KTH ROYAL INSTITUTE
OF TECHNOLOGY

Degree Project in Computer Science and Engineering

Second cycle, 30 credits

Delineation of vegetated water through pre-trained convolutional networks

JOHANNA HANSEN

Stockholm, Sweden, 2024

Delineation of vegetated water through pre-trained convolutional networks

JOHANNA HANSEN

Master's Programme, Computer Science, 120 credits
Date: January 10, 2024

Supervisors: Mats Nordahl, Francisco J. Peña
Examiner: Pawel Herman
School of Electrical Engineering and Computer Science
Host company: Stockholm University
Swedish title: Konturteckning av vegeterat vatten genom förtränade konvolutionella nätverk

Abstract

In a world under the constant impact of global warming, wetlands are decreasing in size all across the globe. As the wetlands are a vital part of preventing global warming, the ability to prevent their shrinkage through restorative measures is critical. Continuously orbiting the Earth are satellites that can be used to monitor the wetlands by collecting images of them over time. In order to determine the size of a wetland, and to register if it is shrinking or not, deep learning models can be used. Especially useful for this task is convolutional neural networks (CNNs). This project uses one type of CNN, a U-Net, to segment vegetated water in satellite data. However, this task requires labeled data, which is expensive to generate and difficult to acquire. The model used therefore needs to be able to generate reliable results even on small data sets. Therefore, pre-training of the network is used with a large-scale natural image segmentation data set called Common Objects in Context (COCO). To transfer the satellite data into RGB images to use as input for the pre-trained network, three different methods are tried. Firstly, the commonly used linear transformation method which simply moves the value of radar data into the RGB feature space. Secondly, two convolutional layers are placed before the U-Net which gradually changes the number of channels of the input data, with weights trained through backpropagation during the fine-tuning of the segmentation model. Lastly, a convolutional auto-encoder is trained in the same way as the convolutional layers. The results show that the autoencoder does not perform very well, but that the linear transformation and convolutional layers methods each can outperform the other depending on the data set. No statistical significance can be shown however between the performance of the two latter. Experimenting with including different amounts of polarizations from Sentinel-1 and bands from Sentinel-2 showed that only using radar data gave the best results. It remains to be determined whether one or both of the polarizations should be included to achieve the best result.

Keywords

Wetland delineation, Satellite image segmentation, Convolutional neural networks, Pre-training, Deep learning, Remote sensing

Sammanfattning

I en värld som ständigt påverkas av den globala uppvärmningen, minskar våtmarkerna i storlek över hela världen. Eftersom våtmarkerna är en viktig del i att förhindra global uppvärmning, är förmågan att förhindra att de krymper genom återställande åtgärder kritisk. Kontinuerligt kretsande runt jorden finns satelliter som kan användas för att övervaka våtmarkerna genom att samla in bilder av dem över tid. För att bestämma storleken på en våtmark, i syfte att registrera om den krymper eller inte, kan djupinlärningsmodeller användas. Speciellt användbar för denna uppgift är konvolutionella neurala nätverk (CNN). Detta projekt använder en typ av CNN, ett U-Net, för att segmentera vegeterat vatten i satellitdata. Denna uppgift kräver dock märkt data, vilket är dyrt att generera och svårt att få tag på. Modellen som används behöver därför kunna generera pålitliga resultat även med små datauppsättning. Därför används förträning av nätverket med en storskalig naturlig bildsegmenteringsdatauppsättning som kallas Common Objects in Context (COCO). För att överföra satellitdata till RGB-bilder som ska användas som indata för det förtränade nätverket prövas tre olika metoder. För det första, den vanliga linjära transformationsmetoden som helt enkelt flyttar värdet av radardatan till RGB-funktionsutrymmet. För det andra två konvolutionella lager placerade före U-Net:et som gradvis ändrar mängden kanaler i indatan, med vikter tränade genom bakåtpropagering under finjusteringen av segmenteringsmodellen. Slutligen tränade en konvolutionell auto encoder på samma sätt som de konvolutionella lagren. Resultaten visar att auto encodern inte fungerar särskilt bra, men att metoderna för linjär transformation och konvolutionella lager var och en kan överträffa den andra beroende på datauppsättningen. Ingen statistisk signifikans kan dock visas mellan prestationen för de två senare. Experiment med att inkludera olika mängder av polariseringar från Sentinell-1 och band från Sentinell-2 visade att endast användning av radardata gav de bästa resultaten. Om att inkludera båda polariseringarna eller bara en är den mest lämpliga återstår fortfarande att fastställa.

Nyckelord

Avgränsning av våtmarker, Segmentering av satellitbilder, Konvolutionella neurala nätverk, Förträning, Djupinlärning, Fjärranalys

Contents

List of Figures

List of Tables

List of acronyms and abbreviations

BCE	Binary Cross Entropy
CNN	Convolutional Neural Networks
DEM	Digital Elevation Model
DSM	Digital Surface Model
DTM	Digital Terrain Model
IoU	Intersection over Union
MSI	Multispectral Imagery
NDWI	Normalized Difference Water Index
RGB	Red, Green and Blue
SAR	Synthetic Aperture Radar

Chapter 1

Introduction

Environmental monitoring is a field that is getting more and more attention within deep learning. Data about Earth is constantly being collected by satellites and through neural networks the data can be processed to draw conclusions. This technique is being used for tasks such as monitoring the air quality in urban areas [1], tracking the progression of wildfires [2], and keeping count of the number of individuals belonging to endangered species [3].

This project works on the task of wetland monitoring. Since wetlands fulfill many vital environmental regulation tasks such as carbon dioxide absorption, water cleansing, and flood prevention [4, 5], we as humans have an interest in ensuring that they are sustained. However, due to changes in the environment and human behavior, Earth's wetlands are starting to dry up [6]. Restorative measures exist that can help wetlands recover if they start to shrink [7], but this requires knowing what wetlands are in danger. A neural network can help with this task by allowing data to be gathered on the size of a wetland over time. As a result, shrinking wetlands can be discovered early and restoration can begin. This project aims to develop a model for determining the size of a wetland trained on a few wetlands in Sweden but capable of delineating most or all wetlands in the country.

Inland wetlands often consist of some amount of vegetated water, that is, some of the water in the wetland is covered by vegetation [8]. The task of determining the size of such a wetland is known as delineation of vegetated water. Determining the surface water extent of an open body of water is a simple task that does not require machine learning. A measurement called Normalized Difference Water Index (NDWI) [9] that uses the green and near-infrared bands of a satellite collecting optical data can be used for this task.

However, when vegetation covers the water, NDWI will not be able to find it. Therefore both machine learning and more data are needed to solve the task at hand. In order to delineate something using deep learning, a technique called semantic segmentation is typically used. This method is similar to image classification (a task where the subject of an image is determined) but differs in that the network classifies each pixel of an image separately, instead of assigning one label to the entire image.

Apart from monitoring wetlands, the work done in this project can also be used by researchers working with satellite data. The type of data used in this project can also be applied to many other Earth monitoring tasks, and efficient pre-training strategies can be helpful for any task where only a small amount of data can be gathered or labeled.

1.1 Problem

Labeling satellite data for delineation is a strenuous and time-consuming process. As a result, obtaining a large data set for the delineation of vegetated water is not feasible. There are several well-known methods for working around smaller data sets, and this project investigates one of them - pre-training.

In image segmentation tasks, pre-training is often done on large, publicly available data sets of Red, Green and Blue (RGB) images. This project examines how best to convert the satellite data into RGB images in order to get the highest accuracy on the task at hand and what information from the satellites Sentinel-1 and Sentinel-2 should be included when training. Sentinel-1 captures Synthetic Aperture Radar (SAR) data while Sentinel-2 captures Multispectral Imagery (MSI) data. For further explanation of the types of data collected by the satellites, see chapter 2.2.1.

The research questions that are examined by this project are:

1. *Out of the methods for converting satellite data into RGB images: mathematical linear transformation, stacked convolutional layers, and autoencoder, which leads to the highest accuracy of vegetated water delineation on a pre-trained network?*

2. *Which combination of SAR and MSI data used for fine-tuning a pre-trained network out of: only VH polarization data, both VH and VV polarization data, both VH and VV polarization data along with the R, G, and B bands from Sentinel-2, both VH and VV polarization data*

along with bands 1 through 9 from Sentinel-2, leads to the highest accuracy of vegetated water delineation?

1.2 Purpose

The purpose of the thesis project is to facilitate wetland restoration efforts by developing a tool for wetland monitoring that can help identify shrinking wetlands. Additionally, the research conducted can be applied not only to wetlands but the field of earth monitoring as a whole.

Through the lens of deep learning, the analysis of satellite data is explored. By investigating methods for converting satellite data into RGB images and optimizing the combination of SAR and MSI data, the work contributes to aerial imaging processing research and aids in finding processing methods that require smaller data sets. As large-scale labeled wetland data sets are rare, the research questions posed in this thesis are closely related to the improvement of wetland monitoring tools.

1.3 Objectives

In order to answer the research questions, the following objectives need to be met:

- Construct an annotated data set of wetlands containing both Sentinel-1 and Sentinel-2 data.
- Train and evaluate a baseline and three additional models using different methods for converting the satellite data into RGB images.
- Compare the four models to each other and analyze the results seen.
- Train and evaluate four models with the same architecture using different input data from Sentinel-1 and Sentinel-2.
- Compare the four new models to each other and analyze the results seen.
- Draw conclusions based on the results and analysis.

1.4 Delimitations

The project focuses on delineation based on satellite images, and the ground truth is labeled by hand by looking at satellite radar data. This means that there might be some discrepancies between where the water is indicated in the data and where the water actually was present during the time the data was collected. Using labels from measurements collected on site would have led to more accurate labels and the possibility for the machine learning models to learn details from the satellite data that are overlooked by humans. However, creating such a data set for this project would not have been feasible.

The data used is taken from four wetlands in Sweden. This means that the model might not generalize to other bodies of vegetated water in other areas of the world with different vegetation and climate. The model also would not work well with coastal wetlands, as these often contain trees that the data used can not see through to find water.

The project does not consider a multitude of different architectures for pre-training. One model that has shown to work well on similar tasks is used, but better results could possibly be obtained by experimenting with the base architecture.

1.5 Structure of the Thesis

Chapter 2 of the thesis describes all the background information for the project. Chapter 3 describes the method used to answer the research question. A presentation of the results obtained is given in chapter 4 along with an analysis of those results. Finally, chapter 5 includes a discussion and presents the conclusions and possible future work.

Chapter 2

Background

This chapter provides the necessary background for understanding the work done during the project. Firstly, the machine learning process of semantic segmentation is described along with common architectures, evaluation metrics, and loss functions used when performing that task. Next, the type of data used for the project is described along with the processing of it. Lastly, other works which tie into or have an impact on this project are introduced.

2.1 Semantic segmentation

Semantic segmentation is a machine-learning task derived from image classification. Image classification means to determine what object is in an image. Similarly, semantic segmentation determines what object each pixel in the image belongs to. This project deals with binary semantic segmentation, meaning that each pixel in the image can either be classified as foreground (in this case water) or background meaning everything else.

In deep learning, three main aspects have to be selected before training is started. These are the architecture of the network, the evaluation metric, and the loss function. In the following sections, options for each of these aspects when performing semantic segmentation are introduced.

2.1.1 Architectures

This section introduces two common architectures used for semantic segmentation. Since there is a wide variety of architectures available, this section has been narrowed down to those that are most often used, are not too deep, and are based on Convolutional Neural Networks (CNN) structures.

CNNs are networks designed to work specifically with image data. They work by using filters, or convolutions, to search for certain shapes in the image. The deeper layers of the network have filters that search for more abstract shapes than the shallower layers.

2.1.1.1 U-Net

The U-Net [10] was introduced in 2015 as an image segmentation network for biomedical imagery, but has since found uses in most areas of semantic segmentation [11]. It specifically works well with smaller data sets [12]. The U-Net consists of an encoder and a decoder section with skip connections between them, see figure 2.1. The encoder section consists of stacked blocks of two 3x3 convolutions and a 2x2 max pooling layer. The convolutions are the filters looking for features in the images with a kernel size of 3, meaning that each filter consists of 3x3 pixels. The max pooling layers reduce the size of the image by only keeping the highest value in each 2x2 pixel area. The decoder section mirrors the encoder section, with stacked blocks built of a 2x2 up-convolutional layer followed by two 3x3 convolutional layers. The arrows marked as ”copy and crop” refer to the skip connections. These help ensure that some of the information regarding the location of pixels is brought into the decoder section, which would otherwise be lost due to the downsampling. The final layer of the U-Net consists of a 1x1 convolution which results in the segmentation output. As can be seen in the figure 2.1, the input image is larger than the output image. This is because, when originally proposed, no padding was used. Instead, the input images were segments of a larger image, and some data from the neighboring segments were included in each input data. This meant that the input image was larger than the label. However, the U-Net can also successfully be applied with padding, ensuring that input and label can keep the same size.

2.1.1.2 SegNet

The SegNet [13] has a very similar structure to the U-Net. It too consists of an encoder and decoder section with stacked blocks of either up-convolutions or max pooling layers along with convolutional layers, see figure 2.2. The softmax layer shown in the figure refers to a function that creates a normalized probability distribution of the output data.

