas was only detected in the October image, while the orange colored areas was only detected in the September image.

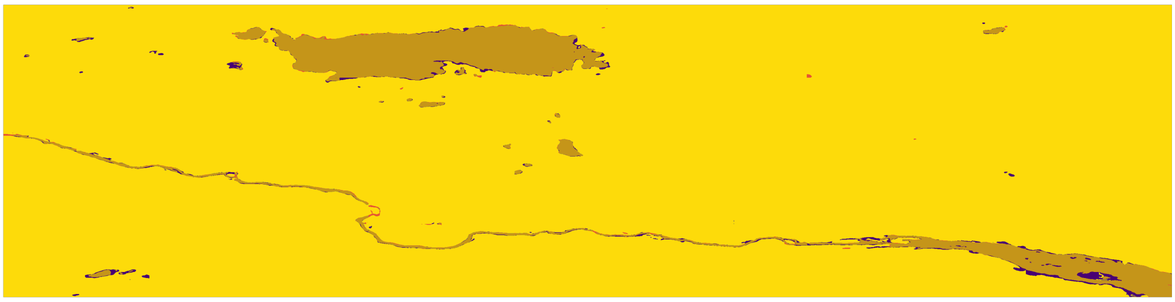


Figure 4.25: Comparison of the two Bismo images predicted by U-Net. The darkest violet colored areas was only detected in the October image, while the orange colored areas was only detected in the September image.

Chapter 5

Discussion

This chapter provides a thorough discussion of the results presented in Chapter 4, giving insight on model training and model performance on test data. The aim is to discuss parameters, setups and data properties that may affect model performance, which creates a foundation for possible future improvements presented in Chapter 6. Data accuracy is discussed due to its large impact on model performance, and to give an indication of the validity of the results. This chapter additionally covers the usability of the work in practice, specifically related to change detection of water extents.

5.1 Model Training

The training results show that DICE loss and accuracy gradually converge for both models. Although DeepLabV3+ shows spikes of increased loss after certain epochs, both models converge evenly on training data and validation data. This commonly indicates that the models do not over or underfit, but such a conclusion is not definitive in this case since the validation dataset consists of the same regions as the training set. The possibility of similar images in the validation data, as compared to the training data, cannot be neglected completely. However, intended mapping is restricted to regions within Norwegian borders which are mostly characterized by mountains and valleys. It is therefore likely that most regions show some form of similarity due to the country's homogeneous topography, and even a dataset consisting of entirely new ROIs will be somewhat similar to the training datasets. Topographical similarities thus suggest that an increase in model performance on training and validation datasets likely also increases model performance on test data, regardless of training depth. This indicates that overfitting may not be an issue since all data that is possible to acquire in Norway is similar. The performance metrics in Section 4.2.1 do, however, show a slight gap in performance between validation data and test data, but this gap is too small to classify the models as overfit. Besides, comparing the metrics shows that a higher training performance is accompanied by higher test performance, despite the gap. This discussion thus concludes that overfitting models are highly unlikely, but care should nonetheless be taken to ensure performance on the test dataset does not start to degrade. With this background, and since results show that the DICE loss function still approaches a minimum after the total of 45 epochs trained, model performance may perhaps increase slightly if trained deeper. However, convergence is already so slow that potential performance increases are likely minimal, and time is better spent improving other aspects of the models and datasets.

Besides training depth, further improvement of the models may be achieved by more in-depth testing and tuning of the training parameters. The most notable parameters to consider are the chosen optimizer, learning rate and batch size. The values of these parameters are chosen empirically by smaller training sessions on a smaller dataset, the method of which is described in Section 3.2.2. Due to competition for GPU resources, the chosen approach is to seek good results while minimizing the time it takes to train each model. The idea of training with lower batch sizes, such as 16 or 8, is discarded due to a substantial increase in training times. Furthermore, larger batches are not tested due to GPU memory limits but may have yielded quicker training times. Additionally, Keskar et. al [Kes+17] provides evidence favoring smaller batch sizes, where they indicate that large batch training degrades model quality. Given an ample training environment for longer, it would thus be beneficial to train the models with lower batch sizes to examine potential improvements in terms of model convergence. However, the effect of batch size on model performance is unknown until tested. Batch size is also closely related to the learning rate, and further tuning and experimentation with these two parameters may or may not give increased performance.

Training optimizers are plenty, and this thesis project uses the Adam optimizer to train every model. Adam is preferred due to being computationally efficient while effectively converging to a minimum. Although the topic is still debated, Gupta et. al [Gup+21] shows that stochastic gradient descent generalizes better than Adam. This implies that SGD commonly performs better than Adam on test and validation data. The reason this project does not train models with SGD is that it is computationally expensive with increased training times, and the expected improvements are not thought to be large enough to compensate for this.

All this being said, the highly empirical nature of all the mentioned training parameters makes this an issue of having the time to perform extensive parameter tuning. Also, the models show healthy training behavior, and although parameter tuning may yield slight improvements it is not close to the largest source of model inaccuracy. Time is therefore rather spent improving and discussing other performance altering aspects of the models and data.

5.2 Model Accuracy

Model performance on test data is presented with a set of numerical performance metrics and some predicted images. Precision, Recall and F1-score are image classification metrics that indicate model predictive power and sensitivity. The results show that these metrics are seemingly high and similar for all models, despite predicted images showing some prediction inaccuracies. This is likely due to the number of pixels that are correctly classified as “not-water”. Since this class constitutes 92.15% of the pixels in the test data, inaccurate predictions of the comparably few pixels that are rivers and small water bodies will not substantially affect the metrics. These metrics are calculated as weighted class averages, but micro-averaging is also tested to ensure it is not the weighting that yields such high results. Micro-averaged precision, recall and F1-score yields nearly the exact same values as the presented weighted averages, proving that this is indeed due to the amount of correct model predictions.

Performance on the minority “water” class is of special interest since the different models’ ability to map water is central to the problem statement. For this reason, the “water”-class Jaccard index and accuracy are the most valuable metrics. These metrics better represent what is visually observable in the presented images. Predictions are accurate but miss the finer details related to streams and very small water bodies, as compared with the masks. Both metrics, however, are likely higher than indicated, but this is not apparent since the masks do not depict the content in the predicted image accurately. This is further discussed as a part of the data accuracy discussion in Section 5.3. Further analysis of the predictive power and sensitivity is provided by PRC curves. These give the same information as the precision and recall metrics, but for all possible probability thresholds over which to classify a pixel as “water”. The shape of the PRCs suggest that the two classes are highly distinguishable across most probability thresholds. As such, the PRC curves motivate SAR images as a data source for water mapping by deep segmentation networks.

