KTH ROYAL INSTITUTE
OF TECHNOLOGY

Degree Project in Technology

Second cycle, 30 credits

Using Satellite Images And Self-supervised Deep Learning To Detect Water Hidden Under Vegetation

IOANNIS IAKOVIDIS

Stockholm, Sweden, 2024

Using Satellite Images And Self-supervised Deep Learning To Detect Water Hidden Under Vegetation

IOANNIS IAKOVIDIS

Master's Programme, Machine Learning, 120 credits
Date: February 18, 2024

Supervisors: Francisco Pena Escobar, Michael Welle
Examiner: Danica Kragic
School of Electrical Engineering and Computer Science
Host organization: Stockholm University
Swedish title: Använda satellitbilder och Självövervakad Deep Learning Till Upptäck vatten gömt under Vegetation

Abstract

In recent years the wide availability of high-resolution satellite images has made the remote monitoring of water resources all over the world possible. While the detection of open water from satellite images is relatively easy, a significant percentage of the water extent of wetlands is covered by vegetation. Convolutional Neural Networks have shown great success in the task of detecting wetlands in satellite images. However, these models require large amounts of manually annotated satellite images, which are slow and expensive to produce. In this paper we use self-supervised training methods to train a Convolutional Neural Network to detect water from satellite images without the use of annotated data. We use a combination of deep clustering and negative sampling based on the paper ”Unsupervised Single-Scene Semantic Segmentation for Earth Observation”, and we expand the paper by changing the clustering loss, the model architecture and implementing an ensemble model. Our final ensemble of self-supervised models outperforms a single supervised model, showing the power of self-supervision.

Keywords

Machine Learning, Convolutional Neural Networks, Semantic Segmentation, Self-supervised Learning, Deep Clustering, Contrastive Learning, Ensemble Learning, Remote Sensing, Wetland Mapping

Sammanfattning

Under de senaste åren har den breda tillgången på högupplösta satellitbilder möjliggjort fjärrövervakning av vattenresurser över hela världen. Även om det är relativt enkelt att upptäcka öppet vatten från satellitbilder, täcks en betydande andel av våtmarkernas vattenutbredning av vegetation. Lyckligtvis kan radarsignaler tränga igenom vegetation, vilket gör det möjligt för oss att upptäcka vatten gömt under vegetation från satellitradarbilder. Under de senaste åren har Convolutional Neural Networks visat stor framgång i denna uppgift. Tyvärr kräver dessa modeller stora mängder manuellt annoterade satellitbilder, vilket är långsamt och dyrt att producera. Självövervakad inlärning är ett område inom maskininlärning som syftar till att träna modeller utan användning av annoterade data. I den här artikeln använder vi självövervakad träningsmetoder för att träna en Convolutional Neural Network-baserad modell för att detektera vatten från satellitbilder utan användning av annoterade data. Vi använder en kombination av djup klustring och kontrastivt lärande baserat på artikeln ”Unsupervised Single-Scene Semantic Segmentation for Earth Observation”. Dessutom utökar vi uppsatsen genom att modifiera klustringsförlusten och modellarkitekturen som används. Efter att ha observerat hög varians i våra modellers prestanda implementerade vi också en ensemblevariant av vår modell för att få mer konsekventa resultat. Vår slutliga ensemble av självövervakade modeller överträffar en enda övervakad modell, vilket visar kraften i självövervakning.

Nyckelord

Maskininlärning, Convolutional Neural Networks, Semantisk Segmentering, Självledd Inlärning, Deep Clustering, Contrastive Learning, Ensembleinlärning, Fjärranalys, Våtmarkskartering

Acknowledgments

I would like to first of all thank my project supervisor Francisco Pena Escobar who not only provided me with the expertise and resources necessary to complete this project, but also personally supported me at every step of the process. I would also like to thank my KTH project supervisor Michael Welle for his guidance when writing this report, as well as Francisco's other master's students, Ezio and Johanna, for their help annotating the Swedish wetlands dataset.

Stockholm, December 2023
Ioannis Iakovidis

Contents

List of Figures

List of Tables

Listings

List of acronyms and abbreviations

CNN	Convolutional Neural Network
DEM	Digital Elevation Model
NDVI	Normalized Difference Vegetation Index
NDWI	Normalized Difference Water Index
NIR	Near Infra-Red
SAR	Synthetic Aperture Radar

Chapter 1

Introduction

In this thesis we present a machine learning algorithm that aims to detect water from remote sensing images, with an emphasis on detection of water hidden under vegetation, without the need for annotated training data. The algorithm uses self-supervision training methods in order to train a model in the task of semantic segmentation of radar satellite images. Semantic segmentation is a computer vision task where the goal is to identify and locate different objects and/or areas in a picture by producing pixel-level labels for the picture. This task has found many applications in the field of remote sensing, especially considering the availability of numerous high-resolution satellite images in recent years. Some examples of such applications are water detection, land use classification and environmental monitoring[1].

The self-supervised semantic segmentation algorithm we use is based on an algorithm presented by Saha Sudipan et al. in the paper ”Unsupervised Single-Scene Semantic Segmentation for Earth Observation”[2]. The term self-supervised learning describes methods of training machine learning models without labels. The algorithm used in this paper is specifically designed for the task of semantic segmentation of remote sensing images and takes advantage of the spatial properties of those images to achive the self-supervised training. The authors demonstrate that the algorithm can obtain state-of-the-art results in a number of different remote sensing datasets.

The algorithm combines two different self-supervision methods, deep clustering and negative sampling. Deep clustering uses some method to cluster representations of the input data that are produced by the model into different classes and afterwards use those cluster assignments as labels to train the machine learning model. Negative sampling methods train a model by encouraging it to produce similar representations for so-called positive pairs

of inputs (such as a data point and an augmented version of it) and produce dissimilar representations for negative pairs (such as two random data points).

In this thesis we examine the performance of the algorithm on the task of segmenting water areas in satellite radar images of Swedish wetlands. After observing high variance in the performance of the algorithm across different experiments, we modified the loss function of the algorithm and implement an ensemble version of the model. Both of these changes improve the stability and performance of the algorithm on our datasets. We also replaced the model used in the original paper with a significantly smaller one, greatly increasing training speed while improving performance in our datasets.

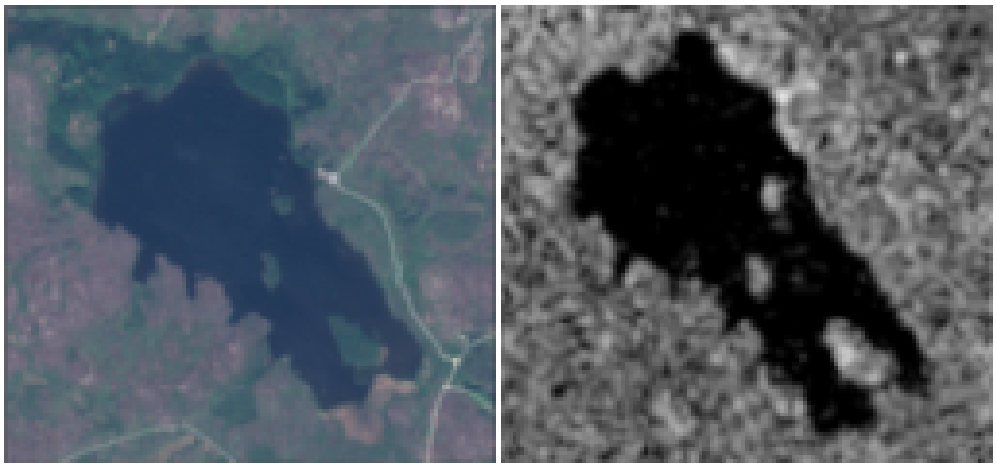
The datasets that we evaluate the algorithm on, the Örebro SAR dataset presented in the paper ”DeepAqua: Self-Supervised Semantic Segmentation of Wetlands from SAR Images using Knowledge Distillation”[3] and the Swedish Wetlands SAR dataset that was created by the author of this project and two other students, both consist of Synthetic Aperture Radar (SAR) images of Swedish wetlands taken by the Sentinel-1 satellites of the Copernicus Programme over a number of years. The labels were produced by manual delineation of the water surface extent of each image.