The main difference is that instead of using skip connections which copy the data from the encoder to decoder blocks, the SegNet passes along the pooling indices. What this means is that in the max-pooling layers, which for

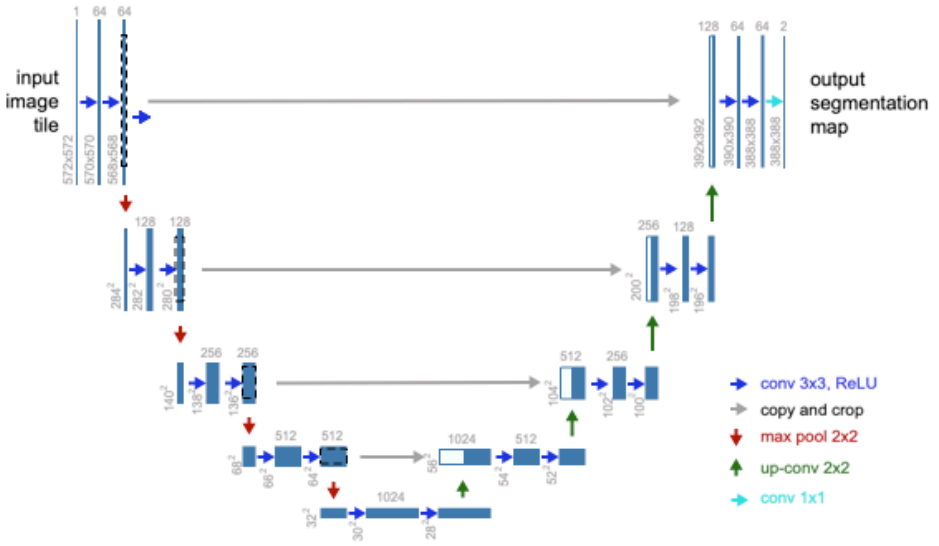


Figure 2.1: Architecture of the U-Net (image based on [10])

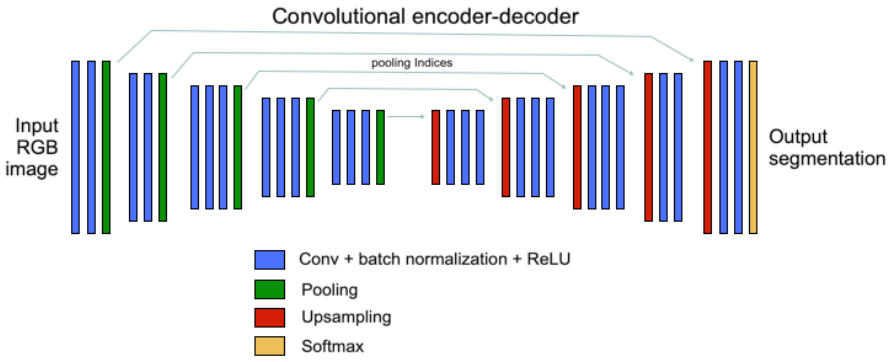


Figure 2.2: Architecture of the SegNet (image based on [13])

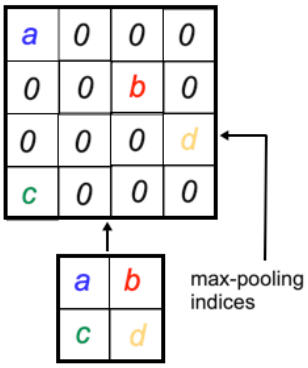


Figure 2.3: The upsampling method of the SegNet (image based on [13])

each 2x2 matrix of pixels in the image selects the pixel with the highest value to use in the new, smaller image, the index in which the highest value was found is remembered. In the up-convolution layers, for each 2x2 matrix of pixels, a new matrix of 4x4 pixels consisting of only zeros is formed. The pooling indices are then used to determine where the values from the 2x2 matrix should be placed in the 4x4 matrix. This process is illustrated in figure 2.3. This is a different method of passing along the position information than used in the U-Net and one that uses less memory.

Another difference between the SegNet and the U-Net is that the encoder part of the SegNet is an exact replica of the first 13 layers of the VGG16 [14] (a very common image classification network). Because of this, the creators of the SegNet recommend that it should be initialized with the weights of a pre-trained VGG16 in order to speed up the learning process.

2.1.2 Evaluation metrics

To determine how well the model is performing, an evaluation metric is required. Evaluation metrics provide an objective measurement that allows for comparison between models. Different evaluation metrics measure different parts of the performance and are more or less appropriate depending on the data. This section introduces some evaluation metrics for image segmentation.

2.1.2.1 Pixel accuracy

Pixel Accuracy is the simplest evaluation metric for image segmentation. It calculates the percentage of pixels that have been correctly classified in the

image. The mathematical formula for Pixel Accuracy is defined as follows:

$$PA(y, \hat{y}) = \frac{\sum_{i=1}^{N} 1 - |y_i - \hat{y}_i|}{N} \tag{2.1}$$

where y is the label, $\hat{y}$ is the prediction and N is the number of pixels in the image.

This metric does come with some issues though. In the case where the foreground and background pixels do not take up 50% of the image each, there is a bias in the metric. Given an example where the object being detected only takes up 1% of the image, a network that classifies the entire image as background would have a 99% accuracy. However, this network would be completely useless since it was designed to find that 1% of the image. Because the foreground and background pixels are rarely completely balanced in an image segmentation task, other evaluation metrics are more commonly used [11].

2.1.2.2 Intersection over Union (IoU)

The Intersection over Union (IoU), also known as the Jaccard Index, solves some of the issues present in Pixel Accuracy. Instead of taking the entire image into account, IoU only looks at the pixels that were predicted and/or labeled as foreground. The intersection refers to the pixels that both the prediction and label define as foreground, that is the true positives. The union on the other hand refers to the pixels that one or both of the prediction and label has defined as foreground. This means that the union includes false positives, false negatives, and true positives. When the prediction is perfect, the union should only include true positives, meaning that the IoU equals 1 [15, 11].

$$IoU(y, \hat{y}) = \frac{\sum_{i=1}^{N} y_i \cdot \hat{y}_i}{\sum_{i=1}^{N} y_i + \hat{y}_i - y_i \cdot \hat{y}_i} \tag{2.2}$$

2.1.2.3 Dice score

The Dice score is quite similar to the IoU. It is calculated as follows:

$$Dice(y, \hat{y}) = \frac{2 \cdot \sum_{i=1}^{N} y_i \cdot \hat{y}_i}{\sum_{i=1}^{N} y_i + \hat{y}_i} \tag{2.3}$$

Both IoU and Dice are commonly used evaluation metrics in image segmentation. The difference between them is that IoU places a more

significant emphasis on errors. This means that the Dice score indicates the average behavior of the model while IoU indicates the worst-case behavior [15].

2.1.3 Loss functions

During the training of a neural network, a loss function is required to update the weights. It is used to calculate how good the predictions of the current batch were by comparing the probabilities returned by the softmax function to the labels. The choice of loss function is very important during training, as it determines what the network prioritizes. This section introduces three commonly used loss functions and explains the applications where they are most useful.

2.1.3.1 Binary Cross-Entropy Loss

A neural network outputs a probability of a sample belonging to each of the available classes. Cross-Entropy Loss [16] considers the probability that the sample belongs to the class it was labeled as and computes the negative log of that value. Binary Cross Entropy (BCE) loss means that it is the version of the function applied during binary classification tasks. The mathematical formula for BCE is defined as:

$$BCE(y, \hat{y}) = -y \cdot \log(\hat{y}) - (1 - y) \cdot \log(1 - (\hat{y})) \tag{2.4}$$

In image segmentation, BCE calculates the loss for each pixel and then averages the loss value for the image. Cross-Entropy loss is one of the most commonly used loss functions for image segmentation [11], and has the strength that very large misclassifications lead to a larger step size when updating the weights in the network than when the error is small. BCE is considered a good option for a loss function when the data is very well balanced. There are however a few variants to the standard BCE loss that add weights to the equation above. This is done when the data set is unbalanced to ensure that the harder samples in the data are learned by the network as well as the easy ones. These functions work best with a very unbalanced data set and are not applied in this project [17].

2.1.3.2 Dice Coefficient

The Dice coefficient [18] as a loss function is the other most commonly used loss function for image segmentation and is derived from the Dice score [11]. There are only two differences between the calculation of Dice loss (Coefficient) and Dice evaluation score. Firstly, the Dice loss function uses the probabilities of a pixel belonging to the foreground instead of the final prediction (i.e., 1 or 0). Secondly, the Loss function subtracts the Dice score from 1 since we want a well-performing network to lead to a low loss but a high accuracy.

$$DC(y, \hat{y}) = 1 - \frac{2 \cdot \sum_{i=1}^{N} y_i \cdot \hat{y}_i}{\sum_{i=1}^{N} y_i + \hat{y}_i} \tag{2.5}$$

Generally, using the Dice Coefficient loss function leads to the best network when the data is moderately unbalanced [17].

2.1.3.3 Shape-Aware loss

Shape-Aware loss [19] was used in the publication that introduced the U-Net [10]. It is a variation of Cross-Entropy Loss designed to help the network learn the specific shape of the object being segmented. This is done by adding a coefficient to the Cross-Entropy Loss calculation that depends on the distance between a point and the edge of the object. This loss function works well when the shape of the border between foreground and background is difficult to learn [17]. Shape-Aware loss is calculated as follows:

$$SAL(y, \hat{y}) = - c(x) \cdot (y \cdot \log(\hat{y}) + (1 - y) \cdot \log(1 - (\hat{y}))) \tag{2.6}$$

where $c(x)$ is the coefficient.

2.2 Data

Here, the type of data that is used for the project is introduced along with methods to handle the fact that large amounts of data are not readily available.

2.2.1 Satellite data

Satellites are a great way to gather information about the surface of the Earth. As orbiting satellites continually rotate the Earth, they can collect data from the same site every few days. This means that researchers have access to

continuous updates on any area they would like to study. Collecting the data also does not require any work from humans, which saves time and resources. Different satellites collect different types of data at different time intervals. This section describes the data collected by the two satellites Sentinel-1 and Sentinel-2 which were used during the project.

2.2.1.1 Sentinel-1

SAR is a technique for capturing high-resolution radar images of Earth from satellites. Radars work by sending out microwaves from an antenna. The microwaves bounce against solid objects and reflect back to the radar where the return wave is registered [20]. The waves captured can be used to model the environment and create an image of the objects that the microwaves interacted with. However, in order to capture high-resolution images of Earth from a satellite, an in-feasibly large antenna would be needed. To deal with this issue, Synthetic Aperture Radar was invented in which multiple radar scans, separated by some physical distance, are added together to simulate the effects of a long antenna [21].

Microwaves have a quite large wavelength, which allows them to pass through certain objects instead of bouncing back from them. For instance, they pass through clouds which allows radar images to be captured during most weathers. They also pass through a thin layer of vegetation, which allows radars to detect vegetated water [21]. Water can be detected in SAR images because it scatters the radar waves instead of returning them, leaving the area where water is present darker than the surrounding areas [20].

Sentinel-1 uses a C-band SAR. This refers to the wavelength which in this case is about 5.5 cm. The data that can be collected from Sentinel-1 consists of two polarizations and the angle with which the microwaves were sent toward the Earth. The polarizations refer to the direction the microwaves oscillate when being transmitted and received. The direction can either be horizontal (H) or vertical (V). Both the polarizations of Sentinel-1 transmit using vertical oscillations but one receives vertically oscillating waves (VV) and the other receives horizontal ones (VH). The different polarizations are better at detecting different types of materials. The VV polarization is best at detecting soil and water, whereas VH is best at detecting vegetation. The resolution of the SAR images from Sentinel-1 is 10m, meaning that a 10 m^2 area is covered in one pixel of the image [22, 21].

2.2.1.2 Sentinel-2

Sentinel-2 captures MSI data which means that it gathers imagery of a scene at several different discrete wavelengths. MSI data typically includes the visual spectrum along with the ultra-violet and infrared spectra [23]. Sentinel-2 uses 13 wavelengths, also known as bands [24]. In table 2.1, the description of each band can be found along with its wavelength, resolution, and common uses. Because optical wavelengths are short relative to microwaves, Sentinel-2 data is highly affected by the weather [20]. During a cloudy day, the imagery gathered is not able to represent the ground, but rather show the clouds. During sunny days, only the surface of the ground can be seen, meaning that the waves can not penetrate vegetation to detect water.

2.2.2 Dealing with small data sets

Deep learning is a very useful tool, but it generally requires very big data sets in order to provide good results. For the task worked on in this project, there are no large-scale data sets available. However, some methods are often used to improve the performance when only smaller data sets exist. These methods, known as pre-training and data augmentation, are introduced in this section.