In addition to model performance on tiled test data, the results also present the performance of all three models on the same, fully sized images. Interestingly, both DeepLabV3+ models perform poorly when predicting full sized images, but very well when predicting tiles. Furthermore, the model that performs best in predicting 256x256px tiles performs worst when predicting fully sized images. This is observed despite all models training with image shape $(\mathrm{None}, \mathrm{None}, 3)$, which enables training and testing with images of all sizes. One possible reason for this behavior is that the architecture of DeepLabV3+ performs contextual feature extraction by Atrous Spatial Pyramid Pooling. This may imply that image context plays a larger role during training and testing than it does for other network architectures. Larger images have an altered field of view, and thus context, as compared to the tile sizes the models are trained on. The DeepLabV3+ model partially trained with frozen weights has not fully learned these context related features, and thus performs better when predicting fully sized images than the best performing model on 256x256px tiles. Evidence suggests that DeepLabV3+’s performance on differing image sizes degrades with an inverse proportionality to its performance on the training image size. U-Net also performs slightly worse on fully sized images than the tile size it is trained on, but mostly retains its performance. The slight decrease in performance is likely due to the same context related feature extraction. This effect is much less prominent with the U-Net architecture, which frequently concatenates deconvolved feature maps in the decoder with low level features from the encoder, giving a more detail-oriented classification approach as opposed to context related. With this proposed background, achieving even performance on all image sizes requires a training dataset with varying image sizes. Although possible, this makes training unpredictable with regards to batch size and available GPU memory capacity. Varying image size may also vary image properties, features and context such that it is unlikely that performance will be as good as achieved when training and testing with a single size. It should also be mentioned that only fully convolutional networks, such as the U-Net and DeepLabV3+ architectures, can accept varying input shapes. If dense layers are appended to these networks this is no longer a possibility.

5.3 Data Accuracy

This section will discuss how the quality of the data affects the performance of the models compared with the utilized masks and predicted images. In terms of performance, inaccuracies in the dataset are thought to be the main limiting factor. This section first covers the composition of the dataset, highlighting potential strengths and weaknesses in data quantity and variation. It then discusses model performance compared with the actual contents in the SAR-images that are acquired with an image resolution of 20x22m. Finally, the section discusses the accuracy of the masks, which limits prediction accuracy compared with ground truth, and thus affects the calculated performance metrics.

5.3.1 Analysis of the Dataset

The composition of the dataset is the foundation of this discussion and can be found in Appendix A. The created dataset is assembled from 175 Sentinel-1 scenes from 2019 and 2020 where about half of the images are captured on an ascending orbit, and half are captured on a descending orbit. Sentinel-1 consists of two satellites, and the distribution of scenes between the two is roughly even. Although these statistics may or may not affect performance, it is beneficial for the models to be equally exposed to both orbit directions and data sources to ensure potential differences are accounted for. The main deviation in the image distribution is the month they are acquired in. As shown in Table A.3, an overwhelming majority of the images are captured in September of both years. This occurs despite specifying the ASF download request to search for images throughout the entire two-year period. However, the models seem to generalize and perform well when predicting images from all times of year. This is proved by the predictions in Section 4.2.2, which shows images of Grytdalen and Storvatnet in winter conditions. Model results thus indicate that the acquired scenes from Sentinel-1 is a good foundation for a dataset, despite images being unevenly distributed throughout the times of year.

The dataset used to train and validate the models is built from the Sentinel-1 scenes after application of a processing pipeline. The resulting dataset consists of 118 745 train and validation images of size 256x256px. U-Net and DeepLabV3+ are based on image classification encoders designed for competitions that encompass datasets containing about one million image tiles [UU]. It is therefore hypothesized that both architectures perform best when trained on large datasets. As such, a decision is made to produce the larger dataset rather than settling for the subset of this dataset for both parameter tuning and results. Since the images consists of Norwegian water landscapes, they are quite monotonous regardless of region. Because of this, it is improbable that an even larger dataset will noticeably improve performance. However, it may be beneficial to include more regions to increase the variation of the dataset without necessarily increasing the total number of images.

One possible source of inaccuracy in the dataset is the imbalance of classes. The imbalance is minimized by attempting to choose the ROIs in Section 3.1.1 to maximize the amount of water to dry land. Despite this attempt, the final dataset consists of only 10.8% water. This may affect model performance in two ways. Firstly, the models may be slightly biased towards predicting dry land solely based on the number of labels in the dataset. Secondly, there are limited representations of the "water"-class for the models to learn from, which results in hesitation to predict this class where they are unsure. These negative implications are worth mentioning, but probably do not have a huge impact on the results since water is very distinguishable from dry land due to its scattering characteristics, as elaborated on in Section 2.3.2. It is therefore unlikely that there are many pixels that lead to model uncertainty.

5.3.2 Resolution of SAR Images

SAR image resolution determines the level of detail in which it represents the true landcover beneath. Each pixel in the acquired images from Sentinel-1 represents 10x10m on the ground, but the GRDH scenes are produced with an original resolution of 20x22m. This resolution implies that very small water bodies and streams are not present in the images due to the radar not being able to distinguish multiple scatterers in a 20x22m area. Furthermore, very small changes in water extents will be invisible. The consequence of this is that the models may not be able to detect streams and small changes in images, since they are simply not there. In terms of performing very small-scale change detection to warn of potential river flooding, this is of course unwanted. However, the images of Bismo in the results show that the models are still capable of detecting relatively small increases in river extents, but anything smaller than this requires higher resolution SAR images. Inaccuracies due to image resolution is not directly measurable by the results in Chapter 4 since all metrics are computed against the masks, which do not depict the entire truth in the SAR images. A discussion of the implications of mismatch between masks and images is provided in Section 5.3.3.

5.3.3 The Relation Between Masks and Images

This thesis project uses highly accurate masks produced from water labels acquired from the Norwegian Mapping Authority. During processing, these masks are sampled to a pixel size of 10x10m to match the pixel size of the SAR images. However, the original labels are produced from aerial imagery with a resolution range of 7cm to 25cm. So, despite being sampled to equal pixel sizes, the level of detail in the masks is much higher than in the corresponding SAR images. Thus, the masks label too much to correctly identify water in the SAR image. The predictions in Section 4.2.2 shows that both model architectures fail to detect small rivers and streams that are present in the masks. Figure 5.1 shows one such stream up close, which is from the predicted image of Gaula presented in the results. The stream is not visible to the human eye in the SAR image, which suggests that it is too small and narrow to be present in the image. All labels that are present in the masks, but not in the SAR-image, is considered noise in the dataset and might worsen the performance of the models.