1.1 Background and problem context

The task of water detection concerns itself with detecting water bodies on the surface of the earth, such as rivers, lakes and wetlands. While we may think of such water bodies as static, in practice their surface extent, depth and even water composition can change drastically over time. Apart from the changes brought by seasonal weather variations, factors such as climate changes due to global warming or human infrastructure can cause permanent changes to surface water bodies. Moreover, extreme weather events can induce rapid changes that can have massive consequences such as floods. For these reasons, continuous monitoring of those water bodies is incredibly important.

Of particular interest to the field of water detection from remote sensing images are wetlands, whose water extent is often covered in small or large part by vegetation. Worryingly, due to a variety of factors including water drainage for agriculture and rising temperatures causing rising sea levels and thawing of permafrost, wetlands around the world are rapidly disappearing. This has resulted in the eradication of over half of all wetlands globally since the start of the twentieth century[4].

For the above reasons monitoring water bodies is very important. However, in-situ (on site) monitoring of them is very expensive and time



(a) Optical satellite image of wetland. (b) Radar satellite image of wetland.

Figure 1.1: Examples of optical and radar images of a wetland. Figures adapted from [3] used under CC BY 4.0 DEED licence.

consuming and in many cases infeasible. In recent years the wider availability of frequent and high resolution satellite images, both optical and radar, has provided scientists with incredible amounts of remote sensing data. This fact, combined with the rapid advancement of machine learning models during the last decade, with their ability to learn and model complex relations in data without requiring any input from human experts, have massively expanded the possibilities of the field of earth observation. Examples of optical and radar satellite images of wetlands can be seen in Figure 1.1.

Unfortunately most methods for training machine learning models require large amounts of annotated data. This is especially problematic for many remote sensing applications, which often require detailed annotations from field experts. In recent years the field of self-supervised machine learning has developed training methods that aim to train neural network models without the use of annotated data. These models have been applied to various remote sensing tasks such as scene classification, land cover classification and change detection, often achieving performances comparable to those of models trained with annotated data[5]. However, to the best of our knowledge, until now self-supervised machine learning methods have seldomly been applied to the task of water detection from satellite images of wetlands.

1.1.1 Previous work in water detection from satellite images

In order to detect water bodies from remote sensing images, traditionally optical images and water indices such as the Normalized Difference Water Index (NDWI)[6] are used. These indices are specific combinations and ratios of different spectral bands that are especially adept at detecting water. In order to automatically segment water from non-water areas, the simplest method is to use a thresholding rule based on the value of one or more of those indices. A more sophisticated method is to use either unsupervised or supervised classification, depending on whether annotated data are available or not. The most common classification algorithm is the Maximum Likelyhood method[7]. Early attempts at applying machine learning methods to water mainly consisted of applying simple machine learning classification methods, such as random forests and support vector machines, to the same data[8].

Although detecting water bodies from satellite images is a relatively simple task in most cases, wetlands have specific characteristics that make remote detection more difficult. The two main confounding factors are wetlands' heterogeneous nature and the fact a a large part of their water surface is often covered by vegetation, which optical signals cannot penetrate. Moreover, the small size of many wetlands means that high resolution satellite data are needed to detect them[9].

1.1.2 Previous work in semantic segmentation

Semantic segmentation is the task of splitting an image into different semantically consistent neighbourhoods. More specifically, each pixel of the image is assigned a class. This is a much more difficult problem that the more common task of image classification, since it demands not only the detection of objects in the image but also their precise location. Semantic segmentation is one of the most important tasks in remote sensing applications, and is required in tasks such as land cover classification and water detection[8]. Although thresholding methods and clustering methods have been used in the past to perform semantic segmentation of satellite images by categorizing each pixel in an image independently, for many remote sensing tasks an understanding of the area neighbouring the pixel is necessary for an accurate classification. For this reason, in the last few years convolutional neural networks have become the main machine learning tool used for semantic segmentation.

Machine learning methods such as random forests, support vector

machines and small artificial neural networks have been used to study remote sensing data and and satellite data in particular. However the field was revolutionized with the invention of the Convolutional Neural Networks (CNNs). CNNs are deep neural networks that are designed explicitly for images as an input[10]. The main distinguishing feature of CNNs is the convolution kernel. A convolution kernel is a group of neurons (usually in a square shape) that scans over the the input image, performing a 2-D convolution on it. This results in a new image where each pixel's value is influenced by the values of all pixels in the original pixel's neighbourhood. A group of convolutional kernels create a convolutional layer, and series of convolutional layers can extract very complex patterns from the input images. The other signature layer of machine learning models designed for semantic segmentation is the pooling layer. In order to allow convolutional layers to take into account larger areas of the image without ballooning the number of trainable parameters, pooling layers are inserted between different convolutional layers. These pooling layers reduce the image's size, usually by half, by replacing all the pixels in an area by an approximation (usually their mean or max value). These two types of layers, along with activation layers such as Relu or Sigmoid[11], are the core building blocks of all convolutional neural networks. While in many application that require a image-wide output such as image classification one or more fully connected layers are attached to the end of the CNN to produce the desired result, for semantic segmentation an 1-width convolutional kernel is used to output predictions for each pixel.

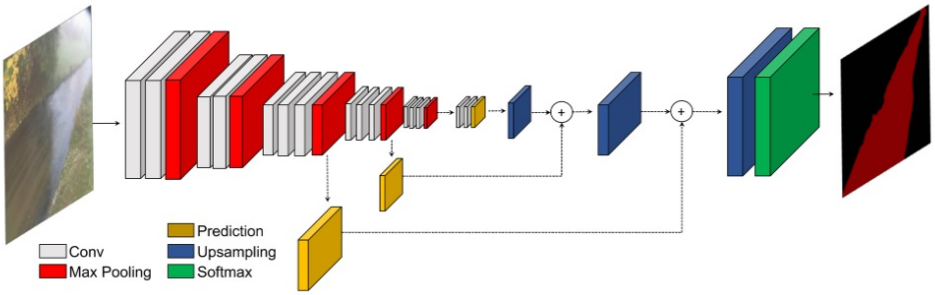


Figure 1.2: Fully convolutional network architecture. Figure adapted from [12] used under CC BY-NC 4.0 Deed licence.

Although conventional CNNs can be used for semantic segmentation of images, the pooling layers present a problem. By repeatedly reducing the size of the image, the output of the network has a much lower resolution that the original image. To solve this problem, the fully convolutional network was

created[13]. This network outputs predictions of various resolutions from a variety of layers in the networks architecture. Then, in order to produce high-resolution predictions while taking advantage of the higher level-features of the later layers, the low-resolution predictions are upsampled using reverse convolution layers and combined with the higher-resolution predictions form previous layers to produce the final predictions. The architecture of a fully convolutional network can be seen in Figure 1.2.