2.2.2.1 Pre-training

Pre-training is a method within machine learning that aims to replicate the way humans can apply knowledge from one task to speed up the learning process for a related task. Normally, neural networks learn to perform one task, such as locating faces in an image. When a new task needs to be solved, say locating humans in an image, an entirely new network is trained that has to learn every aspect of that task from scratch. Transfer learning works by applying the knowledge from the first task to the second one. That way, the second network can immediately know how to detect edges, objects, certain human features, and so on. The only thing that needs to be learned is what components make up a human instead of a face. In pre-training, the first task is known as the source task, and the second one is known as the target. This method saves time when training the target network, leads to higher accuracy in the target task, and allows for a much smaller target data set. It is required that the source and target tasks are within the same domain, meaning that they both deal with the same type of data. For instance, in the example above, both data sets have to consist of images. The more closely related the two tasks are, the more effective transfer learning is. However, it has been shown that

Band no.	Description	Wavelength (nm)	Resolution (m/px)	Good for
1	Coastal aerosol	443	60	Detecting particles in coastal areas
2	Blue	490	10	Differentiating between soil and vegetation, detecting man-made objects
3	Green	560	10	Finding oil on water, discriminating between still and turbulent water
4	Red	665	10	Identifying types of vegetation and soil
5	Vegetation red edge	705	20	Classifying vegetation
6	Vegetation red edge	740	20	Classifying vegetation
7	Vegetation red edge	783	20	Classifying vegetation
8	Near-Infrared	842	10	Classifying vegetation, finding edge between water and land
8a	Vegetation red edge	865	20	Classifying vegetation
9	Water vapor	945	60	Finding water vapour
10	Short wave infrared - Cirrus	1375	60	Detecting clouds consisting of ice crystals
11	Short wave infrared	1610	20	Measure moisture levels in soil, distinguish between cloud and snow
12	Short wave infrared	2190	20	Measure moisture levels in soil, distinguish between cloud and snow

Table 2.1: The description, wavelength, resolution, and uses for each of the bands in Sentinel-2 [25].

even tasks that are not very closely related to each other can still benefit from pre-training [26].

Two different methods of pre-training are commonly used. The first one is called transfer learning and is a method where the network used for the source tasks is re-used completely except for the final layer of the network which performs the classification or regression. When training on the target data set, all of the weights are frozen except for the output layer. This way, the network does not re-learn what structures it is looking for but changes what combination of those structures it is looking for. The second type of pre-training is called fine-tuning. In this method, most or all of the weights of the source network are updated using the target data set. This strategy allows for the target network to adapt more to the specifics of the target data set. By starting training on the weights from the pre-training, the probability of finding the optimal set of weights is higher than when training from scratch on the target data set.

2.2.2.2 Data augmentation

Data augmentation refers to the strategy of creating new data from the existing data set. One way to do this is called data warping and works by creating copies of the existing data and making slight changes to it. [27] When working with images, the changes are typically adding noise, rotating, mirroring, or cropping the images. These types of changes allow for reusing the original label, which is useful as data labeling can be a very difficult and time-consuming task. Depending on the task, different choices of data warping methods are more or less appropriate and they need to be selected to ensure that the data is still useful for the task [28].

Data augmentation can also refer to the generation of entirely new data through a generative model. Generative models are machine learning models that learn the statistical distribution of the data, allowing it to understand how different parts of a sample are related to each other. Because of this, generative models can be used to generate entirely synthetic new data. Synthetic data can be very useful to fill out a data set if it is too small. It does however require a lot of data to train and test a separate model in order to generate the data and generating high-resolution images is very difficult [28].

2.3 Related work

This section describes some previous research done which ties into the research done in this project. Firstly, other works related to wetland mapping are described, followed by works discussing techniques for semantic segmentation in SAR images.

2.3.1 Wetland mapping

Tottrop et al. [29] have created a global map of wetlands using deep learning and data from satellites. They have included SAR, optical, thermal, and Digital Elevation Model (DEM) data in their model. They estimate this technique to be about 70% correct [30]. However, there is no research publication describing the architecture used or how training was performed meaning that it is difficult to replicate or build on their results.

In a project similar to this one, López-Tapia et al. [31] created a method for wetland delineation that they called WetSegNet. They started out working with a residual VGG16 which they later modified. The data consisted of all the wetlands in Illinois manually measured on-site between 2002 and 2018. The areas were captured using RGB and Near-Infrared bands from aerial imagery as well as Digital Surface Models (DSMs) and Digital Terrain Models (DTMs). The last two are created from bouncing laser beams at the earth and are used to create a map of surface features. They achieved an accuracy of 98% for the area under the receiver operating characteristic (AUROC) on their test set. The issue with replicating the method of López-Tapia et al. is that the data set has been collected on-site over a long time period, and that type of data set is rarely available for other areas.

DeepAqua is an unsupervised model for wetland detection developed by Peña et al. and introduced in [32]. Their model is trained on radar data covering the county of Örebro in Sweden. They use a knowledge distillation technique, which utilizes NDWI masks generated on Sentinel-2 data as the teacher model and a U-Net as the student model. The input data to the student model is Sentinel-1 data from the same day and area as the Sentinel-2 data used in the teacher model. The model is evaluated on manually annotated masks from the wetlands Hornborgasjön, Svartådalen, and Hjälstaviken in Sweden. The model achieved an IoU score of 0.94 on Hornborgasjön, 0.88 on Svartådalen, and 0.69 on Hjälstaviken.

2.3.2 Semantic segmentation of SAR images

Wang et al. [33] performed experiments using transfer learning on a data set of SAR images. They were working on classifying types of ships in the SAR images. The pre-training was done on ImageNet, a very large natural image data set for image classification. The SAR images were converted to greyscale RGB images with three channels to fit in the pre-trained network. The method for doing this is not described in the publication, but it was likely done through a linear conversion from amplitude to pixel values, as this is what is done to visualize SAR data. The results of their experiments showed that a smaller network yielded better results on the target data set, even though the results on the source data set were worse, and that fine-tuning the network was more appropriate than transferring the network.

In [34] Gao et al. used a version of the U-Net, where the encoder consisted of a ResNet-34 pre-trained on ImageNet, to segment floating raft aquacultures in Sentinel-1 data. They found that pre-training the network led to a higher IoU than using the same network with random initialization of the weights. It is not stated how they converted the SAR data to fit into the pre-trained network.

SAR data is often visualized as a greyscale image by making a linear or logarithmic transform into the RGB feature space. However, radar data is not the same as optical data and there are some fundamental differences between these two types of images. SAR data is collected using active sensors using microwaves whereas RGB images are collected using passive sensors in the visible spectrum. Additionally, the noise varies greatly between them. SAR data contains speckle noise, a multiplicative kind of noise, whereas RGB images contain additive noise in the form of, for instance, Gaussian noise. In [35] Lv et al. state that the noise in SAR images decreases the effectiveness of segmentation methods. Many attempts have been made to decrease the amount of noise in SAR data, but Singh and Shree reason in [36] that it is not possible to completely remove it. The fact that these types of images are fundamentally different suggests that there might be some information loss when directly converting between the two feature spaces.

To change the feature space of SAR data, autoencoders can be used to learn a representation of the important features of the data. A fully convolutional autoencoder was used for data compression of hyperspectral data by La Grassa et al. in [37], which while being optical data, also consists of aerial imagery. In [38] Dong et al. investigated different autoencoder structures for target recognition in SAR data. They found that autoencoders with a depth of up to 3 layers led to good results, but that increasing the depth further did not

make a large difference. They also experimented with using both linear and convolutional autoencoders. For the linear autoencoders, they found that the code space should optimally be around 1000 nodes for their images of 92x92 pixels. In [39] Song et al. used an autoencoder-like structure to colorize SAR data from one polarization by generating fully polarised data. This was done with a feature extractor consisting of the first seven layers of the VGG-16 and a feature translator with fully connected layers generating nine outputs. In [40] Shakya et al. colorized SAR data by fusing Sentinel-1 and Sentinel-2 data. Their resulting images consist of four channels: red, green, blue, and near-infrared.

Chapter 3

Method

This chapter describes the research method used in the project. First, the data sets used are described, followed by an explanation of the models tested and what experiments were run on each. Next, the steps taken for each experiment are outlined. Lastly, details surrounding the implementation of the code are given.

3.1 Data sets

Three different data sets were used in this project. A source data set was used for pre-training and two target data sets consisting of satellite data of areas containing vegetated water were used for fine-tuning. This section describes the contents of these data sets.

3.1.1 Source data

For pre-training the network, the largest image segmentation data set available was used. It is called COCO [41] (Common Objects in Context) and was constructed by Microsoft. The images are natural, meaning they depict everyday scenes such as animals, living rooms, beaches, etc. The data set consists of 118,287 training images and 5,000 test images. COCO has 80 classes of objects. However, since the target data set only has one foreground class, a binary mask was created where all of the 80 classes in COCO are combined into one class representing any object not belonging to the background. Each image was resized to a size of 64x64 pixels before training since they had to match the size of the images in the target data set.

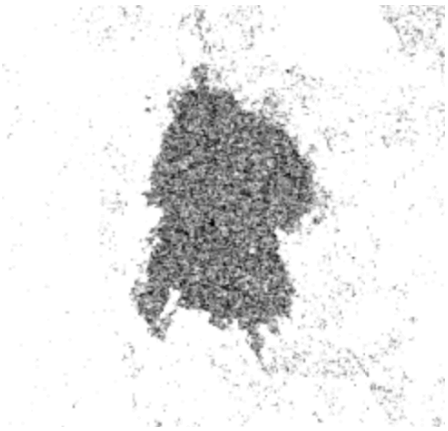
3.1.2 Target data

The target data set consists of data from four wetlands in Sweden, all marked as important to maintain by the Ramsar Convention [42], an international convention on wetlands. The four wetlands are Svartådalen, Mossaträsk, Hjälstaviken, and Hornborgasjön. For each of the wetlands, data was collected once a month, between the months of April and November. The winter months were left out as the presence of snow interfered with the readings. This was done through the years 2014 to 2022.

Every data sample consists of a SAR reading, containing both a VV and VH polarization and a manually annotated mask. Some of the samples also contain an MSI reading, with the first ten bands of Table 2.1. The reason only some of the samples contain optical data is that optical data is highly dependent on weather conditions.

The manual annotations were created in Google Earth Engine, following the process illustrated in Figure 3.1. The annotations were created using the VH polarization as a guide, and outlining the areas where water was present using the polygon tool in Google Earth Engine. Water was determined to be present based on how dark the pixel was, along with the color of the surrounding pixels to find an outline of the body of water. The technique for creating annotations was based on the work of Peña et al. in [32]. In cases where it was difficult to tell if water was present, discussions were held to ensure that the results were as correct as possible. The annotations were made by the author of this thesis and two other thesis students working with the same supervisor. The VH polarization was chosen for annotations as the project's supervisor had found that it leads to better results when training models than VV polarization. The final area outlined was then used to create a binary mask where a pixel value of 1 means water and 0 means anything else. This process is illustrated in figure 3.1.

In total, 230 samples of SAR and annotations were collected, along with 76 MSI samples. Each sample was then split into tiles of 64x64 pixels, as can be seen in figure 3.2. Any tiles on the edges of the images that are less than that size were discarded. The instruments measured some pixel values as NaN (not a number), which disrupted the loss calculations during training. Any tiles that contained a pixel with that value were discarded as well. This resulted in a total number of 35,715 SAR data tiles. All of the tiles from Svartådalen were used for the test set, with a size of 5979 tiles. The tiles from the remaining three areas were randomly split into a training and validation set, with an 80:20 ratio.



(a) The VH polarization data visualized as a greyscale image

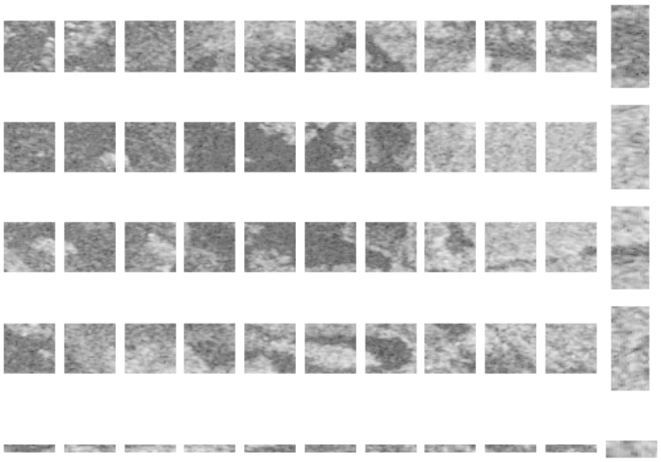
(b) The manual annotation laid on top of the VH polarization data

(c) The binary mask used as the label when training

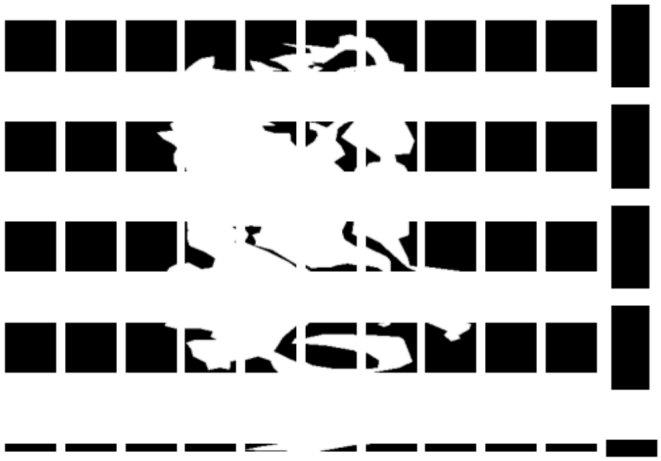
Figure 3.1: The process of annotating the data set and creating the labels

Using cross-validation instead of a validation set was experimented with, and it showed that both methods gave very similar results in terms of hyperparameter tuning. Cross-validation means that the training set is divided up into k sets. The model is then trained and evaluated k times, on k-1 of the sets. Each time, a different set is withheld from training and used to evaluate the model instead. After the model has been trained k times, the average of all the evaluations is calculated and used as the validation performance. Because cross-validation takes much longer, hyperparameter tuning was done using one validation set.