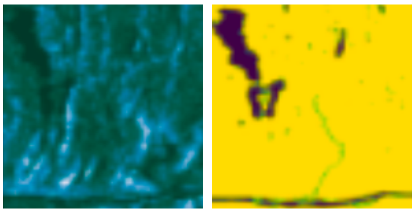
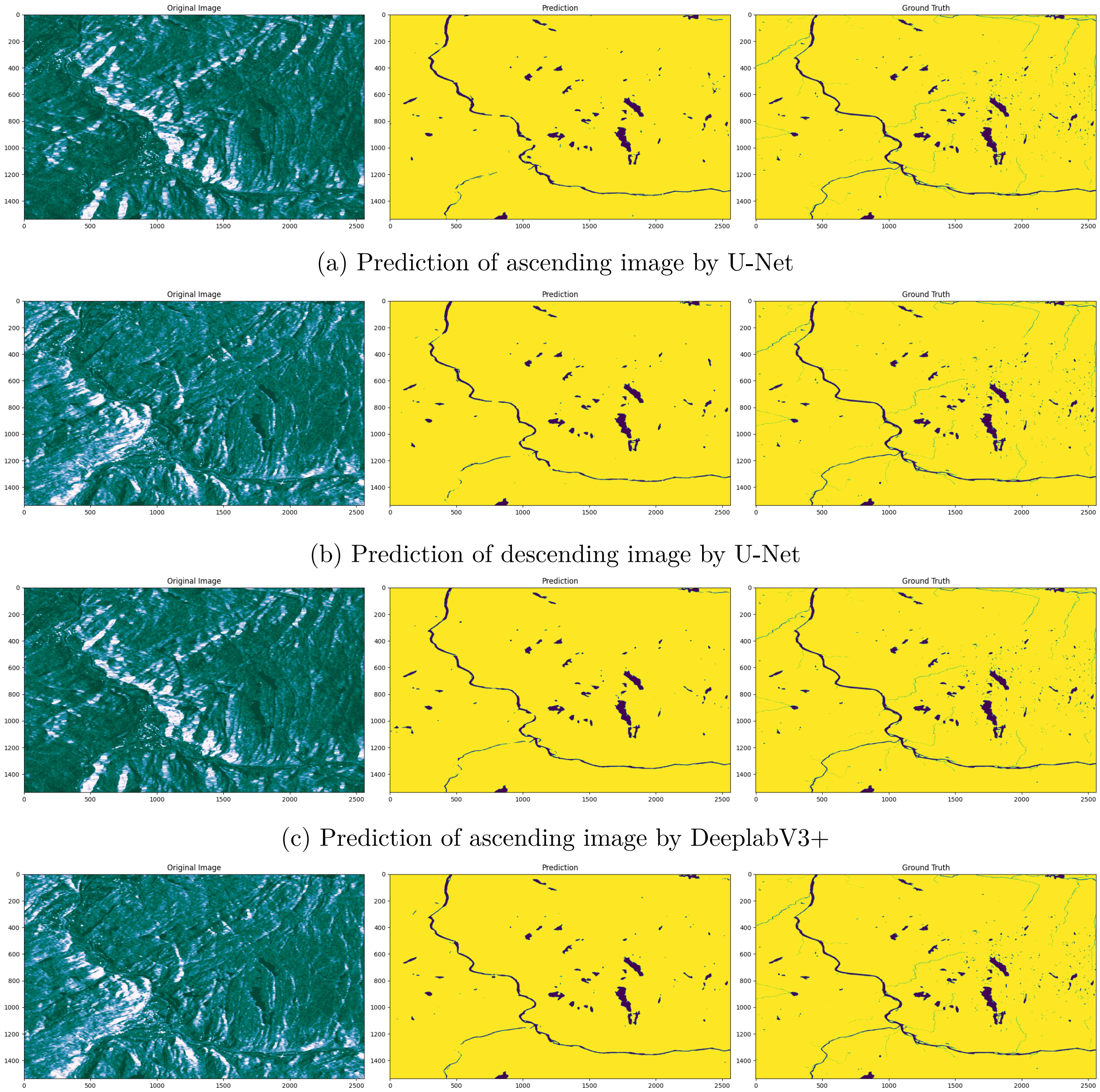


Figure 5.1: Zoomed in SAR image and mask showing that the labeled stream is not visible in the SAR image due to differing levels of detail between images and masks.

It is hard to judge model performance by comparing the predicted images to the ground truth masks. The masked area represents water levels at the time the images used by the Mapping Authorities to produce labels, were captured. As such, the masks are static and do not conform with frequent changes in water extents and landcover. For example, some rivers are dry parts of the year and others overflow its banks due to heavy rain and melting snow in the mountains. These changes in the images are not accurately described by the masks. Such mask inaccuracies, combined with noise, means that all metrics calculated with respect to ground truth will not be truly accurate. With this in mind, the true accuracy and performance of the models is unknown, and even an accuracy of 100% is not necessarily a perfect result. Furthermore, the models can achieve 90% total accuracy by only predicting "not-water" due to class distributions. Other class-based performance metrics are therefore vital for analysis of performance. Due to mask inaccuracies and class imbalances, visual evaluation by comparison of SAR images, predictions and masks are valuable in addition to the calculated performance metrics.

Another reason the masks from "FKB-Vann" are not always accurate is that SAR images suffer from artifacts caused by the side-looking geometry used to capture the images. Norway is a country in which the landscape largely consists of mountains, which makes effects such as layover, foreshortening and shadowing prominent. Accurate geo-referencing by terrain correction limits the effects to a minimum, but they still affect the visibility of streams and rivers located close to mountains and steep slopes. The effects are dependent on the flight direction of the satellite relative to the landscape. Figure 5.2 shows predictions of Gaula running through Hovin, where one image is captured during an ascending orbit, and one image is captured during a descending orbit. In the upper left corner, there are two small spots of water that the masks indicate are a pond and part of a narrow river. Predictions from both U-Net and DeepLabV3+ show that these spots of water are more visible in the images captured during the descending orbit. The narrow river in the lower left corner shows that the vertical part of the river is more visible in the ascending image, while the horizontal part is more visible in the descending image. Also, there is a stream in the very top right corner that is partially visible in the ascending image, but nearly invisible in the descending image. Both images are from mid-September, and the only major difference between the two is thus the orbit direction during which the images are captured. Further analysis of the SAR-images annotated by "Original Image" shows that the white, highlighted part of the mountains is on opposite sides of the valley depending on orbit direction. This indicates that the regions covered by shadows created by the mountains is dependent on satellite flight direction. The challenge of this in terms of the masks, is that they do not account for potential shadows covering parts of the labeled water. Thus, some parts of the labeled water may not be present in the images due to image artifacts caused by flight direction. This results in additional noise in data labels which may have a negative impact on train and test performance.

Tailoring each mask to its assigned image might yield higher model performance in terms of calculated performance metrics, since the masks would then be an accurate representation of the actual image contents. However, this is a labor-intensive task that requires a lot of manual work. To further complicate the process, there is no simple truth to the actual contents of the images. Human eyes are likely not good enough to accurately assess the SAR images due to limited resolution and an unfamiliar representation of data. One possibility is to compare aerial multi spectral images to SAR images, but this is limited to images that are not exposed to cloud cover, atmospheric interference and nighttime. After all, one of the biggest advantages of SAR is the ability to analyze data captured at night and in cloudy conditions. If this is neglected, there is also the issue of removing areas covered by SAR image artifacts, such as shadows and layover, that is not present in the multi spectral images. Despite some challenges, tailoring masks to image contents will result in more accurate results since metrics are calculated with respect to the ground truth masks. It will also reduce noise and inaccuracies in the dataset that may increase general model performance.



(a) Prediction of ascending image by U-Net

(b) Prediction of descending image by U-Net

(c) Prediction of ascending image by DeeplabV3+

(d) Prediction of descending image by DeeplabV3+

Figure 5.2: Comparison of images captured during ascending and descending orbits.

5.4 Change Detection of Water Extents

The intended purpose of the implemented and trained models is to map water accurately enough to detect changes in water extents. Section 4.3 in Chapter 4 shows model predictions on several images with a known change in water extents. In some images, these changes are due to human interference, ie. draining of a lake, and some are caused by known flood events. Both cases may benefit from automated water mapping. For example, accurately mapping flood extents and pre-cursors to these events may help prevent some potential flood-related consequences. This is just one potential application of an automated water mapping system. To fully realize such a solution by semantic segmentation, the deep learning models have to prove the ability to detect changes with high accuracy.