In "U-Net: Convolutional Networks for Biomedical Image Segmentation"[14] the authors iterated on the fully connected network to create U-Net. In this network after the usual convolutional network a sort of 'reverse' convolutional network is attached with the purpose of upscaling the deep features extracted by the first CNN to the resolution of the original image. In this reversed architecture the convolutional and activation layers are similar, but the pooling layers are replaced by reverse convolution layers. In order to correctly upscale the discovered features, the convolutional layers receiving this upscaled image also take as input the output of the convolutional layer on the corresponding level of the first half of the architecture. The U-Net architecture is presented in more detail in Section 2.3.4.

1.1.3 Previous work in wetland detection and classification using machine learning

The paper "Very Deep Convolutional Neural Networks for Complex Land Cover Mapping Using Multispectral Remote Sensing Imagery"[15] was the first one to perform a comprehensive study on the performance of different CNNs at the task of classifying different wetland classes from satellite multispectral images. A wide variety of CNN architectures were used and the effect of pretraining the models on the ImageNet[16] dataset was tested. The paper concluded that most of the CNNs tested achieved state-of-the-art results in the task, outperforming simpler machine learning methods such as Random Forests. Moreover, the study found that using more spectral bands increased the performance of the models. Finally, pretraining on the ImageNet dataset didn't seem to help, which is probably due to a combination of the fact that models pretrained on ImageNet can only make use of the RGB bands and the fact that satellite images are very different in nature from those in ImageNet and most other machine learning image training datasets.

Jiang et. al.[17] examined different ways of using CNNs to combine different types of data for the task of image-level wetland detection and classification. The data used for the classification was RGB Satellite images

and a combination of Digital Elevation Model (DEM) data, the Near Infra-Red (NIR) spectrum channel and the Normalized Difference Vegetation Index (NDVI). The model used was the CNN Resnet-50[18] pretrained on the dataset ImageNet. The authors compared the performance when using only RGB data, when using only DEM/NIR/NDVI data and when combining them. The data were either fused before being inserted into the model, or after being inserted each into a different model the models were merged after some layer. The DEM/NIR/NDVI model performed worst on classifying the wetlands, with the RGB model showing better performance. However, all fusion models outperformed the single models, with the improvement increasing the later the fusion takes place. This suggests that when combining different types of data, it can be more beneficial to combine high-level features instead of lower-level ones.

In 2020 Cui et. al.[19] used a modified U-Net architecture to improve the semantic segmentation of coastal wetlands in China. The architecture featured 2 important modifications. The first modification is the use of depthwise separable convolution kernels, which perform spatial convolution independently on each channel before using an 1×1 convolution to project all the channels to the new channel space. This method drastically reduces the number of trainable parameters in the model, which in turn decreases the data needed for training. The second change was to include deconvolution layers in the skip connection, enriching the info passed to the decoder.

Another interesting model for semantic segmentation of wetlands was designed by Pham et. al. in the paper "A new deep learning approach based on bilateral semantic segmentation models for sustainable estuarine wetland ecosystem management"[20]. Here two different CNNs are used to process the images, and their encodings are combined at the end using an attention mechanism[21]. The two models consist of a shallower CNN that focuses on extracting spatial information from the image and a deeper CNN that extracts higher-level contextual information.

One important parameter regarding tasks involving satellite images is that in a lot of cases images from several different dates are available for each place. This fact creates an opportunity for using machine learning methods that take advantage of this multitemporal dataset. For that reason Hosseiny et. al.[22] used an ensemble of three different models to perform wetland classification from optical and radar satellite images. The first model is a 2-D CNN that processes the radar satellite image, the second is a 3-D CNN that processes the optical satellite image and finally the third one is an LSTM-based[23] model is used to extract temporal information from the data. They concluded

that incorporating temporal features improved the results of the classification model.

1.2 Self-supervised machine learning

Although deep learning networks have achieved incredible results in a wide variety of tasks, the huge number of annotated data they require for training is a huge limitation for their application in many fields. This problems seems to only get worse each year, with bigger and bigger neural networks being designed containing billions of trainable parameters, as the number of training data a network requires is closely correlated to the number of trainable parameters it has. While it has been possible in many cases to create large training datasets by outsourcing the annotation of data to third parties or even to use publicly submitted information, methods like these are not possible for every field. Regarding semantic segmentation in particular, this task requires annotating every pixel in each image, making the creation of large training datasets extremely time consuming. The problem in even worse in the field of remote sensing, where annotating data such as satellite images often requires expert knowledge or even in-situ measurements.

In order to overcome this limitation, several training methods that do not require annotated data have been developed in the last few years. These methods usually fall into one of two categories: unsupervised training methods and self-supervised training methods. Unsupervised methods try to split the training data into groups using only dimensionality reduction and clustering methods. Self-supervised training methods, on the other hand, rely on training the machine learning model on tasks that can automatically be created from the input data, which are called pretext tasks.

Self-supervised training methods can be sorted into three categories: generative methods, predictive methods and contrastive methods[24]. A comparison of the basic mechanics of the three categories can be seen in Figure 1.3.

Generative methods are perhaps the most conceptually simple: they train the model by making it output data from the same distribution as the input data. The two best known types of generative models used are Variational Autoencoders and Generative Adversarial Networks.

Variational autoencoder models consist of two sub-models, an encoder E and a decoder D. They take as input a sample from the data and their goal is to recreate it as closely as possible. The input x is first processed by the encoder module, which outputs a representation z of the input data. This

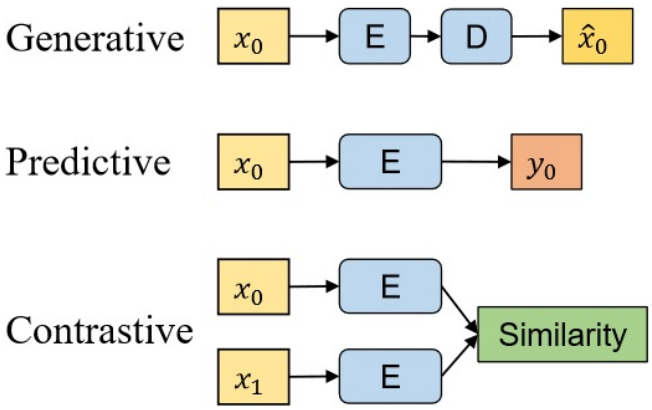


Figure 1.3: A comparison of the general structures between generative, predictive, and contrastive self-supervised learning. E and D refer to the encoder and decoder network. While generative and predictive methods calculate the loss in the output space, contrastive methods calculate the loss in the representation space. Figure from [24], Copyright ©2022, IEEE.

representation then is used as a parameter in a variational distribution P. The decoder then samples a representation from that variational distribution and uses it as input to attempt to recreate the original image at its output. with the training loss usually being $\parallel x - D(V(E(x))) \parallel$. The encoder is therefore trained to extract the most informative features of the input. In order to avoid the trivial solution of both the encoder and the decoder simply mirroring their input as their output, constrains must be placed on the representation z. Either the representation z must be of significantly lower dimentionality of the input x, or some other limitation must be used, such as an extra loss encouraging the representation z to be sparse.