The quality of the annotations varied on the different wetlands, which can have a large impact on the results. Low-quality data makes it more difficult



(a) SAR image



(b) mask

Figure 3.2: One sample of SAR data and mask split into tiles of 64x64 pixels

for the model to understand what it is supposed to learn, leading to poorer results. In some of the wetlands, only water belonging to the largest body of water was marked as water in the annotations. This left out areas that would have been relevant to detect through the models. Some of the wetlands were also given more time for each annotation, which meant that more details could be captured. The Svartådalen annotations are of the highest quality (all bodies of water were included and the most amount of time was given to the annotations) and therefore a second data set for training and evaluating the models was created using only that data. The highest quality refers to the fact the annotations most closely resemble the darker areas in the SAR image which represents water. Creating the Svartådalen data set resulted in a smaller but more accurate data set, the results from which can be compared to the data set with all of the data. In the Svartådalen data set, the training set consists of the tiles collected from April 2014 to May 2020. The validation set uses data from June 2020 to July 2021 and lastly, the test set gets the remaining months from August 2021 to November 2022. The three sets consist of 3822, 980, and 1176 tiles respectively. Because the Svartådalen dataset is trained and evaluated on data from the same area, the results are not an accurate representation of how the models would perform on data from other wetland areas. The results from the models trained on Svartådalen data should only be compared to each other.

Before each epoch of training, the tiles underwent random data augmentations. By a certain probability, determined during hyperparameter tuning, all the tiles in a batch were vertically flipped, horizontally flipped or rotated a random amount between -35 and 35 degrees. This meant that for instance, there could be a 20% chance that the image is rotated, a 50% chance that it is horizontally flipped, and a 70% chance that it is vertically flipped. Several of the augmentations could occur simultaneously. When rotating the image, the contents of the corners were extrapolated by mirroring the nearby contents of the image, and interpolation was done using linear interpolation. The rotation was performed by the function albumentations.Rotate(). The result of the augmentations was that each time a certain data point was passed through the network, it looked slightly different depending on which augmentations were randomly selected at that time. Both input data and labels underwent the same augmentation. The effects of each of these augmentations can be seen in Figure 3.3. These types of data augmentations make the network less sensitive to the orientation of the input data, and thereby its knowledge is more generalized.

(a) Original tile

(b) Horizontally flipped tile

(c) Vertically flipped tile

(d) Rotated tile

Figure 3.3: A mask tile in its original form and after each of the three data augmentations have been applied to it individually

3.2 Project

The research aims to develop a deep learning method for the delineation of vegetated water, based on pre-training on a large data set. Three separate methods were compared to a baseline which was trained on only the target data set (no pre-training). Both the pre-trained models and the baseline used the U-Net as the base architecture for the semantic segmentation. It followed the structure outlined in [10], with the only changes being that padding was used to ensure that the output image was the same size as the input and that the number of input channels varied depending on the data used. The U-Net was selected because it is a model that is effective in many different image segmentation tasks. While the overall performance of the models might increase if different base architectures are used, the research focuses on pre-training and to limit the scope no other architectures were examined.

The same pre-trained U-Net was used for all of the experiments, meaning that all experiments except for the baseline started with the same weights in the base architecture.

Several methods of converting the VH polarization of the SAR data into the RGB feature space before it was used to fine-tune the transferred network were tried through a series of experiments. These methods included both mathematical transformations and neural networks. Each of these methods is described in detail in this section. Once the best transformation method had been selected, based on the segmentation results it generated, the same method was applied while including both polarizations and the optical data. Experiments were run which included different combinations of the radar and optical bands to determine what data should be used to get the best results.

3.2.1 Baseline

To establish a baseline with which to compare later results, a U-Net was trained to perform semantic segmentation on the data set using the VH polarization of the SAR data. Here, only the target data set was used. That is, the model was trained from scratch. The idea behind this baseline was to check if the pre-training improves the final result.

3.2.2 Linear transformation

In [33], where pre-training on natural images was used, the authors mention that they convert the images to three-channel greyscale before passing them

to the network. While they do not specifically describe how they do this conversion, they likely used the same technique as is used for visualizing SAR data. For visualization, the SAR data is linearly transformed to only contain values in the RGB feature space. The linear transformation method is given by:

$$X_t = \frac{X - X_{min}}{X_{max} - X_{min}} \tag{3.1}$$

where X_t is the transformed image, X, is the SAR data, X_{min} is the lowest pixel value found in X and X_{max} is the highest pixel value.

The linearly transformed data is then replicated three times in order to create three channels - as is used by RGB. When all three channels contain the same values, the image comes out as greyscale. The linear conversion must be made before splitting the data into the tiles. Otherwise, pixels containing the same value in the SAR domain that belong to different tiles could end up with different values in the RGB domain. Since this method for processing SAR data when applying transfer learning is popularly used, its results were explored.

This method only takes in one polarization of SAR data. In this work, it is therefore not used to investigate if including more data improves the results.

3.2.3 Convolutional layers

As described in Section 2.1.1, convolutional neural network structures work like filters moving over an image looking for specific properties. They can be used both to change the number of pixels in an image, and to change the number of channels, and are commonly used for feature extraction [43]. The convolutional layers were placed before the U-Net, essentially making the output from the convolutional layers the input into the U-Net. They were trained at the same time as the fine-tuning of the U-Net. This was done by backpropagating the loss from the segmentation through the U-Net and into the convolutional layers.

Experimentation with the performance on the validation set showed that a structure of two convolutional layers with a ReLU activation function after each gave the best performance. Both convolutional layers kept the size of the image the same by using a stride of 1 and setting padding to 'same'. The first convolutional layer increased the number of channels from one to two and the second convolutional layer outputs three channels. The kernel size used was determined by the hyperparameter-tuning. The output from the second ReLU function was used as the input for the pre-trained U-Net.

3.2.4 Autoencoder Structure

Autoencoders are a type of neural network which in its most basic form is used to replicate the input data. This means that, if an image is given to the autoencoder, the goal is for the network to output the exact same image. It does this by creating a representation of the image using a lot fewer data points than the total number of pixels in the image and then recreating the image from that representation. The representation is created using an encoder, and the recreation is done using a decoder. Since the goal here is to take an image and create a new representation of that image in a different feature space, an autoencoder-like structure could be useful. The architecture was added in front of the U-Net structure, and the loss from the segmentation was propagated backward through the autoencoder as well as the U-Net. Some experiments were run by comparing the results on the validation set to find an appropriate autoencoder structure. Trying both linear and convolutional autoencoders showed that the convolutional autoencoder was more effective for the task. Also, attempts to pre-train the encoder part of the autoencoder indicated that random initialization was more useful.

The number of layers in the encoder and decoder part of the network along with the number of channels used in the representation layer (latent space) and the kernel size were determined through hyperparameter tuning. The values that were reasonable to search through were guided by the findings in [38]. The layers were built so that in the encoder, all of the layers except for the input layer doubled the number of channels in the data. The input layer changed the number of channels in the data from its original size to the size where it can be doubled once for every layer and end up with the pre-determined size of the data representation. Similarly, all the layers in the decoder except for the output layer halved the number of channels in the data. The output layer decreased the number of channels to 3, to ensure that it was the correct size for the U-Net. Padding was used in all the layers to ensure that the images remained of size 64x64 pixels.

3.3 Research process

The research process consisted of three steps. First, data was collected and annotated. Secondly, the U-Net architecture was built and pre-trained. Thirdly, the experiments were run comparing the different architectures and types of input data. The steps for performing the experiments were iterative for each model and were as follows:

1. Perform hyperparameter tuning on the model with the appropriate data set.
2. Train five ensemble (combined) models using the best hyperparameters found.
3. Evaluate the model for each run and calculate a confidence interval for each metric.
4. Perform statistical analysis on the data in order to compare the different models to each other.

The details of these steps are explained in this section.

3.3.1 Hyperparameter tuning

Hyperparameter tuning refers to the process of finding the best hyperparameters for a given model with a given data set. The hyperparameters that need tuning can vary between different models but typically include some of the following: learning rate, regularization coefficients, which loss function to use, number of layers in a model, and the size of those layers. These are all parameters that affect how well a model learns, but they need to be set before training starts.

For the experiments in this project, hyperparameter tuning was done using a random search. What this means is that for each hyperparameter, a few possible values are defined. For each run during the tuning, one of those values is randomly selected for each hyperparameter. A set number of runs are completed and a new set of parameters are selected for each run. Using random search does not guarantee that the optimal parameters are found since not all combinations are tried. However, it is much more time-efficient than trying all possible combinations and typically leads to a configuration with results very close to the optimal.

The random search was set up so that each run continued for 50 epochs. The random search conducted 20 runs, with different hyperparameter settings each time, and the results were analyzed to find which run gave the best result on the validation set. For this run, the epoch with the highest IoU and Dice score were found, and the runs evaluated on the test data were then stopped at that epoch. This was to stop over-fitting from occurring. The evaluation was done on the validation data set, to ensure that no knowledge of the test set was present during training. This ensured that when testing occurred, it was on a completely independent data set.

3.3.2 Training ensemble models

Once the best hyperparameters were found and an appropriate number of epochs was determined, the models were trained. All the models were trained with the Dice loss function, as it continuously showed to lead to the best results. While training, a random seed was set. This means that all random functions give the same output when the same seed is used, which is good for the replicability of the results. Because it was found that the output was highly dependent on the random seed, an ensemble method was used to train the models. This means that for one ensemble model, five separate instances of the models were trained using different seeds, and their outputs were combined. It was done by going over the outputs one pixel at a time and taking the majority value. The ensemble method ensures that the results are less vulnerable to random seeds. In total, five ensemble methods were trained and during the training of the models, the seeds were set to the values between 10 and 34.

3.3.3 Evaluation metrics

The performance of the models was measured using both Dice score and IoU. This was because these are the two most commonly used evaluation metrics for image segmentation, which makes it convenient to compare the results to other research. At the end of a training run, both scores were calculated for each image and then an average score was calculated for the entire test data set. When all five runs had been completed, a mean IoU and Dice score were calculated along with a confidence interval for each. Running the model several times and calculating confidence intervals ensures that random values, such as the weight initialization, do not skew the results in favor of one model. The confidence interval states, with a 95% certainty, between what values the mean performance of the model lies. The formula for the confidence interval is defined as:

$$CI_{95} = \bar{x} \pm t_{95} \cdot \frac{\sigma}{\sqrt{N}} \tag{3.2}$$

where $\bar{x}$ is the mean, σ is the standard deviation, N is the number of samples and t_{95} is the t-value for the 95th confidence interval. The t-value is found using a lookup table from the confidence level and the (sample size - 1). The t-value is used because the sample size is smaller than 30.

3.3.4 Statistical analysis

A key component to drawing any conclusions during research is statistical analysis. This process tells whether differences in result values are due to random factors or because there is an actual increase in effectiveness in one of the models.

When performing statistical tests, a null hypothesis is formulated. In this case, the null hypothesis would be that the results from the different models come from the same population. To show that there is a statistical difference between the performance of the model, the null hypothesis needs to be rejected. If there is at least a 95% certainty that the null hypothesis is not true, it is rejected. In order to test the null hypothesis, a one-way ANOVA test [44] is carried out, followed by pairwise Tukey tests [45]. The ANOVA test compares all of the models to each other in order to determine if at least one of them differs from the others. The pairwise Tukey tests are then used to compare all models to each other in order to find what models are statistically different from each other.

An F-value is calculated with the ANOVA test as follows:

$$F = \frac{\frac{\sum_{i=1}^{k} \bar{Y}_i - \bar{Y}}{k-1}}{\frac{\sum_{i=1}^{k}\sum_{j=1}^{n_i} Y_{ij}^2 - \sum_{i=1}^{k} \bar{Y}_i}{n-k}} \tag{3.3}$$

where n is the total number of measurements from all the models, n_i is the number of measurements from model i, k is the total number of models, Y_{ij} is the j^{th} measurement of the i^{th} model, $\bar{Y}_i$ is the mean of the measurements from model i and $\bar{Y}$ is the mean of all the measurements. From a lookup table, the significance level of the F-value can be found.

To perform Tukey's test, a T-value is calculated from all of the data as follows:

$$T = q \cdot \sqrt{\frac{MSE}{n}} \tag{3.4}$$

where q is found in a lookup table, and MSE is the mean squared error found by the ANOVA test. The T-value is then compared to the absolute difference between the mean performances of each pair of models. If the absolute difference between the means is larger than the T-value, then those two models have significantly different performances with a 95% certainty.