By observation of the results, it is apparent that the models detect the larger drained lakes with high accuracy. To further prove this, Figure 5.3 shows up close images of Øverlandsvannet lake as presented in the results, along with predictions made by the best performing DeepLabV3+ model. The lake is highlighted by a red box in the SAR image. Visual observation shows that the model nearly perfectly maps the lake in both cases. However, the representation of the lake in the SAR image makes it difficult to know the exact extent of the lake simply by looking at it. It is, however, very obvious that the model detects the change in water extent from one image to the other.

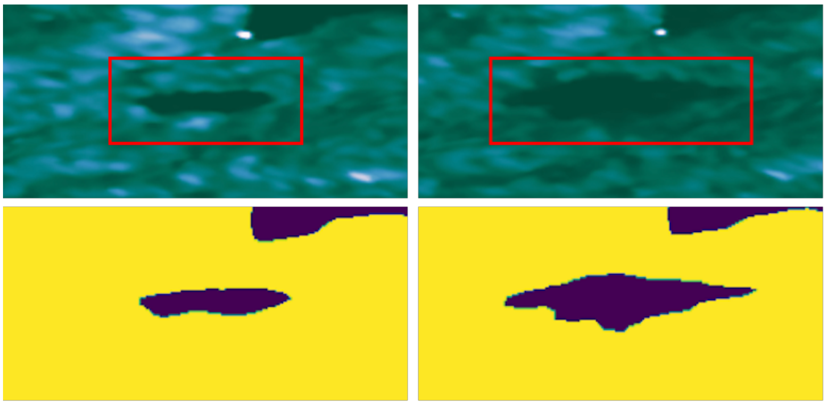
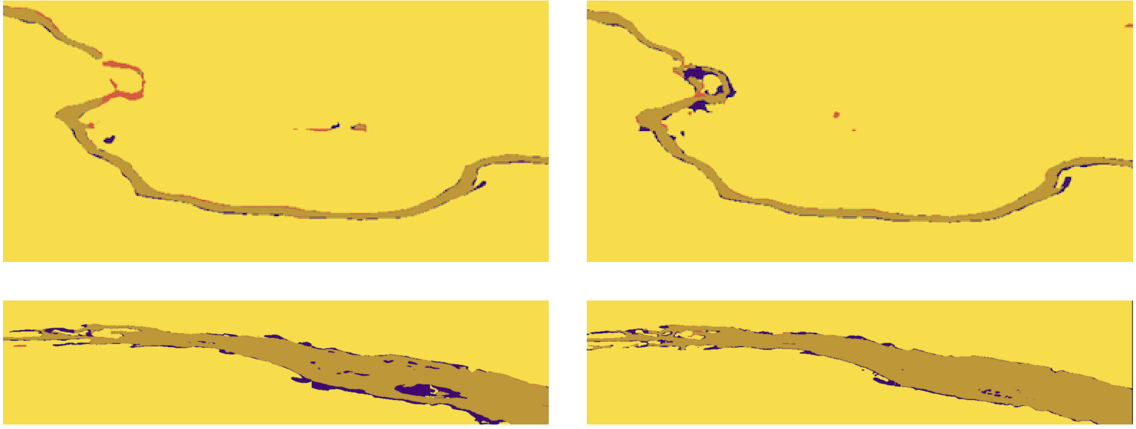


Figure 5.3: Zoomed in SAR image of Øverlandsvannet lake along with DeepLabV3+ water prediction. The left side image and prediction shows a drained lake, while the right image shows the lake's natural condition. The lake is highlighted in a red box in the SAR images for simplified observation.

Although both models are capable of accurately detecting changes to larger water extents, river flooding is normally characterized by much smaller extent variations. Rivers are also harder to detect under normal conditions due to their narrow shape. This presents challenges in terms of change detection with the purpose of detecting and monitoring river flood events. Figures 4.24 and 4.25 in the results show that U-Net and DeepLabV3+ differ slightly in the water extents they detect on such a small scale. For easier observation, Figure 5.4 shows the flood exposed areas of the same images up close. The top left corner of Figure 5.4a shows that U-Net fails to detect the river where it splits during the flood. Excluding this, both models generally seem to agree that the river is slightly wider shortly after the flood, although U-Net predicts lower water levels prior to the flood than DeepLabV3+. The lack of an accurate ground truth makes it difficult to decide on which model performs better on such small scale change detection. Cloud cover during the flood also eliminates the possibility of using optical satellite images from Sentinel-2 for comparison.



(a) U-Net predictions of Bismo town (top) and slightly east of Bismo (bottom).

(b) DeepLabV3+ predictions of Bismo town (top) and slightly east of Bismo (bottom).

Figure 5.4: Zoomed in difference image presented in the results. Orange is water only detected during normal conditions in September, violet is water only detected during flooding in October, while dark yellow is water detected in both images.

Based on Figure 5.4, both models are able to detect widening of river extents. However, none of the models predict substantial overspill along the river banks in Bismo town. There are a few possibilities that may explain this. Firstly, and most importantly, the 16. October image that is predicted here is captured one day after the peak of the flood. News outlets[1] inform of open roads since most of the spilled water has receded on 16. October. Thus, there might not be much overspill for the models to detect in the SAR image. Furthermore, any potential leftover parts of the flood water that is not part of the river may be very shallow. If the water is shallow enough, the C-band radar waves may penetrate and reflect off the ground underneath. This results in backscatter from the ground beneath, rather than a lack of backscatter due to specular reflection at the water boundary. Although both of these possible explanations result in a SAR image with very little spilled water, the latter is unlikely since C-band SAR wavelengths have a low penetration depth in water, which is a specular reflector. However, the spilled water may be contaminated by debris that may result in some backscatter. Another possibility is that the radar receives a mix of strong backscatter from the buildings and weak backscatter from potential overspill. Due to the 20x22m image resolution, these two signals may not be distinguishable, implying that the image is not able to separate buildings and water but shows a mix of these. Despite the limited resolution, this is rather unlikely since buildings in such areas are commonly built with adequate spacing for the radar to image the space in between. Besides, if this were the case, there would likely still be some overspill in areas without buildings. This being said, the slightly increased water extent that is predicted by the models is likely relatively accurate. But again, a lack of an actual ground truth makes it impossible to be sure.

The thesis results show that both U-Net and DeepLabV3+ are capable of accurately mapping water and detecting changes in water extents. Small-scale change detection proves more

[1] Timeline of the flood event covered by NRK: `https://www.nrk.no/innlandet/flom-skjak-og-lom-1.14248129`.

challenging due to the limited resolution of freely available SAR data. Lack of perfectly accurate masks also limits the potential performance of these models. This indicates that further improvements are necessary to realize the full potential of modern segmentation architectures. In terms of flood detection and monitoring, the trained models are not quite accurate enough to reliably yield perfectly precise flood and river extents at all times. However, it is possible to use these models in a semi-automatic flood monitoring system. For example, the current iteration of the models may prove good enough to issue warnings upon detection of changes in water extents. All results indicate that the models notice widening of rivers and draining of dams. By issuing a warning whenever such a change exceeds some threshold, human labor may be used to accurately assess the water extents and flood risk. Chapter 6 further elaborates on some improvements that may bring this implementation one step closer to the accuracy needed for fully automatic flood risk assessments.