Generative adversarial networks on the other hand do not try to recreate specific training data, but simply generate outputs from the same distribution as that of the training data. Generative adversarial networks consist of two sub-modules, a generator G and a discriminator D. The generator G takes as input a randomly sampled representation vector z and outputs a fake sample $\overline{x}$, while the discriminator D tries to discriminate between real and fake samples.

In the last few years other generative model types, such as denoising diffusion probabilistic models[25] and and normalizing flow models[26], have seen success in the field of self-supervised learning[27, 28].

Instead of trying to replicate the low-level details of the training data like generative models, predictive models instead are trained on tasks that can be

automatically generated from the input data without human annotation. The most common type of task are spatial tasks, like determining the relative position of different patches extracted from an image or determining the correct orientation of an image that has been rotated. In a similar vein, for multitemporal data like video, pretext tasks that ask the model to order frames can be used. Finally, there are tasks that focus on reconstructing a missing part of the input. This can mean reconstructing part of an image that has been masked, or coloring a grey-scale version of the input image.

The third and perhaps most interesting category of self-supervised learning are contrastive training methods. Instead of focusing on one data sample at a time, contrastive training methods compare the model representations of multiple data points.

One type of contrastive training methods are clustering methods. These methods utilise k-means clustering or other, mode complex algorithms to separate the data representations produced by the training model into different groups. These groups then are treated as pseudo-labels, and the model weights are updated using usual machine learning losses such as cross entropy loss[29].

Other contrastive training algorithms, called negative sampling algorithms, instead train the model by giving the model pairs of inputs that should have similar representations, usually an original data point and an augmented version of it, and training the model to output similar representations. In order to avoid the degenerate solution of outputting the same representation regardless of input, the model is simultaneously penalized for producing similar representations between so-called negative samples. For most negative sampling algorithms these negative samples are other training data samples in the same batch. In order to be able to use more negative samples and not be constrained by the batch size, other algorithms create and update a separate memory of representations to be used as negative samples.

Finally, a third category of contrastive training algorithms are those that rely on knowledge distillation, the process of training a student model using a teacher model as supervision. While this method is usually used to train a small student model from the predictions of a larger teacher model to achieve a reduction in model size and complexity, in self-supervised learning the process is different. Here the two models are used to produce two representations of the same input and are trained to produce similar representations. In order to avoid the two models degenerating into the same one, they use different architectures or training methods.

1.2.1 Previous work in self-supervised machine learning in remote sensing

The fact that there are multiple satellites with different sensors providing observation data at various dates means that there are many different data samples for the same area, which many papers have exploited to produce positive and negative samples for contrastive-learning-based self-supervised training methods. Scheibenreif et. al.[30] adapted the SimCLR contrastive learning method[31] to perform multi-label image classification of satellite images by using 2 different ResNet18-based encoders, one for optical images and one for radar images. Contrasting learning is achieved by using images of the same area but from different sensors as positive pairs, while images from different areas are used as negative pairs. Two different contrastive methods are studied. The first one is the standard contrastive learning method where after the encoder produces a per-pixel feature vector for each image, a projector head transforms those vectors into a single 1-dimensional image feature vector, which is used for the contrastive loss. The second method omits the projector head and compares the 3-D encodings directly. The authors found that using self-supervised learning and then fine-tuning the resulting model with 10% of the annotated training data outperformed a fully-supervised model. Like in the original paper, their results showed that the use of projector heads is beneficial for contrastive learning.

A similar method was used in the paper ”Self-supervised vision transformers for land-cover segmentation and classification”[32]. As with the previous paper, optical and radar images of the same areas were fed into two models and the embeddings, after being passed through fully-connected contrastive heads, are compared with contrastive learning. This paper used CNNs enhanced with an attention mechanism as encoders and again found that pretraining with self-supervision and using 10% of the training dataset outperformed fully-supervised training methods.

Manas et.al.[33] took advantage of the multitemporal nature of satellite data by using satellite images of the same area during different seasons to create positive pairs for contrastive learning. This ”seasonal augmentation” is complimented with other traditional augmentations and multiple projector heads are used to produce positive groups of images from multiple augmentation methods. As usual, images of other areas (augmented and not) are used as negative samples. The authors found that this self-supervision method outperforms pretraining on the Imagenet dataset and Momentum Contrast pretraining[34] in the task of change detection in satellite images.

Another method that can be used to generate similar and dissimilar pairs of data points for contrastive learning is unsupervised clustering methods. Cai et. al.[35] used an unsupervised image segmentation algorithm to segment remote sensing images and used patches from the same segment as positive pairs and patches from different segments as negative pairs to pretrain a CNN model for change detection.

Wu et. al.[36] used a generative adversarial network that was trained to reconstruct remote sensing images in order to train a semantic segmentation model to the task of change detection. In this paper initially a generative adversarial network is trained to recreate remote sensing images. Afterwards, the segmentation network is trained by taking the original and the changed images as input and trying to predict which areas need to be ignored so that the generator model can reconstruct the changed image from the original.

Finally, Pena et. al.[3] used knowledge distillation to train a U-Net model on the task of water detection. They use the NDWI method on optical satellite images of wetlands in order to automatically generate labels. Afterwards, these labels are combined with SAR satellite images of the same areas to train a machine learning model to detect water from SAR satellite images, effectively transferring the labels from the optical to the radar domain.

1.3 Ensemble learning

Ensemble learning[37] refers to a field of machine learning that combines the predictions of multiple models to achieve improved performance and reduced variance over a single model. More specifically, by combining multiple base models ensemble models are better at generalizing to unseen data, are more often able to avoid being trapped into local minima and are able to, in aggregate, create more complex representations of the data. In order to achieve those results, it is important that all the models of the ensemble are sufficiently different from each other. Most common ensemble machine learning methods achieve this by training a number of models of identical architecture with different subsections of the training data and then combining their predictions on test data using an (often weighted) voting method.

The two most common ensemble training methods are bagging and boosting. The bagging method ensures difference between the models by simply training each base model using a sampling of the original dataset and then combining the different predictions with a simple majority vote. Boosting on the other hand trains each model sequentially. The first model is trained normally on the whole training dataset. For the second model each data point

is assigned a weight based on whether the first model correctly predicts it or not in order to focus the new model on the data that the first one cannot deal with. This process is repeated for all the models in the ensemble and each model is given a vote weight based on it's performance on the whole dataset.

1.4 Motivation

In this section we will detail why we believe our research is important to the scientific community but also more broadly society as a whole.

Our work is of interest to the scientific community because our aim is to increase our understanding of the field of self-supervised machine learning. While machine learning models have achieved enormous success in the last decade, one of the biggest limitations in their applications is the lack of sufficient training data. This becomes more and more true as model sizes increase over the years. Acquiring sufficient annotated data for machine learning models can be time consuming and expensive, and often brings about other concerns such as privacy issues or exploitative labour. For the above reasons advancing our understanding of self-supervised learning can have a major impact in the future of the field of machine learning.

Our research also has important implications to society as a whole. Water bodies are incredibly important to adjacent regions, since they provide numerous benefits to any nearby ecosystems, acting as sources of water and food for nearby flora and fauna while also providing a habitat for for numerous forms of life. At the same time, surface water bodies provide invaluable services to nearby human activities contributing to tasks such as crop irrigation, transportation and energy production[38]. Wetlands in particular hold immense importance to the global ecosystem. Even though they cover only about 6% of the earth's surface, they contain a huge percentage of the earth's biodiversity and provide a variety of other benefits to nearby regions, such as flood protection and water quality improvement[39]. The most important function of wetlands however is their ability to store enormous amounts of greenhouse gasses, which makes their protection vital in the fight against global warming[40]. For the above reasons expanding our ability to efficiently monitor our planet's water bodies is of outmost importance.