The ANOVA test requires that the data is independent, normally distributed, and have similar variances. The data being independent means that the data sets do not have an impact on each other. That is, in this case,

the performance of one model is not dependent on the performance of another. Since the models are trained separately, we know that this is the case. In order to ensure that the data is normally distributed, the Shapiro-Wilks test [46] is performed on the results from each model. The Shapiro-Wilks test assumes the null hypothesis that the data is normally distributed. As long as the p-value is larger than 0.05, the null hypothesis is assumed to be true. The calculation of the test statistic, W, with the Shapiro-Wilks test, is given by:

$$W = \frac{(\sum_{i=1}^{n} a_i x_{(i)})^2}{\sum_{i=1}^{n}(x_i - \bar{x})^2} \tag{3.5}$$

where $x_{(i)}$ is the i^{th} lowest value out of the measurements, x_i is the i^{th} measurement, $\bar{x}$ is the average performance of the model, and a_i is a weight that can be found in a lookup table. From W, the p-value can be found using a lookup table. To ensure that the variances are similar, Levene's test [47] is performed. It works well even if the distribution of the data is not perfectly normal. Levene's test assumes the null hypothesis that the variances are equal, meaning that a p-value larger than 0.05 makes us accept the null hypothesis. Levene's test takes in the results from all the models at once and the mathematical formula is defined as:

$$W = \frac{n-k}{k-1} \cdot \frac{\sum_{i=1}^{k} n_i(\bar{Z}_i - \bar{Z})^2}{\sum_{i=1}^{k}\sum_{j=1}^{n_i}(Z_{ij} - \bar{Z}_i)^2} \tag{3.6}$$

where $Z_{ij} = |Y_{ij} - \bar{Y}_i|$, $\bar{Z}_i$ is the mean Z for model i and $\bar{Z}$ is the mean Z for all of the models. Again, W can be used to find the p-value from a lookup table. All of the statistical tests performed during this project are based on an average value of the two evaluation metrics used.

3.4 Implementation

The code for the project was written in Python 3.9.7 using PyTorch. The Torch library was used for the convolutional, max-pooling, and sigmoid layers, ReLU activation function, and batch normalization. It was also used for the optimizer and gradient scaling as well as calculating Binary Cross-Entropy loss and IoU. Data processing and augmentation were done using the Albumentations library. This includes normalizing the images, transforming them to tensors, and performing rotations and flips. "Weights and Biases" (wandb) was used for logging the results and selecting the sets of

hyperparameters during each run of the hyperparameter tuning. Reading the satellite data was done through the Rasterio library.

With the help of these libraries, code was mainly written by the author of this thesis. This includes for instance the architectures, training and evaluation methods, Dice loss function, and evaluation metric, along with functions to visualize the data. The method for splitting the data into tiles was largely based on [48] and the data processing methods were inspired by [49].

The models were trained on Alvis, an online server for data science researchers. All of the experiments were run using a container through the platform Apptainer. The experiments were run on a T4 server for a total approximate run time of 210 hours.

Chapter 4

Results and analysis

This chapter describes the results obtained through the experiments performed in the project. Section 4.1 examines the data collected to answer the first research question, while section 4.2 presents the data relating to research question 2, see section 1.1. For each section, the hyperparameters used and the results found are presented. Possible reasons for the results are also discussed and a statistical analysis is performed in each section.

4.1 Experiments with different models

This section presents and analyses the results of the experiments devised to answer the first research question. The idea is to determine what pre-processing of the data before entering it into the segmentation network gives the best results with a pre-trained model. The pre-trained models are also compared to a baseline where no pre-training or processing is performed. All of the models in this section were trained using only the VH-polarization as the data.

4.1.1 Hyperparameters

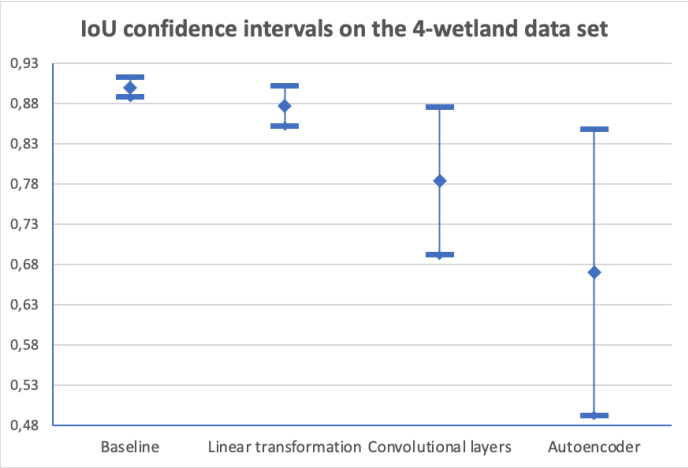
The hyperparameter settings for all of the models trained in this experiment were determined through hyperparameter tuning, as described in section 3.3.1. The hyperparameters selected for each architecture on the full data set can be seen in Figure 4.1. Similarly, the hyperparameter settings on the data set containing only data from Svartådalen can be found in Table 4.2. Not all of the hyperparameters are relevant to every model, and in those cases, the value is marked with N/A (not applicable).

	Model			
Hyperparameter	Baseline	Linear Transformation	Convolutional layers	Autoencoder
Learning rate	0.00001	0.00005	0.00001	0.00001
Weight decay	0.1	0.001	0.05	0.001
Rotation probability	0.3	0.6	0.9	0.9
Vertical flip probability	0.9	0.6	0.0	0.0
Horizontal flip probability	0.6	0.6	0.6	0.0
Kernel size	N/A	N/A	3	5
Number of layers in encoder (and decoder)	N/A	N/A	N/A	2
Number of channels in code layer	N/A	N/A	N/A	64

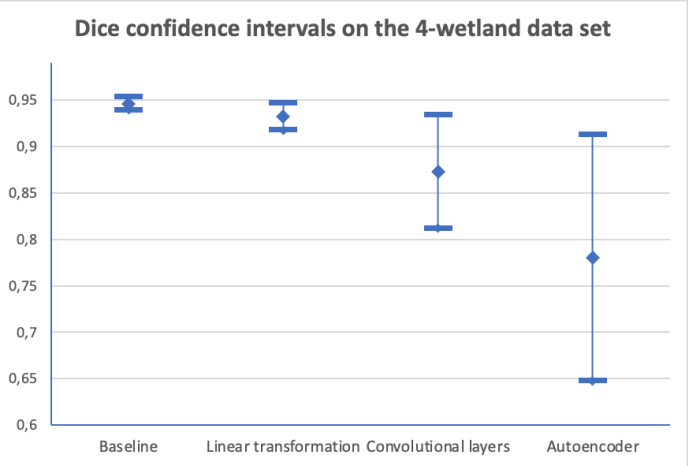
Table 4.1: The hyperparameters used for each of the models comparing architectures trained on the full data set

	Model			
Hyperparameter	Baseline	Linear Transformation	Convolutional layers	Autoencoder
Learning rate	0.0001	0.00005	0.0005	0.00001
Weight decay	0.001	0.001	0.001	0.005
Rotation probability	0.6	0.0	0.0	0.0
Vertical flip probability	0.3	0.9	0.9	0.3
Horizontal flip probability	0.0	0.3	0.0	0.6
Kernel size	N/A	N/A	3	5
Number of layers in encoder (and decoder)	N/A	N/A	N/A	2
Number of channels in code layer	N/A	N/A	N/A	32

Table 4.2: The hyperparameters used for each of the models comparing architectures trained on the Svartådalen data set



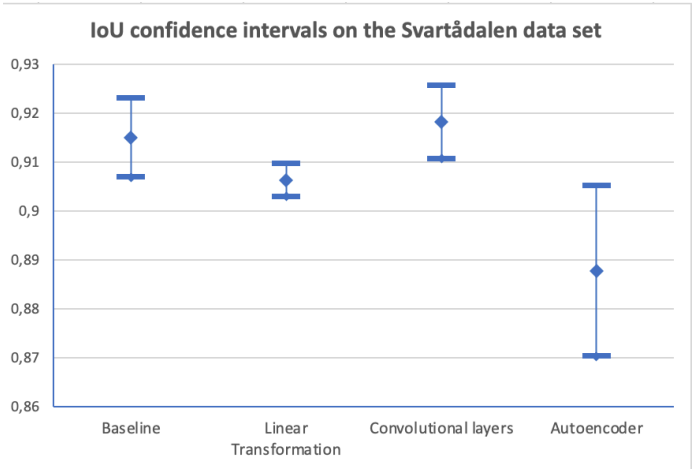
(a) IoU confidence intervals on the full data set



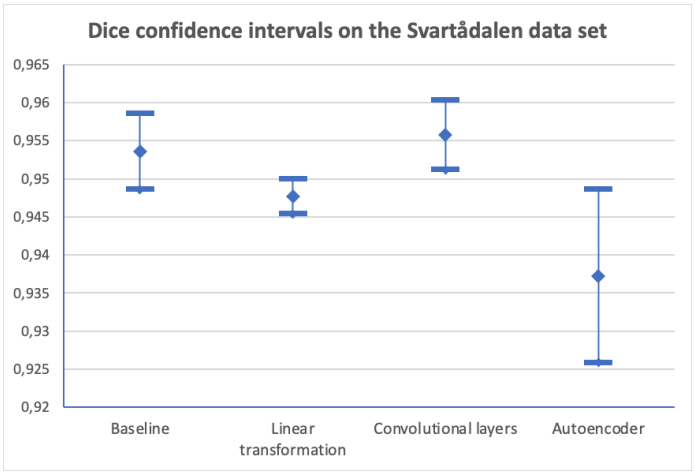
(b) Dice confidence intervals on the full data set

4.1.2 Performance analysis

The baseline and each of the three pre-training architectures are compared to each other in this section. Each of the models is trained and evaluated on the full data set, and then the Svartådalen data set. The means and confidence intervals for the calculated IoU and Dice scores on the test set of each of the data sets can be seen in Figure 4.1. The confidence intervals were calculated using equation 3.2. The diamond in the plots represents the average value over all five runs, and the line represents the confidence interval. For each of the models, the tiles from one month are reconstructed into the full wetland area. In Figure 4.2, the annotation and the output from the highest average performing model trained during each of the experiments can be seen.

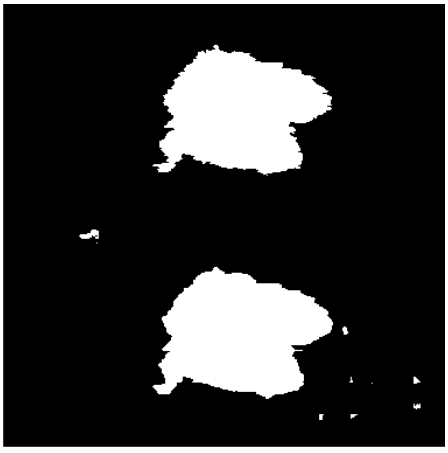


(c) IoU confidence intervals on the Svartådalen data set



(d) Dice confidence intervals on the Svartådalen data set

Figure 4.1: Visualizations of the confidence intervals and means for IoU and Dice on the test set for each of the models comparing architectures



(a) Annotation and output from August 2020 from baseline model trained on the full data set

(b) Annotation and output from August 2022 from convolutional layers model trained on the Svartådalen data set

Figure 4.2: The output of the best-performing model on each of the data sets. Top=annotation, bottom=output from model

From this data, a large difference can be seen in the performance of the models depending on the data set. Overall, the data set with only Svartådalen leads to higher average performance for all of the models, and often smaller confidence intervals. While initially, this might seem counterintuitive since a smaller data set is used, there is a feasible explanation. In the data set with 4 wetlands, none of the data seen by the model during training comes from the same wetland as is used in the test set. As a result, the shapes unique to the Svartådalen body of water are new to the model. On the other hand, the data set with only Svartådalen is familiar with those shapes from training. While no annotation is identical, some physical characteristics do not change much over the years that have been annotated. As a result, the performance metrics of the models trained and evaluated on Svartådalen should only be used to compare those models to each other. They are not an accurate representation of how well the models would perform on wetlands from other regions.

Statistical significance tests were performed on the results from the experiments on both data sets, in order to compare the different models to each other within each data set. The analysis was performed using the equations described in section 3.3.4. Both one-way ANOVA tests showed that there was some statistical significance between the different models' performances ($F(3, 16) = 4.08$, $p = 0.025$ on the full data set and $F(3, 16) = 6.31$, $p = 0.00499$)

Pair-wise comparison	Data set	
	4-wetland data set	Svartådalen data set
Baseline - Linear transformation	p = 0.991	p = 0.647
Baseline - Convolutional layers	p = 0.458	p = 0.972
Baseline - Autoencoder	**p = 0.0290**	**p = 0.0133**
Linear transformation - Convolutional layers	p = 0.629	p = 0.400
Linear transformation - Autoencoder	p = 0.0514	p = 0.128
Convolutional layers - Autoencoder	p = 0.387	**p = 0.00555**

Table 4.3: P-values calculated from the pair-wise Tukeys tests on all of the model pairs for both the full data set and Svartådalen data set.

Model	Data set	
	4-wetland data set	Svartådalen data set
Baseline	p = 0.949	p = 0.537
Linear transformation	p = 0.447	p = 0.437
Convolutional layers	p = 0.436	p = 0.839
Autoencoder	p = 0.128	p = 0.638

Table 4.4: P-values calculated from the Shapiro-Wilks test on the results from all of the models on both the full data set and the Svartådalen data set.

on the data set with only data from Svartådalen). Each test has 3 degrees of freedom. The results from the planned pair-wise Tukey tests can be seen in Table 4.3. To ensure that the results from the ANOVA and Tukey tests are reliable, tests for normality and similarity of variance were performed. The results from the Shapiro-Wilks test for normality of data on both data sets can be seen in Table 4.4. From the results, it can be assumed that the results are normally distributed in both sets of experiments. To check the similarity of variances, Levene's test was performed. On the results from the full data set, it showed that the variances were not similar ($F(3, 16) = 3.31$, $p = 0.0471$). This means that the results from the statistical analysis tests can not be completely trusted on the full data set, as all of the criteria for the tests are not fulfilled. However, on the Svartådalen data set results, the similarity of variances can be assumed ($F(3, 16) = 2.32$, $p = 0.114$) and the statistical tests performed on those results are therefore valid.