Chapter 6

Further Work

This additional chapter provides a brief overview of possible improvements that are not implemented during this thesis project due to time or resource limitations. All suggestions of further work are based on the results along with discussions of these, presented in Chapters 4 and 5, respectively. Current assessments of model performance show that the main factor limiting performance is the resolution of the acquired SAR images from Sentinel-1, and how well the masks correspond with actual image contents. Some possible improvements to model parameters are also suggested, although performance gains from this is thought to be minimal compared with improvement of the dataset.

6.1 Data Improvements

The thesis project utilizes GRDH SAR products from Sentinel-1 with a 20x22m resolution. Sentinel-1 is chosen since it provides the highest resolution SAR images available free of charge. Sentinel-1 provides higher resolution SAR images than acquired in this project, but product specifications are pre-determined by the mission depending on region, meaning that full resolution images are not always available. With access to adequate funds, these models may be trained and tested on higher resolution SAR imagery, for example from the Radarsat-2 mission, which provides multilooked SAR products at resolutions up to 6.8x7.6m [Tec18]. The rapid development of Earth observing satellites is likely to provide other future alternatives as well, such as NASA's NISAR mission, expected to launch in 2023 [NASb]. With higher resolution SAR images from other sources, streams, small rivers and water bodies will be visible in the images. As such, the models will have the necessary preconditions to map water that is not present in the current SAR data. It likely also increases the ability to detect small-scale changes with higher accuracy. It is notable that altering data source may require modifications of the SAR processing pipeline implemented in this thesis.

Image labels acquired from the Norwegian Mapping Authority are produced with a much higher level of detail than the SAR-images from Sentinel-1. For this reason, they do not always correctly portray the water content in the SAR images. As discussed in Chapter 5, water that is labeled but not present in the images is considered noise and affects the measured performance of the models. To better represent the water that is actually present in the SAR images, water that is invisible should be removed from the masks. Also, masks should be tailored to each individual image to account for seasonal variations of water extents. The discussion briefly elaborates the challenges surrounding hand-labeling the images, although this likely produces the most accurate masks. Some performance may be gained in terms of mapping smaller water extents, but the main limiting factor here is the SAR resolution. However, masks that truly represent their respective images will most certainly increase measured performance. This is because all metrics are calculated by comparing predictions with ground truth, and not the actual image.

One possible solution to improve the masks without hand-labeling is using an iterative segmentation approach. This works by predicting all images and using the predicted masks to train a new model. This may be repeated, and removes water features that are not present, or that is obstructed by image artifacts. Since the models make predictive mistakes, this approach does not ensure perfect labels, but the masks will likely contain less noise and match their respective images better.

Images of all regions used in this thesis are quite similar due to the discussed, homogeneous Norwegian topography. Achieving a model that generalizes well to other, but similar, topographies will most likely require more varied data. A possible solution to this is to produce synthetic data using a General Adverserial Network (GAN) [Sax21]. This technique is not covered in depth here, and is considered as “optional” further work. Synthetic data produced by GANs are interesting if there is a need of a larger training dataset of Norway as well. Producing a larger dataset will result in many very similar images, and thus, data variance may be too low. To solve this, it is possible to only acquire a limited number of images and synthetically produce more varied data from these. Another approach that achieves the same goal is to perform more extensive image augmentation.

6.2 Model Improvements

Two segmentation model architectures are implemented and trained in this thesis project, U-Net and DeepLabV3+. Results prove that both architectures perform well in mapping water extents given the provided data. Slight gains in model performance may be achieved with further exploration of training parameters. As suggested in the discussion, this may include training the models for longer, as both models still seem to converge after the 45 epochs trained in this project, albeit very slowly. Training with a lower batch size with the SGD optimizer may also improve results. Further training parameters, such as the learning rate, may also be tweaked if deemed beneficial. All this, however, is highly experimental and will depend on data and architectures, and it is therefore hard to give definitive suggestions. Special care should be taken if training longer with lower batch sizes, to ensure the model does not overfit and generalize poorly on unseen data.

6.3 Application

If the models or any future iterations of them are to be deployed, it would be useful to convert the predictions to a format that is compatible with existing GIS software. The SAR images are accurately geo-referenced during processing, and can be used as a template to geo-reference the predicted masks. Python libraries such as `GDAL`, `RasterIO`, `shapely` and `GeoPandas` contain methods that handle vector and raster data, which is helpful for converting GeoTIFF image bands to shapefiles [Ger16][EDP13]. Converting the predictions to a shapefile format allows easier comparison to other shapefile masks and data available in GIS. It also enables analysis of the predictions using existing GIS tools and maps. Furthermore, shapefiles are representable in any map projection and CRS, which foregoes the need to re-project the predictions if they are used in projects where these parameters differ from the data used by the models.

Prior to potential deployment, model performance and accuracy should be further verified in terms of flood risk evaluation, flood monitoring and flood management. Verification should encompass extended testing of the models on multiple flood images, as well as time series images of floods where available. Ideally, some sort of ground truth should be available to assess the predictive accuracy of these images. Such verification has been challenging since there has not been adequate time to search for multiple images captured during flood events, and simultaneously check whether these images are captured at the peak of the floods. Additionally, the image and discussion of the flood event in Bismo reveals that proper assessments are difficult due to a lack of ground truth at the specific time of the flood. There is no way of knowing how accurate the predictions are, or how much water was present at the specific time of the captured image. It would therefore be beneficial to further search and test the models on flood images where some comparative truth may exist.

Chapter 7

Conclusion

The current increase in global temperature has contributed to increased rainfall and more unstable and frequent changes in weather. Combined with glacial melting and melting snow, increased precipitation causes more frequent and severe flood events in Norway, a country that is already highly exposed to flooding due to its topography and northern location. Flood activity has the potential to cause massive economical and structural damage and may threaten human lives. Preventative measures are necessary to lessen the consequences of floods, and real time monitoring of ongoing flood events may aid local emergency efforts in their work to save property and human lives.

With rapid technological development, combining Earth observing satellite technology with machine learning proves valuable for automated detection and monitoring of floods and flood hotspots. Radar imaging enables monitoring in all kinds of weather without atmospheric interference. This thesis project leverages deep learning to map water extents by semantic segmentation of Sentinel-1 radar images, aiming to create a proof of concept exploring the possibilities of combining deep learning with remote sensing for flood monitoring, risk assessment and management.

SAR imagery from Sentinel-1 provides the foundation for the dataset created for this thesis. Correctly geo-referenced and noise reduced images are acquired by applying a series of SAR specific processing steps in combination with a high-resolution digital elevation model. Masks are generated from water labels by the Norwegian Mapping Authorities, and finally the dataset of images and masks is tiled and augmented. Two image segmentation model architectures are trained on this data, U-Net with a ResNet50 encoder and DeepLabV3+ with an Xception encoder, both trained to minimize the DICE loss function.

Both architectures are tested on unseen images tiled to the training image size, as well as the fully sized images. While U-Net achieves good results on both tiled images and fully sized images, DeepLabV3+ shows a large decrease in performance when tested on fully sized images. On the tiled images, DeepLabV3+ achieves the best result. This is likely due to DeepLabV3+ being more context dependent than U-Net, which generalizes well on larger images.