1.5 Aim

The aim of this thesis is to improve scientists' ability to efficiently monitor water bodies, especially those fully or partially covered under vegetation, using remote sensing images. More specifically, we hope to improve the performance of self-supervised machine learning models on the task of semantic segmentation of SAR satellite images.

On a more granular level, our various experiments attempt to shed a light on the mechanics of self-supervised semantic segmentation and examine how differences in the training algorithm and model architecture influence the machine learning model's performance on this task.

1.6 Research questions

In order to achieve the aim of this thesis we will attempt to answer the following research questions:

- Can a machine learning model trained exclusively through deep clustering and negative sampling methods on the task of semantic segmentation of radar satellite images produce segmentations that delineate water from land areas?
- What is the relationship between the number of classes that the model produces and their suitability for the task of water detection?
- How do changes in model architecture effect the performance of the algorithm?

1.7 Delimitations

In this section we will describe the delimitations of our project in order to present a clear view of what we seek to accomplish.

The most important delimitation of our machine learning algorithm is that while its purpose it to create segmentations that separate water areas from land areas in the satellite images, it doesn't actually assign each of those segmented areas to the class of water or land. Instead, the model segments the image into different classes that are then assigned to water or land using ground truth annotations in order to judge the performance of the model. This is due to the fact that the model receives no input except from the SAR satellite image.

Another important delimitation of our method is that we do not attempt to make any finer differentiations than that of water or land. We do not test our model segmentations of whether they can differentiate between different kinds of water bodies (for example lakes, wetlands, rivers) or if they differentiate between open and vegetated water.

Chapter 2

Self-supervised semantic segmentation of SAR satellite images for water detection

In order to detect water in satellite images, with an emphasis on water hidden under vegetation, without the use of annotated data we use a machine learning model to perform self-supervised semantic segmentation on SAR satellite images. Afterwards each model class is assigned the label of water or land based on its intersection over union with manual annotations of the satellite images. The algorithm we use to perform the self-supervised training was first presented in the paper "Unsupervised Single-Scene Semantic Segmentation for Earth Observation"[2]. This algorithm was designed specifically for the task of semantic segmentation of remote sensing images and takes advantage of the idiosyncrasies of remote sensing images to aid in the training of the machine learning model used without the use of annotated data. In this chapter we will provide a detailed explanation of the algorithm and the model it's used to train, along with the modifications we made. First of all however we will precisely define the task of semantic segmentation.

2.1 Semantic segmentation

In machine learning, semantic segmentation is a vision task that seeks to assign a class label to each pixel of an input image. More specifically, we denote the input image as X. This rectangular image has spatial dimensions $R \times W$ with $r, w \in \mathbb{N}$, such that an arbitrary pixel in the image can be denoted as $X_{(r,w)}$ with $0 \leq r < R$ and $0 \leq w < W$. Although our datasets consist of single

channel images, the input image can have any number of channels $V \in \mathbb{N}$, such that the dimensions of the image are $R \times W \times V$ and each pixel $X_{(r,w)}$ is an v-dimensional vector. The goal is to produce a new image Y with dimensions $R \times W \times 1$ with each pixel having value $0 \leq X_{(r,w)} < K$ and$X_{(r,w)} \in \mathbb{N}$, with $K \in \mathbb{N}$ being the number of different classes the model cal label pixels as.

2.2 Unsupervised Single-Scene Semantic Segmentation for Earth Observation

In the paper "Unsupervised Single-Scene Semantic Segmentation for Earth Observation"[2] the authors use a combination of deep clustering and negative sampling methods to perform self-supervised semantic segmentation of a large remote sensing image. This task is accomplished through two algorithms. The first algorithm uses a machine learning model trained with self-supervised techniques to segment the input image into arbitrary classes, while the second algorithm takes as input the segmentation image produced by the model and the image of ground truth segmentations and corresponds each of the model produced classes to the ground truth class it has the greatest intersection with. In this section we will first describe the machine learning model architecture used and the algorithm used to train it. Afterwards we will describe the algorithm used to perform the semantic segmentation with the trained model.

2.2.1 Model training

Following the example of other machine learning projects that use negative sampling methods, the authors of "Unsupervised Single-Scene Semantic Segmentation for Earth Observation" use a Siamese network structure that utilises two models that have the same architecture to produce two different representations of the input data. We denote as X the large remote sensing image of size $R \times W$ that the model will be trained on. In order to produce positive data pairs for negative sampling, an augmented image of the same size, denoted as $\hat{X}$, is created by applying a Gaussian blur filter to X. Each of those images is split into a large number of smaller image patches of size $R' \times W'$ with $R' < R$ and $W' < W$, from which batches of training samples can be drawn during the training process. A batch of size B of patches from the original image X is denoted as $\mathcal{X} = \{x^1, ..., x^B\}$. The corresponding batch of patches from the augmented image $\hat{X}$ is denoted as $\hat{\mathcal{X}}$ and contains the patches that spatially correspond to the patches in $\mathcal{X}$. Moreover, a third batch $\hat{\mathcal{X}}'$ is

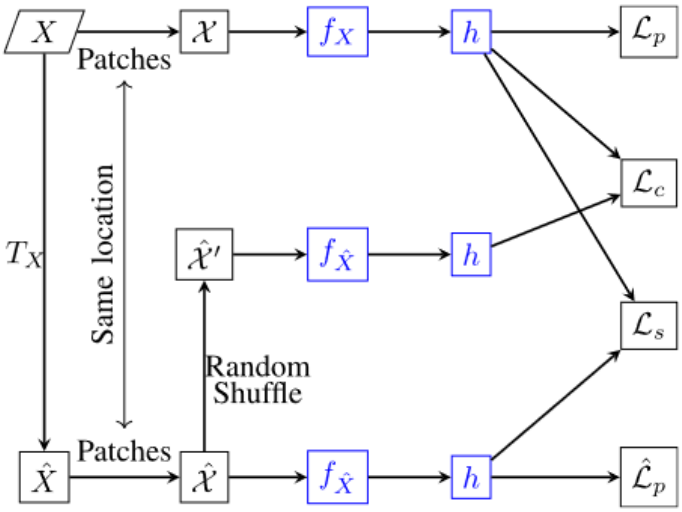


Figure 2.1: Computation of losses for self-supervised segmentation training framework. Network components are shown in blue outlined nodes to distinguish them from inputs, intermediate tensors, and losses. Figure from [2] used under CC BY 4.0 licence.

created by shuffling the order of the patches in $\hat{\mathcal{X}}$. These batches are processed by two different machine learning models, one for the batch of patches from the original image $\mathcal{X}$ and one for the batches of patches from the augmented image $\hat{\mathcal{X}}$ and $\hat{\mathcal{X}}'$. The two models, called projection modules, are denoted as f_X and $f_{\hat{X}}$ and have the same architecture but independent weights. During training the projection module f_X processes the batch of patches $\mathcal{X}$ and outputs pixel-level representations denoted as $f_X(\mathcal{X})$. In parallel, the projection module $f_{\hat{X}}$ processes $\hat{\mathcal{X}}$ and $\hat{\mathcal{X}}'$ to produce the representations $f_{\hat{X}}(\hat{\mathcal{X}})$ and $f_{\hat{X}}(\hat{\mathcal{X}}')$. Afterwards, each of those representations is passed through a single model, called the prediction module and denoted as h, which outputs the final class activation volumes $h(f_X(\mathcal{X}))$, $h(f_{\hat{X}}(\hat{\mathcal{X}}))$ and $h(f_{\hat{X}}(\hat{\mathcal{X}}'))$. More specifically, $h(f_X(\mathcal{X}))$, $h(f_{\hat{X}}(\hat{\mathcal{X}}))$ and $h(f_{\hat{X}}(\hat{\mathcal{X}}'))$ each have dimensions $B \times R' \times W' \times K$, where K is the number of classes which is a hyperparameter of the algorithm and not necessarily equal to the number of classes contained in the ground truth segmentation of X. These class activation volumes are used to calculate the losses that are used to train the three different modules.