The results from the experiments on both data sets demonstrate that the autoencoder leads to both the lowest average and most uneven performance.

This conclusion is also somewhat supported by the statistical analysis, as the autoencoder was shown to perform statistically worse than the model with the highest mean performance in both sets of experiments and the two best-performing models in the Svartådalen data set tests. On the other hand, no statistical significance can be shown as to which model performed the best. When using the full data set, the baseline model has the highest average performance and the smallest confidence intervals. This indicates that the pre-training is not useful at all for this particular model and data set. This could be due to the fact that the types of images in the source data set are not similar enough to the target data set. Out of the pre-training models, the linear transformation model has the highest average and most consistent performance. It was followed by the model incorporating convolutional layers and lastly the autoencoder model. The poor performance of the autoencoder model could be a result of it containing the most amounts of layers out of the tested models. It therefore has more weights to tune, which normally requires a larger data set to achieve the same performance as a model with fewer weights.

On the Svartådalen data set, the model with the highest average is the one with convolutional layers, closely followed by the baseline. The model incorporating linear transformation has a slightly worse average performance but it is more dependable. It is however worth noting that all three of these models have a rather small confidence interval. The autoencoder model once again gives the lowest average performance. On this data set, the difference between the models is also smaller, with the difference in average performance between best and worst models being about 2.5 percent units, as opposed to approximately 20 percent units for the larger data set.

4.2 Experiments with SAR and MSI data

To answer the second research question, experiments were run with a few different combinations of polarizations from the SAR data and bands from the MSI data. From the experiments with the full data set, the best pre-trained model was found to be the linear transformation model. Since this method only takes in one SAR polarization, the conclusion is that only using VH is the best option for that data set. In the experiments with the smaller data set, however, the model with the highest average performance was the one using convolutional layers to transform the data from SAR to RGB. Therefore, the experiments with changing the number of input channels were performed using the convolutional layer model and the Svartådalen data set. This data set is used to work with data of higher quality and to perform experiments to answer

	Convolutional layer 1		Convolutional layer 2	
Model	Input channels	Output channels	Input channels	Output channels
VH	1	2	2	3
VH + VV	2	2	2	3
VH + VV + RGB	5	4	4	3
All channels	12	7	7	3

Table 4.5: The number of input and output channels of the two convolutional layers in each of the experiments performed

the second research question. The results should only be used to compare models trained on the same data set, not to predict how well these models perform on other data. Because not all months had matching SAR and MSI data, the data set for these experiments was slightly smaller. Due to this, the model trained on only VH was run again during these experiments with the smaller data set in order to make a fair comparison to the other inputs. The other combinations of data used as input were:

1. Both of the polarizations from the SAR data - VH and VV.
2. Both the VH and VV polarizations along with the R, G, and B bands from MSI.
3. All of the available channels which include both polarizations and bands 1 through 9 from Sentinel-2 (see Table 2.1).

In order to keep the experiments fair, all of the models consisted of two convolutional layers, each followed by a ReLU activation function, before the U-Net. Since the number of input channels was changing for these experiments, each convolutional layer's input and output channels had to be adopted to fit the data. In Table 4.5, the variables set for those layers can be seen. The idea was to, as closely as possible, halve the distance between the number of input channels and the three channels needed for an RGB image in the first layer and then output three channels after the second one.

4.2.1 Hyperparameters

In Table 4.6 the hyperparameters setting used during training for each of the models can be found. The settings were found through hyperparameter tuning, as described in section 3.3.1. For the model trained on only VH data, the hyperparameter settings from the model trained on all the Svartådalen data

	Model			
Hyperparameter	VH	VH + VV	VH + VV + RGB	All channels
Learning rate	0.0005	0.00001	0.00005	0.00005
Weight decay	0.001	0.05	0.01	0.05
Rotation probability	0.0	0.9	0.6	0.9
Vertical flip probability	0.9	0.3	0.0	0.3
Horizontal flip probability	0.0	0.3	0.6	0.9
Kernel size	3	7	7	5

Table 4.6: The hyperparameters used for each of the models comparing input data

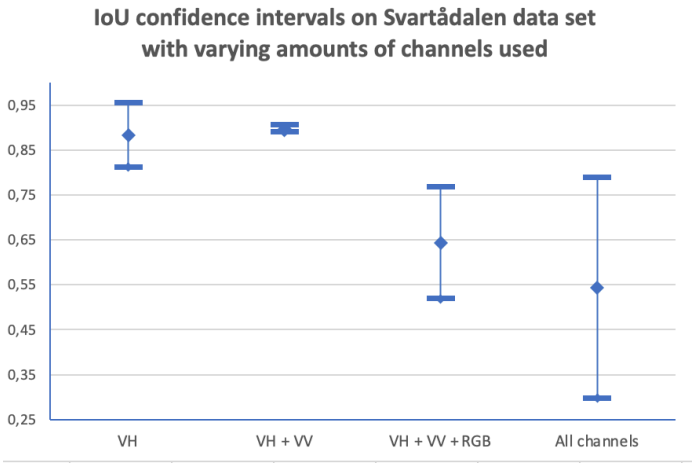
were reused as the data set was essentially the same, with only a few months of the data removed.

4.2.2 Performance analysis

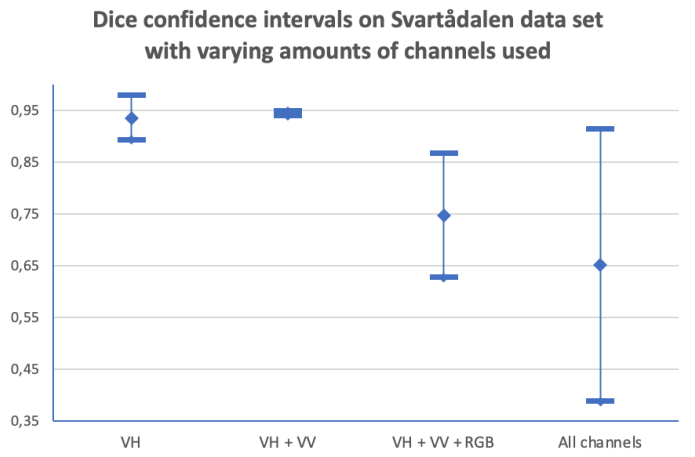
The means and confidence intervals at a 95% confidence level for both IoU and Dice on the Svartådalen data set with different input channels can be seen in Figure 4.3 where the mean value for each confidence interval is marked by the diamond. The output from the highest average performing model trained on only VH and both VH and VV is shown in Figure 4.4. The tiles from one month are reconstructed into the full wetland area and shown along with the annotation.

A statistical analysis was also performed on these results. The ANOVA test showed that there was some statistical significance between the performance of the models ($F(3, 26) = 4.80$, $p = 0.0144$). The results of the pair-wise Tukey tests can be seen in Table 4.7. To verify the validity of the statistical tests, the data was tested for normality and similarity of variances. Levene's test showed that we can assume similarity in variance in the data ($F(3, 16) = 1.18$, $p = 0.349$). However, the results from the Shapiro-Wilks test can be seen in Table 4.8 and since normality can not be assumed in the samples from all of the models, the statistical tests are not valid.

The data indicates that the model has a lower average performance, both in terms of stability and mean value once the MSI data is included in training. The model with all of the channels included exhibited the lowest average performance of them all. It was also shown to perform statistically significantly worse than the two models that only included SAR data. One reason for this might be that the rather small data set favors less complex data.



(a) IoU confidence intervals



(b) Dice confidence interval

Figure 4.3: Visualizations of the confidence intervals and means for IoU and Dice on the test set for each of the models comparing input data

Model	P-value
VH - (VH + VV)	p = 0.999
VH - (VH + VV + RGB)	p = 0.204
VH - All channels	**p = 0.0371**
(VH + VV) - (VH +VV + RGB)	p = 0.172
(VH + VV) - All channels	**p = 0.0300**
(VH + VV + RGB) - All channels	p = 0.782

Table 4.7: P-values calculated from the pair-wise Tukey tests on all of the model pairs used to compare input data.

(a) Annotation and output from August 2022 from convolutional layers model trained with only VH as input

(b) Annotation and output from August 2022 from convolutional layers model trained with both VH and VV as input

Figure 4.4: The output of the best-performing model trained on VH and VH + VV. Top=annotation, bottom=output from model

Model	P-value
VH	p = 0.0261
VH + VV	p = 0.872
VH + VV + RGB	p = 0.0771
All channels	p = 0.0373

Table 4.8: P-values calculated from the Shapiro-Wilks test on the results from all of the models comparing input data.

Adding in more channels means that more weights are present in the model which need to be tuned. In this case, it also makes the problem more complex, since two different types of data need to be combined into an RGB image that makes sense to the U-Net. In the model with all channels as input data, the data also comes in different resolutions, which might impact the results. However, it could also be an indication that combining SAR and MSI data is not best done through the convolutional layers method. Running these tests with a larger data set could help tell which of these is the case.

The data shows that including both polarizations leads to a slightly higher average performance and smaller confidence interval than only inputting VH to the model, although no statistical significance could be shown in the difference between their performances. It seems that since both of these bands are very similar, they are combined nicely to provide a more detailed input image for the U-Net. However, the models with highest evaluation scores that input only VH got higher results than the models with highest evaluation scores that used both VH and VV as input data.

Chapter 5

Discussion

This chapter discusses what aspects of the work can call into question the validity of the method used. It also brings up the limitations of the work and what changes would have been made if the project could have been redone. Following that, an inspection of how the project impacts sustainability is given along with societal and ethical aspects to be considered in relation to the project.

5.1 Issues with validity of method

Analyzing the methodology of this work, the main aspect that calls into question the validity of the work is the data set. Because the annotations are made by hand based on the radar images and not collected on-site, this creates a limit as to how much the model can learn. Ideally, a machine learning model's knowledge should surpass that of a human's, but with human-labeled data, it becomes more difficult. Especially since in this case, the human brain is not trained for the task. Humans are very good at discerning natural objects, which means they are adept at annotating natural images. When dealing with radar images, however, the objects and shapes are more abstract, something that the human brain is not as good at discerning. Hand-annotated data was chosen because no applicable data set was available for the task where the ground truth was measured at the actual wetlands and creating one was not feasible within the scope of this project. Also, the idea behind the project is to create a streamlined approach to delineating wetlands, which works better when the data is faster to collect. However, the neural networks would likely have been more powerful and had the ability to surpass humans' ability to delineate wetlands based on radar imagery had the ground truth come from

on-site measurements.

Another issue with the method is that the models are implemented by hand, rather than using already existing and validated code. As a result, it is difficult to validate if the models are implemented correctly. While they all produce the correct output, there might be small errors that are difficult to identify which cause the models to perform worse than they otherwise would have. This means that the results in some cases might reflect how well constructed the models are rather than how well they ideally would have performed.

5.2 Limitations and possible improvements

The most prominent limitation of the project is the lack of high-quality data, where the labels confidently can be said to closely correlate to reality. Labeling the data is not entirely objective, as the SAR images consist of shades of grey. It is very difficult to know where to draw the line as to what shades count as water and which ones do not. Having annotations from data collected on-site would have been ideal, but was not feasible. Because labeling is also time-consuming, there was not enough time to go back and collect more data once it was discovered that the quality was not ideal. The data available for the project makes the results occasionally difficult to interpret. This is mainly true for the experiments run on the full data set, but it also affects the Svartådalen data set. Since this data set only contains one wetland, it is hard to know if the good results obtained are due to the method working well or because the test data is too similar to the training and validation data.

Using larger samples for the statistical significance testing would mean that the tests can be more certain of the mean that those samples originate from. As a result, more models might have been found to have statistically significant differences in performance if more data had been collected. Collecting a significantly larger sample for each model would probably be a good choice for further investigations.

Building and pre-training the U-Net took a large amount of time. Due to this, fine-tuning the network and getting the best possible segmentation result on the COCO data set was not given as much time as originally planned. Because of this, the segmentation performance could probably have been improved on COCO, which also could have improved the results on the target data set. While this should not impact the relative performance of the pre-trained models to each other, it might have an impact on how they compare to the baseline.

5.3 Sustainability

The main goal of this project is to aid the battle against climate change by providing a tool to help restoration efforts of wetlands. Since wetlands have a major impact on the climate through their strong ability to absorb carbon dioxide, it is sustainably important to make sure that wetlands are thriving and take up a vast amount of space. Maintaining wetlands also helps flood prevention, makes sure the groundwater is clean and protects ecosystems. Overall, making sure that the nature around us is healthy is very important for a sustainable society, as we rely on nature for everything needed to sustain our lifestyle.

While restoring wetlands has a positive effect on the climate, machine learning in and of itself can have quite negative effects. Training machine learning models require a large amount of energy to power the servers. While most of the energy in Sweden comes from non-fossil fuel sources, they are still often disruptive to the ecosystems in which they are placed.

5.4 Societal aspects

There is much debate these days as to the ethical aspects of developing machine learning technologies. Following the release of ChatGPT, artificial intelligence has taken a huge leap, and many researchers warn against continuing the research until the effects can be assessed and appropriate laws formulated. Because machine learning technologies can be used to invade the privacy of citizens and replace their jobs, they can have a huge impact on our society. While the technology worked on here does not use any private data, nor is it meant to replace human labor but rather aid in it, there is no way to guarantee that the knowledge gained can not be used for unethical purposes. On the other hand, climate change does not only impact our environment but also poses risks to our food supplies, financial systems, and infrastructure. If the work in this project can in any way help prevent those things, it will also have some very positive effects on our society.