On the tiled test data, U-Net achieves a Jaccard index of 0.846 on the water class, while DeepLabV3+ achieves 0.856. The accuracy of the models in predicting the “water”-class is calculated as an additional metric to the Jaccard index. U-Net and DeepLabV3+ achieve accuracies of 88.6% and 89.2%, respectively. As such, both the Jaccard index and class accuracy agree that DeepLabV3+ performs marginally better than U-Net in correctly predicting water pixels. The AUPRC is measured to summarize the precision-recall relationship of the models, and U-Net has an AUPRC of 0.912 and DeepLabV3+ 0.927.

Visual interpretation of predictions shows that DeepLabV3+ is slightly more accurate in detecting narrower rivers that are barely visible in the images. Considering limitations caused by inaccuracies in the masks, as well as the limited spatial resolution of the SAR images, the models are accurate. However, these limitations prevent the models from predicting very narrow rivers and streams, and in detecting very small-scale changes in river extents.

The models that are trained and tested in this thesis can perform change detection of inland water bodies but struggle to detect very slight increases in river extents. The primary limiting factors are available SAR image resolutions and mask inaccuracies. This thesis highlights the possibility of acquiring higher resolution data with adequate funding, while masks may be improved by tailoring masks to their respective images by hand-labeling or iterative segmentation approaches. Given higher resolution SAR imagery for training and testing, results and discussions show that segmentation models may be accurate enough to detect small-scale changes.

This project aims to automate water extent monitoring commonly performed by qualified individuals. In doing so, this thesis proposes methods to aid in flood risk evaluation and flood monitoring that is economically viable and frequently updated. The current versions of the models are likely accurate enough to be used as part of a semi-automatic flood monitoring notification system. Yet, more accurate data is needed to fully realize the potential of a reliable, fully automated system. As new SAR satellite platforms are being launched frequently, it is our hope that such a data source may become freely available to users in the future. If so, this work provides a baseline for automated flood applications aiming to minimize consequences of natural disasters emerging in correlation with global warming.

Appendix A

Dataset Description

This Appendix chapter contains some statistics regarding the data collected from Sentinel-1. The aim is to give readers insight into the distribution of the data. As the thesis briefly discusses, data distribution is related to the variance of the dataset produced from these images. As such, it may play a part in the final model performance achieved in this thesis project. Table A.4 is just an excerpt of the total dataset description available at: `https://github.com/masteroppgave2022/Project/blob/main/dataset_description.csv`.

Table A.1: Amount of Sentinel-1 scenes for each flight direction of the satellite included in the dataset.

Flight direction	**Amount**
Ascending	93
Descending	82

Table A.2: Amount of Sentinel-1 scenes from each of the two satellites per year included in the dataset.

Sentinel-1A	**Year**	**Amount**
	2019	32
	2020	51
	2021	1
Sentinel-1B	**Year**	**Amount**
	2019	34
	2020	54
	2021	3

Table A.3: Amount of Sentinel-1 scenes from each of the two satellites per month and year included in the dataset.

	S-1A			**S-1B**		
Year	2019	2020	2021	2019	2020	2021
January	-	1	-	-	-	1
February	-	1	-	-	-	1
March	-	-	-	-	1	1
April	-	1	1	-	-	-
May	-	-	-	-	1	-
June	-	1	-	-	-	-
July	-	1	-	-	-	-
August	-	1	-	-	2	-
September	32	37	-	34	42	-
October	-	-	-	-	1	-
November	-	1	-	-	-	-
December	-	7	-	-	7	-

Data set description. AM - Acquisition Mode, PRF - Pulse Repetition Frequency, RF - Radar Frequency

	Name
0	S1A_IW_GRDH_1SDV_20210429T165515_20210429T1655...
1	S1B_IW_GRDH_1SDV_20210330T165433_20210330T1654...
2	S1B_IW_GRDH_1SDV_20210227T170240_20210227T1703...
3	S1B_IW_GRDH_1SDV_20210129T053826_20210129T0538...
4	S1A_IW_GRDH_1SDV_20201230T165515_20201230T1655...
5	S1A_IW_GRDH_1SDV_20201129T054704_20201129T0547...
6	S1B_IW_GRDH_1SDV_20201030T170244_20201030T1703...
7	S1A_IW_GRDH_1SDV_20200927T163847_20200927T1639...
8	S1A_IW_GRDH_1SDV_20200830T171132_20200830T1711...
9	S1A_IW_GRDH_1SDV_20200727T165514_20200727T1655...
10	S1A_IW_GRDH_1SDV_20200628T164709_20200628T1647...
11	S1B_IW_GRDH_1SDV_20200529T164625_20200529T1646...
12	S1A_IW_GRDH_1SDV_20200429T164706_20200429T1647...

	Width	Height	Orbit	Type	AM	First Line Time	PRF	RF	Size (MB)
0	26615	16675	ASCENDING	GRD	IW	29-APR-2021 16:55:15	1717.1289	5405.0	1696
1	26747	16689	ASCENDING	GRD	IW	30-MAR-2021 16:54:33	1717.1289	5405.0	1706
2	26747	16686	ASCENDING	GRD	IW	27-FEB-2021 17:02:40	1717.1289	5405.0	1706
3	26781	16681	DESCENDING	GRD	IW	29-JAN-2021 05:38:26	1717.1289	5405.0	1708
4	26622	16675	ASCENDING	GRD	IW	30-DEC-2020 16:55:15	1717.1289	5405.0	1697
5	26544	16665	DESCENDING	GRD	IW	29-NOV-2020 05:47:04	1717.1289	5405.0	1691
6	26747	16686	ASCENDING	GRD	IW	30-OCT-2020 17:02:44	1717.1289	5405.0	1706
7	26606	16672	ASCENDING	GRD	IW	27-SEP-2020 16:38:47	1717.1289	5405.0	1695
8	25454	16648	ASCENDING	GRD	IW	30-AUG-2020 17:11:32	1717.1289	5405.0	1619
9	26615	16675	ASCENDING	GRD	IW	27-JUL-2020 16:55:14	1717.1289	5405.0	1696
10	26638	16673	ASCENDING	GRD	IW	28-JUN-2020 16:47:09	1717.1289	5405.0	1698
11	26551	16674	ASCENDING	GRD	IW	29-MAY-2020 16:46:25	1717.1289	5405.0	1692
12	26639	16674	ASCENDING	GRD	IW	29-APR-2020 16:47:06	1717.1289	5405.0	1698

Table A.4: Data set description. AM - Acquisition Mode, PRF - Pulse Repetition Frequency, RF - Radar Frequency.

Bibliography

[Adm21] ASF Admin. *Asf_search*. GitHub repository. GitHub, 2021. URL: https://github.com/asfadmin/Discovery-asf_search.

[Ant+21] Oleg Antropov et al. "Wide-Area Land Cover Mapping with Sentinel-1 Imagery using Deep Learning Semantic Segmentation Models". In: *IEEE Journal of Selected Topics in Applied Earth Observation and Remote Sensing* 14 (Sept. 2021), pp. 10357–10374. ISSN: 2151-1535. DOI: 10.1109/JSTARS.2021.3116094. URL: https://ieeexplore.ieee.org/document/9551783.

[Arca] ArcGIS. *Sentinel-1 Radiometric Calibration*. URL: https://pro.arcgis.com/en/pro-app/latest/help/analysis/raster-functions/sentinel-1-radiometric-calibration.htm.