The first kind of loss the algorithm uses is a self-supervised deep clustering loss. More specifically, for each pixel $x^b_{(r,w)}$ in image x^b in batch $\mathcal{X}$ the architecture has produced a K-sized vector of activation values for the model classes denoted as $h(f_X(x^b))_{(r,w)}$. The class assignment of the model for

the pixel is the index of the max value of the vector, denoted as $c^b_{(r,w)} = \arg\max_k h(f_X(x^b))_{(r,w),\,k}$. The algorithm treats those class assignments as pseudo-labels, effectively using the prediction module to simulate pixel-level deep clustering. Afterwards these pseudo-labels are used to train the model using a per-pixel cross-entropy loss. This loss is defined as $l^b_{(r,w)} = -\log h(f_X(x^b))_{(r,w),\,c^b_{(r,w)}}$. Aggregating this loss over each pixel in each image in batch $\mathcal{X}$ we get the deep clustering loss $\mathcal{L}_p = \sum_b \sum_r \sum_w l^b_{(r,w)}$ that is used to update the weights of the projection module f_X and the prediction module h. The same deep clustering process is repeated with the activation values $h(f_{\hat{X}}(\hat{x}^b))_{(r,w)}$ of the patches of the augmented image in batch $\hat{\mathcal{X}}$ to calculate the clustering loss $\hat{\mathcal{L}}_p$ that is used to update the weights of the projection module $f_{\hat{X}}$ and the prediction module h.

In order to compliment these deep clustering losses, the authors add a spatial consistency loss by using the patches of the augmented image in batch $\hat{\mathcal{X}}$ as positive pairs with the patches from the original image in batch $\mathcal{X}$. This is achieved by adding the per-pixel absolute difference between the class probability predictions of the spatially corresponding patches of the two images as a loss for the modules, which encourages the model to output similar predictions for the original and the augmented patches. This loss is denoted as $L_s = \|h(f_X(\mathcal{X})) - h(f_{\hat{X}}(\hat{\mathcal{X}}))\|$. In order to produce negative pairs that encourage the model to produce dissimilar predictions for different inputs, the authors again take advantage of the spatial nature of the remote sensing images and pair up patches from $\mathcal{X}$ with the corresponding shuffled patches from $\hat{\mathcal{X}}'$. This loss is denoted as $L_c = -\|h(f_X(\mathcal{X})) - h(f_{\hat{X}}(\hat{\mathcal{X}}'))\|$.

The total loss of the model is $L = L_p + \hat{L}_p + L_s + L_c$. This loss is used to train the model with standard iterative optimization through back-propagation using a stochastic gradient descent algorithm with momentum. The training is carried for $\mathcal{I}$ epochs, but also the training procedure is repeated $\mathcal{J}$ times for each batch of images. It should be noted that during the first epoch of training only the deep clustering losses L_p and $\hat{L}_p$ are used and the other two losses are ignored, due to a difference in the range of values of the different losses. The model weights are initialized using the He initialization strategy[41]. The training algorithm can also be seen in Listing 2.1 and in Figure 2.1.

Listing 2.1: Algorithm for Self-Supervised Training for Semantic Segmentation in Earth Observation Data from [2]

```
Initialize the weights of the network
for i ← 1 to 𝓘 do
    Sample 𝒳 = x^1, ..., x^B from X
```

Obtain spatially corresponding B patches from $\hat{X}$, denoted as $\hat{\mathcal{X}} = \hat{x}^1, ..., \hat{x}^B$
Obtain $\hat{\mathcal{X}}'$ by randomly shuffling $\hat{\mathcal{X}}$
for $j \leftarrow 1$ to $\mathcal{J}$ **do**
for $b \in B$ **do**
$y^b = h(f_X(x^b))$
$\hat{y}^b = h(f_{\hat{X}}(\hat{x}^b))$
$\hat{y}'^b = h(f_{\hat{X}}(\hat{x}'^b))$
end for
Estimate pseudo label losses $\mathcal{L}_p, \hat{\mathcal{L}}_p$
Estimate spatial consistency loss $\mathcal{L}_s$
Estimate loss similar to contrastive learning $\mathcal{L}_c$
Use the losses to train the network
end for
end for

2.2.2 Image segmentation and class matching

After the model has been trained using the algorithm 2.1, the modules f_X and h can be used to produce a semantic segmentation of any image. For any input image x the class assignment for a pixel in position (r, w) is $c^b_{(r,w)} = \arg\max_k h(f_X(x^b))_{(r,w),\,k}$. However, since the model was trained in an self-supervised fashion without any of the ground truth labels, these is no inherent correspondence between the classes assigned by the model and the ground truth classes. In fact, the authors suggest that the model should be trained to assign slightly more classes than the number of classes in the ground truth segmentation. In order to correspond the model-produced classes with the ground truth ones, the algorithm 2.2 is used. Given a set of N ground truth classes $\{\mathcal{S}_j\}_{j=1}^N$ ordered by number of pixels and a set of K model classes $\{\mathcal{T}_i\}_{i=1}^K$, the algorithm assigns to each ground truth class the model class that it has the biggest intersection with. It should be noted that the reference and segmented image described in the algorithm refer to the large input image X and the collage of model segmentations of all the patches from that image. That is to say, each model class is assigned to water or land over the entire dataset and that assignment does not change between different image patches.

Listing 2.2: Algorithm for Matching self-supervised Segmented Map to the Reference Map from [2]

Input: Reference image and set of its constituent N classes $\{\mathcal{S}_j\}_{j=1}^{N}$, ordered by number of pixels
Input: Obtained segmented image and set of its constituent K classes $\{\mathcal{T}_i\}_{i=1}^{K}$
$\hat{\mathcal{T}}^1 \leftarrow \{\mathcal{T}_i\}_{i=1}^{K}$
for $j \leftarrow 1$ to N **do**
    Find $\mathcal{T}_{\mathbb{D}^j}$ as the class in $\hat{\mathcal{T}}^j$ having highest intersection/overlap with $\mathcal{S}_j$
    Assign $\mathcal{T}_{\mathbb{D}^j}$ as match for $\mathcal{S}_j$
    $\hat{\mathcal{T}}^{j+1} = \hat{\mathcal{T}}^j \setminus \mathcal{T}_{\mathbb{D}^j}$
end for
Assign any remaining class in $\hat{\mathcal{T}}^{N+1}$ to background.