5.5 Ethical aspects

The models trained in this project do not use any data related to humans in any form. As a result, any potential biases in the model can not be used to perpetuate societal biases towards any group of people. However, there is

very little transparency as to how the model makes its decisions. In order to apply a model to a real-world problem, it is important to be able to verify the decision-making process.

Chapter 6

Conclusions and Future Work

This chapter discusses what conclusions can be drawn from the data presented in the previous chapter. The answers found to the two research questions are described, as well as any other conclusions drawn during the project. Finally, possible directions for future work are listed.

6.1 Conclusions

This section discusses the conclusions that can or can not, be drawn from the data collected. Neither of the research questions can be fully answered, but some insights have been gained that can help the next researcher continue the work in the right direction. Some other observations have also been made which are explained at the end of this section.

6.1.1 Research question 1

The first research question addressed by this project is: *Out of the methods for converting satellite data into RGB images: mathematical linear transformation, stacked convolutional layers, and autoencoder, which leads to the highest accuracy of vegetated water delineation on a pre-trained network?*

Based on the experiments, no conclusions can be drawn as the statistical analysis does not indicate that one method led to the highest accuracy. The highest mean performance varied between methods depending on the data set used. The following tendencies can be seen from the mean performances measured:

- Working with a larger, but lower-quality data set where the test set consists of a wetland not previously seen by the model seems to require

the linear transformation method.

- However, a smaller data set with higher quality seems to respond well to the convolutional layer method of converting the data.

That being said, in neither of the data sets any statistical significance could be shown between the performance of those two methods for converting the data. As a result, they could both be interesting choices for someone attempting a similar project, and the best method would be found by experimenting with the data set used. Currently, the linear transformation method is in use by some researchers [33, 34], while no evidence has been found of anyone applying the convolutional layers method.

What can be seen from the results is that using an autoencoder for converting the SAR data into RGB does not work particularly well. Of course, there is a huge number of variants of autoencoders, and only a few were considered for this project. Therefore there may be an autoencoder that would perform the task well. However, a standard linear or convolutional autoencoder is not an appropriate choice.

6.1.2 Research question 2

Secondly, the project considered the research question: *Which combination of SAR and MSI data used for fine-tuning a pre-trained network out of: only VH polarization data, both VH and VV polarization data, both VH and VV polarization data along with the R, G, and B bands from Sentinel-2, both VH and VV polarization data along with bands 1 through 9 from Sentinel-2, leads to the highest accuracy of vegetated water delineation?*

No statistical evidence was found that can be used to draw conclusions that answer this research question. For one of the data sets, the pre-trained model with the highest average performance was the linear transformation method, which only takes in one polarization of SAR data. As a result, the best input data was concluded to only include VH. The smaller data set however was run with different input data on the convolutional layers model. Here, the average performance tended toward indicating that using only VH or both VH and VV polarizations would be the most appropriate. Using only VH tended toward showing that some models have higher performance while using both polarizations seemed to lead to a higher average performance and more similar results each run. However, no statistical significance could be seen between the two types of input data.

Overall, it seems as though including MSI data in the input data has a negative result on the performance of the model for the data set used. It could be shown with statistical significance that using all available channels led to a worse result. The average performance of the models tended towards indicating that the input data that includes RGB also produces worse results. This could be because more complex input data requires a larger data set for the model to learn it well, or it could mean that the two types of data are not well suited to be combined through the convolutional layers model. It could also be an indication that the particular task worked on in this project is not aided by including MSI data.

6.1.3 Other conclusions

In the experiments that compare different architectures, the models were also compared to a baseline that did not incorporate pre-training. In the experiments involving the full data set, the baseline had the highest mean performance and the second highest when the smaller data set was used. No statistical difference could be found between the performance of the baseline and either the linear transformation or convolutional layers model in either experiment. This shows that even on very small data sets, the time and resources required for pre-training might not be necessary. Using another data set for pre-training might lead to different results, however. Viable options could be the ImageNet which is much larger than COCO but is only used for classification, or the Sen1Floods11 data set which is a labeled water segmentation data set with SAR images consisting of just under 5000 images.

Another conclusion that can be drawn is that the quality of the data is important for the outcome of the training. While this is already known by researchers, the results from these experiments support that conclusion. Not only does the higher quality data lead to every model performing better, but it also shows that the random seed becomes much less important. This indicates that the higher quality data allows the model to find good weights, no matter the random initialization. On the other hand, lower-quality data is dependent on the right initialization to be able to deliver good results, giving the impression that the model has to guess more.

6.2 Future work

Some possible directions for continuing this research have already been touched upon in the thesis, but they are collected here for convenience.

- Collecting a data set with more than one high-quality labeled wetland in order to verify some of the results found here.
- Training all of the models more times to see if more statistical significance can be shown.
- Experiment with different methods for combining SAR and MSI data, as the methods used in this work have shown not to be ideal for this.
- Try different architectures for the segmentation part of the model. The U-Net has worked well, but perhaps other architectures would lead to even better results. Examples to try are the SegNet and DeepLab models.
- Experiment with different source data sets to see if this improves the performance of the pre-trained networks. Here good places to start would be the ImageNet and Sen1Floods11 data sets.

References

[1] M. Sorek-Hamer, M. Von Pohle, A. Sahasrabhojanee, A. Akbari Asanjan, E. Deardorff, E. Suel, V. Lingenfelter, K. Das, N. C. Oza, M. Ezzati, and M. Brauer, "A deep learning approach for meter-scale air quality estimation in urban environments using very high-spatial-resolution satellite imagery," *Atmosphere*, vol. 13, no. 5, p. 696, Apr 2022. doi: 10.3390/atmos13050696. [Online]. Available: http://dx.doi.org/10.3390/atmos13050696 [Page 1.]

[2] Y. Ban, P. Zhang, A. Nascetti, A. Bevington, and M. Wulder, "Near real-time wildfire progression monitoring with sentinel-1 sar time series and deep learning," *Scientific Reports*, vol. 10, pp. 1–15, Jan 2020. doi: 10.1038/s41598-019-56967-x [Page 1.]

[3] I. Duporge, O. Isupova, S. Reece, D. Macdonald, and T. Wang, "Using very-high-resolution satellite imagery and deep learning to detect and count african elephants in heterogeneous landscapes," *Remote Sensing in Ecology and Conservation*, vol. 7, pp. 369–381, 12 2020. doi: 10.1002/rse2.195 [Page 1.]

[4] Department of Sustainability Environment Water Population & Communities and Wetlands and Waterbirds Taskforce, *The role of wetlands in the carbon cycle*, Australian Government, July 2012. [Page 1.]

[5] United States Environmental Protection Agency, "Why are wetlands important?" March 2023. [Online]. Available: https://www.epa.gov/wetlands/why-are-wetlands-important [Page 1.]

[6] United Nations Framework Convention on Climate Change, "Wetlands disappearing three times faster than forests," October 2018. [Online]. Available: https://unfccc.int/news/wetlands-disappearing-three-times-faster-than-forests [Page 1.]

[7] United States Environmental Protection Agency, “Principles of wetland restoration,” May 2022. [Online]. Available: https://www.epa.gov/wetlands/principles-wetland-restoration#restorenaturalstructer [Page 1.]

[8] J. Evers, “Wetland,” May 2022. [Online]. Available: https://education.nationalgeographic.org/resource/wetland [Page 1.]

[9] S. K. McFeeters, “The use of the normalized difference water index (ndwi) in the delineation of open water features,” *International Journal of Remote Sensing*, vol. 17, no. 7, pp. 1425–1432, 1996. doi: 10.1080/01431169608948714. [Online]. Available: https://doi.org/10.1080/01431169608948714 [Page 1.]

[10] O. Ronneberger, P. Fischer, and T. Brox, “U-net: Convolutional networks for biomedical image segmentation,” in *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, N. Navab, J. Hornegger, W. M. Wells, and A. F. Frangi, Eds. Cham: Springer International Publishing, 2015. ISBN 978-3-319-24574-4 pp. 234–241. [Pages ix, 6, 7, 11, and 25.]

[11] A. Garcia-Garcia, S. Orts-Escolano, S. Oprea, V. Villena-Martinez, P. Martinez-Gonzalez, and J. Garcia-Rodriguez, “A survey on deep learning techniques for image and video semantic segmentation,” *Applied Soft Computing*, vol. 70, pp. 41–65, 2018. doi: https://doi.org/10.1016/j.asoc.2018.05.018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1568494618302813 [Pages 6, 9, 10, and 11.]

[12] X. Yuan, J. Shi, and L. Gu, “A review of deep learning methods for semantic segmentation of remote sensing imagery,” *Expert Systems with Applications*, vol. 169, p. 114417, 2021. doi: https://doi.org/10.1016/j.eswa.2020.114417. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0957417420310836 [Page 6.]

[13] V. Badrinarayanan, A. Kendall, and R. Cipolla, “Segnet: A deep convolutional encoder-decoder architecture for image segmentation,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, pp. 2481–2495, 2015. [Online]. Available: https://api.semanticscholar.org/CorpusID:60814714 [Pages ix, 6, 7, and 8.]

[14] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *3rd International Conference on Learning Representations (ICLR 2015)*, pp. 1–14, 2015. [Page 8.]

[15] D. Müller, I. Soto-Rey, and F. Kramer, "Towards a guideline for evaluation metrics in medical image segmentation," *BMC Research Notes*, vol. 15, pp. 1–8, 12 2022. doi: 10.1186/s13104-022-06096-y [Pages 9 and 10.]

[16] M. Yi-de, L. Qing, and Q. Zhi-bai, "Automated image segmentation using improved pcnn model based on cross-entropy," in *Proceedings of 2004 International Symposium on Intelligent Multimedia, Video and Speech Processing, 2004.*, 2004. doi: 10.1109/ISIMP.2004.1434171 pp. 743–746. [Page 10.]

[17] S. Jadon, "A survey of loss functions for semantic segmentation," in *2020 IEEE Conference on Computational Intelligence in Bioinformatics and Computational Biology (CIBCB)*, 2020. doi: 10.1109/CIBCB48159.2020.9277638 pp. 1–7. [Pages 10 and 11.]

[18] C. H. Sudre, W. Li, T. Vercauteren, S. Ourselin, and M. J. Cardoso, "Generalised dice overlap as a deep learning loss function for highly unbalanced segmentations," in *Deep Learning in Medical Image Analysis and Multimodal Learning for Clinical Decision Support*. Springer International Publishing, 2017, pp. 240–248. [Online]. Available: https://doi.org/10.1007%2F978-3-319-67558-9_28 [Page 11.]

[19] Z. Hayder, X. He, and M. Salzmann, "Boundary-aware instance segmentation," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Los Alamitos, CA, USA: IEEE Computer Society, jul 2017. doi: 10.1109/CVPR.2017.70. ISSN 1063-6919 pp. 587–595. [Online]. Available: https://doi.ieeecomputersociety.org/10.1109/CVPR.2017.70 [Page 11.]

[20] J. Brown, "Sar 101: An introduction to synthetic aperture radar," February 2020. [Online]. Available: https://www.capellaspace.com/sar-101-an-introduction-to-synthetic-aperture-radar [Pages 12 and 13.]

[21] NASA Earth data, "What is synthetic aperture radar?" [Online]. Available: https://www.earthdata.nasa.gov/learn/backgrounders/what-is-sar [Page 12.]

[22] The European Space Agency, "Sentinel-1 sar user guide." [Online]. Available: https://sentinel.esa.int/web/sentinel/user-guides/sentinel-1-sar [Page 12.]

[23] O. Nicolis and C. Gonzalez, "19 - wavelet-based fractal and multifractal analysis for detecting mineral deposits using multispectral images taken by drones," in *Methods and Applications in Petroleum and Mineral Exploration and Engineering Geology*, S. Gaci, O. Hachay, and O. Nicolis, Eds. Elsevier, 2021, pp. 295–307. ISBN 978-0-323-85617-1. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B9780323856171000175 [Page 13.]

[24] The European Space Agency, "Sentinel-2 msi user guide." [Online]. Available: https://sentinel.esa.int/web/sentinel/user-guides/sentinel-2-msi [Page 13.]

[25] Sentinel-Hub by Sinergise, "Sentinel-2 bands." [Online]. Available: https://custom-scripts.sentinel-hub.com/custom-scripts/sentinel-2/bands/ [Pages xi and 14.]

[26] X. Han, Z. Zhang, N. Ding, Y. Gu, X. Liu, Y. Huo, J. Qiu, Y. Yao, A. Zhang, L. Zhang, W. Han, M. Huang, Q. Jin, Y. Lan, Y. Liu, Z. Liu, Z. Lu, X. Qiu, R. Song, J. Tang, J.-R. Wen, J. Yuan, W. X. Zhao, and J. Zhu, "Pre-trained models: Past, present and future," *AI Open*, vol. 2, pp. 225–250, 2021. doi: https://doi.org/10.1016/j.aiopen.2021.08.002. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2666651021000231 [Page 15.]

[27] S. C. Wong, A. Gatt, V. Stamatescu, and M. D. McDonnell, "Understanding data augmentation for classification: when to warp?" 2016. [Page 15.]

[28] C. Shorten and T. M. Khoshgoftaar, "A survey on image data augmentation for deep learning," *Journal of Big Data*, vol. 6, no. 1, pp. 1–48, 2019. doi: 10.1186/s40537-019-0197-0 [Page 15.]