[Arcb] ArcGIS. *Speckle function*. URL: https://pro.arcgis.com/en/pro-app/latest/help/analysis/raster-functions/speckle-function.htm.

[BK15] Jimmy Lei Ba and Diederik P. Kingma. "ADAM: A Method for Stochastic Optimization". In: (2015). ISSN: 1412.6980. URL: https://arxiv.org/pdf/1412.6980.pdf.

[Bro18] Jason Brownlee. *How to Use ROC Curves and Precision-Recall Curves for Classification in Python*. Machine Learning Mastery, Aug. 2018. URL: https://machinelearningmastery.com/roc-curves-and-precision-recall-curves-for-classification-in-python/.

[Bus+20] Alexander Buslaev et al. "Albumentations: Fast and Flexible Image Augmentations". In: *Information* 11.2 (2020). ISSN: 2078-2489. DOI: 10.3390/info11020125. URL: https://www.mdpi.com/2078-2489/11/2/125.

[Che+] Liang-Chieh Chen et al. *Encoder-Decoder with Atrous Separable Convolution for Semantic Image Segmentation*. Google Inc. URL: https://arxiv.org/pdf/1802.02611.pdf.

[Cho17] François Chollet. "Xception: Deep Learning with Depthwise Separable Convolutions". In: (Apr. 2017). ISSN: 1610.02357. DOI: 10.48550. URL: https://arxiv.org/abs/1610.02357.

[EDP13] Jared Erickson, Cort Daniel, and Michael Payne. *Polygonize a Raster Band*. GDAL Documentation, 2013. URL: http://pcjericks.github.io/py-gdalogr-cookbook/raster_layers.html#polygonize-a-raster-band.

[Ekm21] Magnus Ekman. *Learning Deep Learning*. Pearson, 2021. ISBN: 978-0-13-747035-8.

[ESAa] ESA. *Copernicus in Detail.* Copernicus: Europe's eyes on Earth. URL: https://www.copernicus.eu/en/about-copernicus/copernicus-detail.

[ESAb] ESA. *Level-1 Ground Range Detected.* Product Description. European Space Agency. URL: https://sentinels.copernicus.eu/web/sentinel/user-guides/sentinel-1-sar/resolutions/level-1-ground-range-detected.

[ESAc] ESA. *Sentinel-1 SAR User Guide.* European Space Agency. URL: https://sentinels.copernicus.eu/web/sentinel/user-guides/sentinel-1-sar.

[ESAd] ESA. *SNAP.* European Space Agency. URL: https://earth.esa.int/eogateway/tools/snap.

[ESA17] ESA. *Thermal Denoising of Products Generated by the S-1 IPF.* S-1 Mission Performance Centre, 2017. URL: https://sentinel.esa.int/documents/247904/2142675/Thermal-Denoising-of-Products-Generated-by-Sentinel-1-IPF.

[ESA20] ESA. *How to use the SNAP API from Python.* European Space Agency, 2020. URL: https://senbox.atlassian.net/wiki/spaces/SNAP/pages/19300362/How+to+use+the+SNAP+API+from+Python.

[Fis+03] Robert Fisher et al. *Contrast Stretching.* HIPR2, 2003. URL: https://homepages.inf.ed.ac.uk/rbf/HIPR2/stretch.htm.

[FL99] Giorgio Franceschetti and Riccardo Lanari. *Synthetic Aperture Radar Processing.* Taylor Francis Group, 1999, p. 3. ISBN: 0849378990.

[Gar+21] Rajat Garg et al. *Semantic segmentation of PolSAR image data using advanced deep learning model.* Electronic Distribution: https://doi.org/10.1038/s41598-021-94422-y. Nature, 2021. URL: https://www.nature.com/articles/s41598-021-94422-y.pdf.

[Geo22] Geovekst. *SOSI-standardisert produktspesifikasjon: FKB-Vann 5.0.* GeoNorge, 2022. URL: https://sosi.geonorge.no/produktspesifikasjoner/FKB-Vann/5.0/#truegenerelt-om-spesifikasjonen.

[Ger16] Jeffrey Gerard. *rasterio_polygonize.py.* GitHub Inc., 2016. URL: https://github.com/rasterio/rasterio/blob/fb93a6425c17f25141ad308cee7263d0c491a0a9/examples/rasterio_polygonize.py.

[Gup+21] Aman Gupta et al. *Adam vs. SGD: Closing the Generalization Gap on Image Classification.* OPT2021, 2021. URL: https://opt-ml.org/papers/2021/paper53.pdf.

[Hal09] Lena Halounová. *Light SAR 1, Radar Basics.* ESA Advanced Training, 2009. URL: https://earth.esa.int/landtraining09/D1La1_Halounova_SARBasics.pdf.

[Her+20] Kelsey Herndon et al. *What is Synthetic Aperture Radar?* Earth Science Data Systems, 2020. URL: https://earthdata.nasa.gov/learn/backgrounders/what-is-sar.

[Hey19] Shannon Heyck-Williams. *Climate Change, Natural Disasters, and Wildlife.* National Wildlife Federation, Nov. 2019. URL: https://www.nwf.org/-/media/Documents/PDFs/Environmental-Threats/Climate-Change-Natural-Disasters-fact-sheet.ashx.

[Jan21] E. David Jansing. *Introduction to Synthetic Aperture Radar Concepts and Practice.* McGraw Hill, 2021, p. 20. ISBN: 9781260458978.

[Kes+17] Nitish Shirish Keskar et al. *On Large-Batch Training for Deep Learning: Generalization Gap and Sharp Minima.* ICLR 2017, Feb. 2017. URL: https://arxiv.org/pdf/1609.04836.pdf.

[KG22] Kartverket and Geovekst. *FKB-Vann.* GeoNorge, 2022. URL: https://kartkatalog.geonorge.no/metadata/fkb-vann/595e47d9-d201-479c-a77d-cbc1f573a76b.

[Leu22] Kenneth Leung. *Micro, Macro Weighted Averages of F1 Score, Clearly Explained.* Towards Data Science, Jan. 2022. URL: https://towardsdatascience.com/micro-macro-weighted-averages-of-f1-score-clearly-explained-b603420b292f.

[Lil+19] Reidar L. Lillestøl et al. *Anvendelser av SAR-bilder fra satellitter over land.* Forsvarets forskningsinstitutt, 2019. ISBN: 978-82-464-3231-1. URL: https://publications.ffi.no/nb/item/asset/dspace:6420/19-01695.pdf.

[Liu+15] Wei Liu et al. *ImageNet Large Scale Visual Recognition Challenge 2015 (ILSVRC2015.* ImageNet, 2015. URL: https://image-net.org/challenges/LSVRC/2015/index.php.

[Liu16] Chang Liu. *Analysis of Sentinel-1 SAR data for mapping standing water in the Twente region.* University of Twente, 2016. URL: https://webapps.itc.utwente.nl/librarywww/papers_2016/msc/wrem/cliu.pdf.

[Moh21] Nibedita Mohanta. *How many satellites are orbiting the Earth in 2021?* Geospatial World, May 2021. URL: https://www.geospatialworld.net/blogs/how-many-satellites-are-orbiting-the-earth-in-2021/.

[NASa] NASA. *Atmospheric Windows.* ICC Durham University. URL: http://www.icc.dur.ac.uk/~tt/Lectures/Galaxies/Images/Infrared/Windows/irwindows.html.