2.2.3 Model architecture

The authors of "Unsupervised Single-Scene Semantic Segmentation for Earth Observation" opt to use simple and lightweight CNN-based architectures for the modules, since the algorithm is designed to work on a single large remote sensing image, which is typically of relatively low resolution and also limits the number of patches available. More specifically, in the experiments in the paper the projection modules used consist of four to six blocks, each consisting of a convolutional layer with a 3×3 sized kernel, a ReLU activation layer and a batch normalization layer. The convolutional layers in the middle of the modules have a higher number of channels than those at the beginning and the end. A batch normalization layer[42] is a layer that aims to increase the training speed of machine learning models and perform regularization by normalizing the data at it's input to a standard mean value and variance. This allows later layers to receive inputs with a standardized mean and variance regardless of the changing weights of the previous layers, which in turn means that higher learning rates can be used to train the model, allowing for faster convergence. Moreover, these layers act as a form of regularization on the model, since the output of the model for a given data point may change depending on the rest of the data points in the mini batch. The prediction module used is much simpler, consisting of a single convolutional layer with a 1×1 sized kernel followed by a rectified linear unit activation function and a batch normalization layer.

2.3 Our implementation

Like we mentioned previously, for the experiments in this project we followed the training method from the paper ”Unsupervised Single-Scene Semantic Segmentation for Earth Observation”. However, after some initial experiments and in order to adapt the method to better for our problem and datasets, we made a number of modifications and additions. We describe those changes bellow.

2.3.1 Modification on the number of epochs and iterations

As we mentioned in 2.2.1, instead of the usual training strategy of training the model once on each batch of data every epoch, the authors repeat the training procedure for each batch multiple times. During early experiments we found no noticeable difference in the performance of the model on our datasets when using this training strategy when compared to the classic training strategy. For this reason we use the simpler training strategy of processing each batch once per epoch for our experiments.

2.3.2 Modification on the number of classes

In the original paper, the authors found that the algorithm performed best when the model was trained to produce slightly more classes K than the ground truth number of classes N ($K = 8$ for $N = 6$ and $K = 12$ for $N = 8$). However, during early experiments we discovered that following this strategy did not lead to satisfactory results in our dataset, since the different classes the model creates are not always conductive to separate water from land. Some examples of this problem can be seen in Figures 2.2 and 2.3. After experimentation we settled on $K = 10$ as the best setting for our datasets. Some experiments regarding the effect of K on the performance of the algorithm can be found in section 3.6.5.

2.3.3 Modification on the model losses

As we mentioned in 2.2.1, the authors of the original paper use only the deep-clustering losses for the first epoch of training due to the fact that they have a different range of values than the other two losses. During our experiments in our datasets however we observed something different. More specifically we

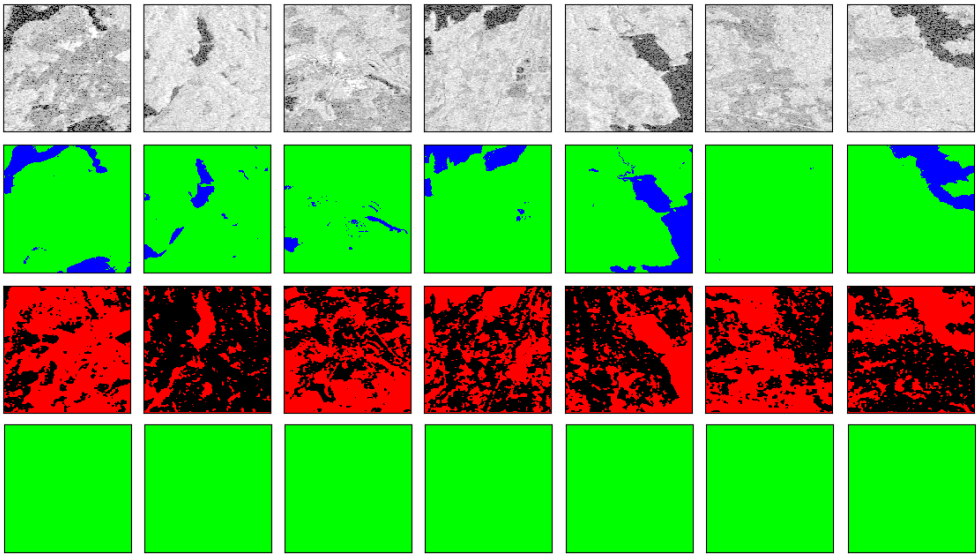


Figure 2.2: Examples of classes created by the algorithm in wetland SAR images for K=2. The first row shows the original SAR images, the second row shows shows the ground truth segmentation, the third row shows the model-produced classes and the fourth row shows the final class assignments. In this example both of the classes produced by the model have a greater intersection over union with the land class of the ground truth annotations. The class colors in the third row are arbitrary.

noticed that absolute value of the dissimilarity loss L_c continued to increase as the training epoch increased and dominated the other losses. After some experiments we concluded that the best option in order to balance the relative effect of the difference losses is to scale the dissimilarity loss L_c by a factor of 0.1 and apply all losses from the first epoch. The relationship between losses in our experiments can be seen in section 3.6.3.

As mentioned in the previous subsection, a high number of classes K is important for the performance of the algorithm in our datasets. Unfortunately during early testing we noticed that the model would often stop using some of the classes after a few epochs of training. That is to say, in a batch of patches almost none or none of the pixels would be assigned maximum probability for some of the K classes. In order to combat this problem, instead of normal cross-entropy, we use a class-balanced version of the cross-entropy loss for L_p and $\hat{L}_p$. In each mini-batch we count the number of pixels that are assigned each class as the pseudo-label and weight the loss for each class by multiplying it with the reverse of the number of pixels assigned to that class. More

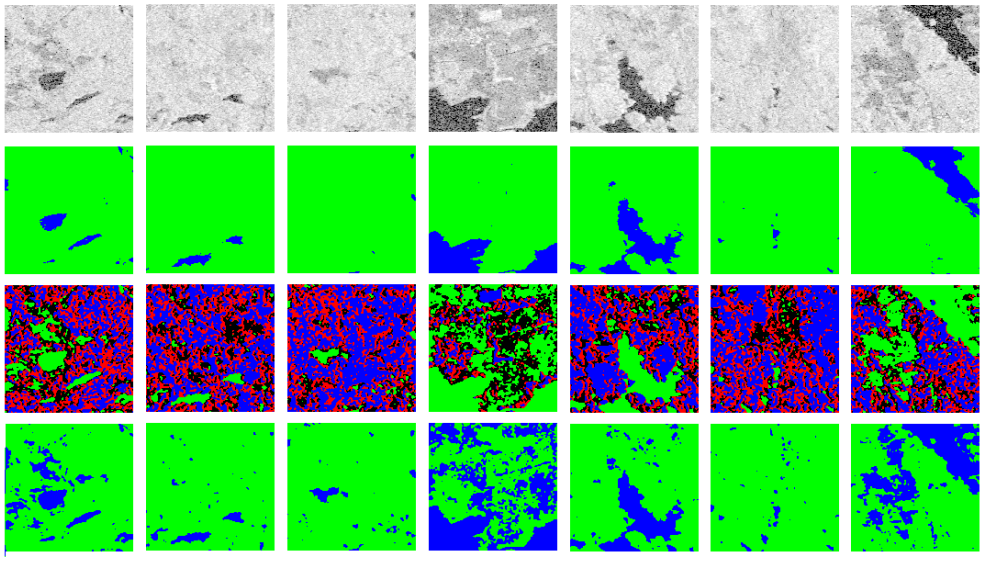


Figure 2.3: Examples of classes created by the algorithm in wetland SAR images for K=4. The first row shows the original SAR images, the second row shows shows the ground truth segmentation, the third row shows the model-produced classes and the fourth row shows the final class assignments. The class colors in the third row are arbitrary.

specifically, if $K_i, 0 \leq i < K$ denotes the number of pixels that are assigned the pseudo-label i in a training batch, then the class i is assigned a weight of $\frac{\frac{\varepsilon}{K_i+\varepsilon}}{\sum_{j=0}^{K-1}\frac{\varepsilon}{K_j+\varepsilon}}$ This results in classes that are assigned to less pixels being given more importance, which ensures that the model uses all of the classes during all epochs of training. A comparison on the number of classes used by the model and the effect that has on it's performance can be seen in 3.6.4.