[29] C. Tottrup, D. Druce, X. Tong, E. Barvels, M. Christensen, K. Grogan, S. Huber, and S. Crane, *The Global Wetland Extent: Towards a high-resolution global-level inventory of the spatial extent of vegetated wetlands*, UN Environment, Nairobi, Kenya, 2020. [Page 16.]

[30] Habitat Seven, "Measuring wetland area." [Online]. Available: https://files.habitatseven.com/unwater/Measuring-wetland-area.pdf [Page 16.]

[31] S. López-Tapia, P. Ruiz, M. Smith, J. Matthews, B. Zercher, L. Sydorenko, N. Varia, Y. Jin, M. Wang, J. B. Dunn, and A. K. Katsaggelos, "Machine learning with high-resolution aerial imagery and data fusion to improve and automate the detection of wetlands," *International Journal of Applied Earth Observation and Geoinformation*, vol. 105, p. 102581, 2021. doi: https://doi.org/10.1016/j.jag.2021.102581. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0303243421002889 [Page 16.]

[32] F. J. Peña, C. Hübinger, A. H. Payberah, and F. Jaramillo, "Deepaqua: Semantic segmentation of wetland water surfaces with sar imagery using deep neural networks without manually annotated data," *International Journal of Applied Earth Observation and Geoinformation*, vol. 126, p. 103624, 2024. doi: https://doi.org/10.1016/j.jag.2023.103624. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1569843 22300448X [Pages 16 and 20.]

[33] Y. Wang, C. Wang, and H. Zhang, "Ship classification in high-resolution sar images using deep learning of small datasets," *Sensors*, vol. 18, no. 9, p. 2929, Sep 2018. doi: 10.3390/s18092929. [Online]. Available: http://dx.doi.org/10.3390/s18092929 [Pages 17, 25, and 50.]

[34] L. Gao, C. Wang, K. Liu, S. Chen, G. Dong, and H. Su, "Extraction of floating raft aquaculture areas from sentinel-1 sar images by a dense residual u-net model with pre-trained resnet34 as the encoder," *Remote Sensing*, vol. 14, no. 13, p. 3003, Jun 2022. doi: 10.3390/rs14133003. [Online]. Available: http://dx.doi.org/10.3390/rs14133003 [Pages 17 and 50.]

[35] N. Lv, C. Chen, T. Qiu, and A. K. Sangaiah, "Deep learning and superpixel feature extraction based on contractive autoencoder for change detection in sar images," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 12, pp. 5530–5538, 2018. doi: 10.1109/TII.2018.2873492 [Page 17.]

[36] P. Singh and R. Shree, "Analysis and effects of speckle noise in sar images," in *2016 2nd International Conference on Advances in Computing, Communication, & Automation (ICACCA) (Fall)*, 2016. doi: 10.1109/ICACCAF.2016.7748978 pp. 1–5. [Page 17.]

[37] R. La Grassa, C. Re, G. Cremonese, and I. Gallo, "Hyperspectral data compression using fully convolutional autoencoder," *Remote Sensing*, vol. 14, no. 10, p. 2472, May 2022. doi: 10.3390/rs14102472. [Online]. Available: http://dx.doi.org/10.3390/rs14102472 [Page 17.]

[38] G. Dong, G. Liao, H. Liu, and G. Kuang, "A review of the autoencoder and its variants: A comparative perspective from target recognition in synthetic-aperture radar images," *IEEE Geoscience and Remote Sensing Magazine*, vol. 6, no. 3, pp. 44–68, 2018. doi: 10.1109/MGRS.2018.2853555 [Pages 17 and 27.]

[39] Q. Song, F. Xu, and Y.-Q. Jin, "Radar image colorization: Converting single-polarization to fully polarimetric using deep neural networks," *IEEE Access*, vol. 6, pp. 1647–1661, 2018. doi: 10.1109/ACCESS.2017.2779875 [Page 18.]

[40] A. Shakya, M. Biswas, and M. Pal, "Fusion and classification of sar and optical data using multi-image color components with differential gradients," *Remote Sensing*, vol. 15, no. 1, p. 274, Jan 2023. doi: 10.3390/rs15010274. [Online]. Available: http://dx.doi.org/10.3390/rs15010274 [Page 18.]

[41] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *Computer Vision – ECCV 2014*, D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, Eds. Cham: Springer International Publishing, 2014. ISBN 978-3-319-10602-1 pp. 740–755. [Page 19.]

[42] Ramsar, 2023. [Online]. Available: https://www.ramsar.org [Page 20.]

[43] Y. H. Liu, "Feature extraction and image recognition with convolutional neural networks," *Journal of Physics: Conference Series*, vol. 1087, no. 6, p. 62032, sep 2018. doi: 10.1088/1742-6596/1087/6/062032. [Online]. Available: https://dx.doi.org/10.1088/1742-6596/1087/6/062032 [Page 26.]

[44] R. Fisher, *Statistical methods for research workers*. Edinburgh Oliver & Boyd, 1925. [Page 30.]

[45] J. W. Tukey, "Comparing individual means in the analysis of variance," *Biometrics*, vol. 5, no. 2, pp. 99–114, 1949. [Online]. Available: http://www.jstor.org/stable/3001913 [Page 30.]

[46] S. S. Shapiro and M. B. Wilk, “An analysis of variance test for normality (complete samples),” *Biometrika*, vol. 52, no. 3/4, pp. 591–611, 1965. [Online]. Available: http://www.jstor.org/stable/2333709 [Page 31.]

[47] H. Levene, “Robust tests for equality of variances,” *Ingram Olkin; Harold Hotelling; et al. (eds.). Contributions to Probability and Statistics: Essays in Honor of Harold Hotelling.*, pp. 278–292, 1960. [Page 31.]

[48] [Online]. Available: https://gis.stackexchange.com/a/290059 [Page 32.]

[49] [Online]. Available: https://github.com/aladdinpersson/Machine-Learning-Collection/tree/master/ML/Pytorch/image_segmentation/semantic_segmentation_unet [Page 32.]

TRITA-EECS-EX-2024:11

€€€€ For DIVA €€€€

{
"Author1": { "Last name": "Hansen",
"First name": "Johanna",
"Local User Id": "u100001",
"E-mail": "johhanse@kth.se",
"organisation": {"L1": "School of Electrical Engineering and Computer Science",
}
},
"Cycle": "2",
"Course code": "DA231X",
"Credits": "30.0",
"Degree1": {"Educational program": "Master's Programme, Computer Science, 120 credits"
,"programcode": "TCSCM"
,"Degree": "Masters degree"
,"subjectArea": "Computer Science and Engineering"
},
"Title": {
"Main title": "Delineation of vegetated water through pre-trained convolutional networks",
"Language": "eng" },
"Alternative title": {
"Main title": "Konturteckning av vegeterat vatten genom förtränade konvolutionella nätverk",
"Language": "swe"
},
"Supervisor1": { "Last name": "Nordahl",
"First name": "Mats",
"Local User Id": "u100003",
"E-mail": "mnordahl@kth.se",
"organisation": {"L1": "School of Electrical Engineering and Computer Science",
"L2": "Computer Science" }
},
"Supervisor2": { "Last name": "Peña",
"First name": "Francisco J.",
"E-mail": "frape@kth.se",
"Other organisation": "Stockholm University"
},
"Examiner1": { "Last name": "Herman",
"First name": "Pawel",
"Local User Id": "u1d13i2c",
"E-mail": "paherman@kth.se",
"organisation": {"L1": "School of Electrical Engineering and Computer Science",
"L2": "Computer Science" }
},
"Cooperation": { "Partner_name": "Stockholm University"},
"National Subject Categories": "10201, 10206",
"Other information": {"Year": "2024", "Number of pages": "xiii,59"},
"Copyrightleft": "copyright",
"Series": { "Title of series": "TRITA-EECS-EX" , "No. in series": "2023:0000" },
"Opponents": { "Name": "A. B. Normal & A. X. E. Normalè"},
"Presentation": { "Date": "2022-03-15 13:00"
,"Language":"eng"
,"Room": "via Zoom https://kth-se.zoom.us/j/ddddddddddd"
,"Address": "Isafjordsgatan 22 (Kistagången 16)"
,"City": "Stockholm" },
"Number of lang instances": "2",
"Abstract[eng]": €€€€

In a world under the constant impact of global warming, wetlands are decreasing in size all across the globe. As the wetlands are a vital part of preventing global warming, the ability to prevent their shrinkage through restorative measures is critical. Continuously orbiting the Earth are satellites that can be used to monitor the wetlands by collecting images of them over time. In order to determine the size of a wetland, and to register if it is shrinking or not, deep learning models can be used. Especially useful for this task is convolutional neural networks (CNNs). This project uses one type of CNN, a U-Net, to segment vegetated water in satellite data. However, this task requires labeled data, which is expensive to generate and difficult to acquire. The model used therefore needs to be able to generate reliable results even on small data sets. Therefore, pre-training of the network is used with a large-scale natural image segmentation data set called Common Objects in Context (COCO). To transfer the satellite data into RGB images to use as input for the pre-trained network, three different methods are tried. Firstly, the commonly used linear transformation method which simply moves the value of radar data into the RGB feature space. Secondly, two convolutional layers are placed before the U-Net which gradually changes the number of channels of the input data, with weights trained through backpropagation during the fine-tuning of the segmentation model. Lastly, a convolutional auto-encoder is trained in the same way as the convolutional layers. The results show that the autoencoder does not perform very well, but that the linear transformation and convolutional layers methods each can outperform the other depending on the data set. No statistical significance can be shown however between the performance of the two latter. Experimenting with including different amounts of polarizations from Sentinel-1 and bands from

Sentinel-2 showed that only using radar data gave the best results. It remains to be determined whether one or both of the polarizations should be included to achieve the best result.

€€€€,
"Keywords[eng]": €€€€
Wetland delineation, Satellite image segmentation, Convolutional neural networks, Pre-training, Deep learning, Remote sensing €€€€,
"Abstract[swe]": €€€€

I en värld som ständigt påverkas av den globala uppvärmningen, minskar våtmarkerna i storlek över hela världen. Eftersom våtmarkerna är en viktig del i att förhindra global uppvärmning, är förmågan att förhindra att de krymper genom återställande åtgärder kritisk. Kontinuerligt kretsande runt jorden finns satelliter som kan användas för att övervaka våtmarkerna genom att samla in bilder av dem över tid. För att bestämma storleken på en våtmark, i syfte att registrera om den krymper eller inte, kan djupinlärningsmodeller användas. Speciellt användbar för denna uppgift är konvolutionella neurala nätverk (CNN). Detta projekt använder en typ av CNN, ett U-Net, för att segmentera vegeterat vatten i satellitdata. Denna uppgift kräver dock märkt data, vilket är dyrt att generera och svårt att få tag på. Modellen som används behöver därför kunna generera pålitliga resultat även med små datauppsättning. Därför används förträning av nätverket med en storskalig naturlig bildsegmenteringsdatauppsättning som kallas Common Objects in Context (COCO). För att överföra satellitdata till RGB-bilder som ska användas som indata för det förtränade nätverket prövas tre olika metoder. För det första, den vanliga linjära transformationsmetoden som helt enkelt flyttar värdet av radardatan till RGB-funktionsutrymmet. För det andra två konvolutionella lager placerade före U-Net:et som gradvis ändrar mängden kanaler i indatan, med vikter tränade genom bakåtpropagering under finjusteringen av segmenteringsmodellen. Slutligen tränade en konvolutionell auto encoder på samma sätt som de konvolutionella lagren. Resultaten visar att auto encodern inte fungerar särskilt bra, men att metoderna för linjär transformation och konvolutionella lager var och en kan överträffa den andra beroende på datauppsättningen. Ingen statistisk signifikans kan dock visas mellan prestationen för de två senare. Experiment med att inkludera olika mängder av polariseringar från Sentinell-1 och band från Sentinell-2 visade att endast användning av radardata gav de bästa resultaten. Om att inkludera båda polariseringarna eller bara en är den mest lämpliga återstår fortfarande att fastställa.

€€€€,
"Keywords[swe]": €€€€
Avgränsning av våtmarker, Segmentering av satellitbilder, Konvolutionella neurala nätverk, Förträning, Djupinlärning, Fjärranalys €€€€,
}

acronyms.tex

```
%%% Local Variables:
%%% mode: latex
%%% TeX-master: t
%%% End:
% The following command is used with glossaries-extra
\setabbreviationstyle[acronym]{long-short}
% The form of the entries in this file is \newacronym{label}{acronym}{phrase}
%                                      or \newacronym[options]{label}{acronym}{phrase}
% see "User Manual for glossaries.sty" for the  details about the options, one example is shown below
% note the specification of the long form plural in the line below

\newacronym{ndwi}{NDWI}{Normalized Difference Water Index}

\newacronym{sar}{SAR}{Synthetic Aperture Radar}
\newacronym{msi}{MSI}{Multispectral Imagery}
\newacronym{bce}{BCE}{Binary Cross Entropy}
\newacronym{iou}{IoU}{Intersection over Union}
\newacronym{cnn}{CNN}{Convolutional Neural Networks}
\newacronym{dem}{DEM}{Digital Elevation Model}
\newacronym{dsm}{DSM}{Digital Surface Model}
\newacronym{dtm}{DTM}{Digital Terrain Model}
\newacronym{rgb}{RGB}{Red, Green and Blue}
```