[NASb] NASA. *NISAR: NASA-ISRO SAR Mission.* NASA.gov. URL: https://nisar.jpl.nasa.gov.

[Nat21] The United Nations. *The 17 Sustainable Development Goals.* From The United Nations Web Resource. Nov. 2021. URL: https://sdgs.un.org/goals.

[NHH15] Hyeonwoo Noh, Seunghoon Hong, and Bohyung Han. "Learning Deconvolution Network for Semantic Segmentation". In: (May 2015). ISSN: 1505.04366. URL: https://arxiv.org/pdf/1505.04366.pdf.

[OA10] Asil Ozdarici and Zuhal Akyurek. *A comparison of SAR filtering techniques on agricultural area identification.* ASPRS Conference, Apr. 2010. URL: http://www.asprs.org/a/publications/proceedings/sandiego2010/sandiego10/Ozdarici.pdf.

[Obi+] Cristopher B. Obida et al. *River Network Delineation from Sentinel-1 SAR Data.* Lancaster University. URL: https://eprints.lancs.ac.uk/id/eprint/135037/1/JAG_preprint.pdf.

[PM17] Evelyn Marcia Possa and Philippe Maillard. "Precise Delineation of Small Water Bodies from Sentinel-1 Data using Support Vector Machine Classification". In: *Canadian Journal of Remote Sensing* 44.3 (Sept. 2017), pp. 179–190. DOI: 10.1080/07038992.2018.1478723. URL: https://www.tandfonline.com/doi/full/10.1080/07038992.2018.1478723.

[Pod18] Erika Podest. *Basics of Synthetic Aperture Radar*. 2018. URL: https://appliedsciences.nasa.gov/join-mission/training/english/arset-introduction-synthetic-aperture-radar.

[Rao+] G. Srinivasa Rao et al. "Advantage of Multi-polarized SAR data for Flood Extent Delineation". In: *SPIE* 6410.3 (). ISSN: 64100Z-10. URL: https://www.researchgate.net/publication/257137342_Advantage_of_multi-polarized_SAR_data_for_flood_extent_delineation_-_art_no_64100Z.

[RFB] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. *U-Net: Convolutional Networks for Biomedical Image Segmentation.* accessed 18. november 2021. Computer Science Department and BIOSS Centre for Biological Signalling Studies, University of Freiburg, Germany. URL: https://arxiv.org/pdf/1505.04597.pdf.

[RSH10] Mark A. Richards, James A. Scheer, and William A. Holm. *Principles of Modern Radar, Vol. I: Basic Principles.* SciTECH Publishing, 2010, pp. 4, 219, 835–888. ISBN: 9781891121524.

[San+16] Ardhi Wicaksono Santoso et al. *Comparison of Various Speckle Noise Reduction Filters on Synthetic Aperture Radar Image.* International Journal of Applied Engineering Research, 2016. URL: https://www.researchgate.net/publication/306285628_Comparison_of_Various_Speckle_Noise_Reduction_Filters_on_Synthetic_Aperture_Radar_Image.

[Sar09] Sarmap. *Synthetic Aperture Radar and SARscape.* 2009. URL: https://www.sarmap.ch/pdf/SAR-Guidebook.pdf.

[Sav] Savyakhosla. *Introduction to Pooling Layer.* Geeks for Geeks. URL: https://www.geeksforgeeks.org/cnn-introduction-to-pooling-layer/.

[Sax21] Pawan Saxena. *Synthetic Data Generation Using Conditional-GAN.* Towards Data Science, Aug. 2021. URL: https://towardsdatascience.com/synthetic-data-generation-using-conditional-gan-45f91542ec6b.

[Sci97] Atlantis Scientific. *Theory of Synthetic Aperture Radar.* Atlantis Scientific Inc., 1997. URL: https://www.geo.uzh.ch/~fpaul/sar_theory.html.

[Sky21] Skywatch. *Top 9 free sources of satellite data.* Skywatch Space Applications Inc., 2021. URL: https://www.skywatch.com/blog/free-sources-of-satellite-data.

[SL09] Marina Sokolova and Guy Lapalme. "A Systematic Analysis of Performance Measures for Classification Tasks". In: *Information Processing and Management* 45 (2009).

[Sma11] Davis Small. *Flattening Gamma: Radiometric Terrain Correction for SAR Imagery.* IEEE Transactions on Geoscience and Remote Sensing, Aug. 2011. URL: https://www.geo.uzh.ch/microsite/rsl-documents/research/publications/peer-reviewed-articles/201108-TGRS-Small-tcGamma-3809999360/201108-TGRS-Small-tcGamma.pdf.

[Spa22] SpaceNet. *SpaceNet: Accelerating Geospatial Machine Learning.* spacenet, 2022. URL: https://spacenet.ai.

[SS19] David Small and Adrian Schuber. *Guide to Sentinel-1 Geocoding.* University of Zurich, 2019. URL: https://sentinel.esa.int/documents/247904/1653442/Guide-to-Sentinel-1-Geocoding.pdf.

[Sta18] StanPro. *The Reflection of Light: What is Specular Reflection?* Standard Products Inc., 2018. URL: https://www.standardpro.com/what-is-specular-reflection/.

[Ste16] Stephanie. *Jaccard Index / Similarity Coefficient.* 2016. URL: https://www.statisticshowto.com/jaccard-index/.

[Tec18] Maxar Technologies. *Radarsat-2 Product Description.* Technical Specification. Maxar Technologies Ltd., Sept. 2018. URL: https://earth.esa.int/eogateway/documents/20142/0/Radarsat-2-Product-description.pdf/f2783c7b-6a22-cbe4-f4c1-6992f9926dca.

[Tom20] Nikhil Tomar. *Data Augmentation for Semantic Segmentation.* Medium, Oct. 2020. URL: https://nikhilroxtomar.medium.com/data-augmentation-for-semantic-segmentation-deep-learning-idiot-developer-e2b58ef5232f.

[UU] Stanford University and Princeton University. *About ImageNet.* Stanford Vision Lab, Princeton University. URL: https://image-net.org/about.php.

[VA21] Mikael Vågen and Renate Askevold. *Semantic Segmentation of Satellite Imagery to Monitor Changes in Land Cover.* Project Report – TFE4580. Department of Electronic Systems, NTNU – Norwegian University of Science and Technology, Dec. 2021.

[Wan08] Bu-Chin Wang. *Digital Signal Processing Techniques and Applications in Radar Image Processing.* Wiley, 2008, p. xiii. ISBN: 2008004941.

[Wol] Christian Wolff. *Synthetic Aperture Radar.* URL: https://www.radartutorial.eu/20.airborne/ab07.en.html.

[Yak19] Pavel Yakubovskiy. *Segmentation Models.* GitHub, 2019. URL: https://github.com/qubvel/segmentation_models.

[You18] Kok Yeow You. “Emerging Microwave Technologies in Industrial, Agricultural, Medical and Food Processing”. In: IntechOpen, Jan. 2018. Chap. Introductory Chapter: RF/Microwave Applications. URL: https://www.intechopen.com/chapters/58958.

[Zak21] Emil Zakirov. *Keras Implementation of Deeplabv3+.* GitHub Inc., 2021. URL: https://github.com/bonlime/keras-deeplab-v3-plus.