2.3.4 Modifications on model architecture

One of the most significant changes we made was to change the architecture of the projection modules. We decided to explore the performance of the algorithm when replacing the architecture created by the authors of the paper with the U-Net model[14], since U-Net is one of the first CNN-based models designed specifically for the task of semantic segmentation and has been the basis for many other semantic-segmentation-focused model architectures. As with the previous model, U-Net contains groups of convolutional, activation and batch normalization layers. After two such groups a max pooling layer is inserted. This layer reduces each dimension of the image by half by replacing

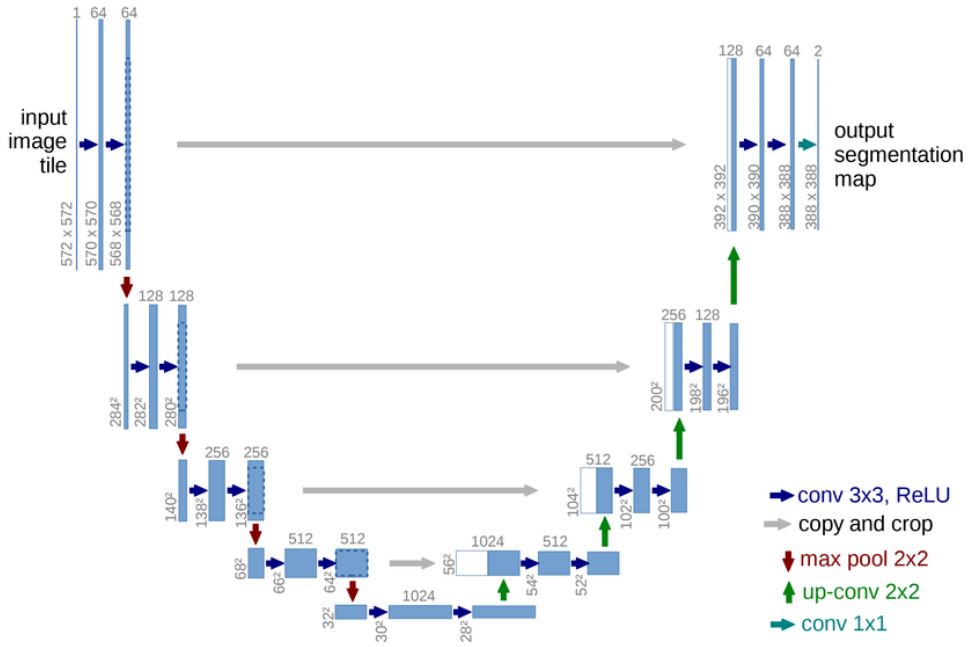


Figure 2.4: U-Net architecture (example for 32x32 pixels in the lowest resolution). Each blue box corresponds to a multi-channel feature map. The number of channels is denoted on top of the box. The x-y-size is provided at the lower left edge of the box. White boxes represent copied feature maps. The arrows denote the different operations. Figure from [14] used under CC BY 4.0 licence.

each square of four pixels in the original image with a single pixel with the maximum of the four values for each channel. This structure is repeated a number of times, with the convolutional layers in each depth having twice the number of channels as those in the previous depth in order to accommodate more complex representations. The repeating pooling layers have the effect that the later convolutional layers can take into account information from a large area of the input image while still using only a 3×3 size kernel, which requires few training parameters. Unfortunately this results in the resolution of features after many pooling layers being many times smaller than that of the original image. In order to generate pixel-level class assignments for the original image dimensions, the second half of the model upsamples the generated features to the correct scale. This second half is a mirror of the first one, with the downsampling layers replaced with up-convolutional (or transposed convolutional) layers that double the dimensions of their input. In order to help the upscaling process, at each depth the upscaled features are

concatenated with the same scale features that are produced by the first half of the U-Net at the same depth to help offset the loss of positional information that pooling layers cause. Again mirroring the first half of the model the number of channels of the convolutional layers are halved as the depth decreases. At the end of this process the model has produced features with the same resolution as that of the input image. In order to transform those features into class assignments, a convolutional layer with a 1×1 size kernel is used. An example of the U-Net architecture from the original paper can be seen in Figure 2.4. We modify slightly the original design by using using zero padding on the convolutional layers so that the output segmentation map has the exact same dimensions as the input image.

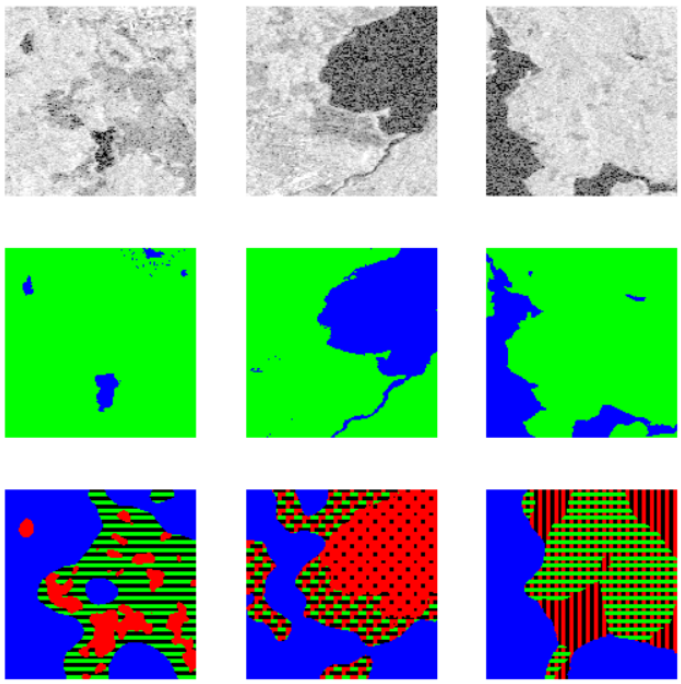


Figure 2.5: Patterns produced by U-Net with up-convolutional layers during our experiments. The top row shows the input SAR image, the middle row shows the ground truth labels and the bottom row shows the model's output. The colors in the bottom row represent arbitrary model classes.

During our initial experiments with U-Net we observed that the segmentations produced by our models very often produced linear or checkerboard patterns, as seen in Figure 2.5. These patterns often appear in models that use transposed convolutional layers[43]. In order to avoid this problem we replaced the transposed convolutional layers in U-Net with simple

upsampling layers, which solved the issue.

A comparison of the performance of the two architectures in our datasets can be seen in section 3.6.9.

2.3.5 Modifications on class matching algorithm

The original matching algorithm used corresponds at maximum one model-produced class to one ground truth class. Since we use a high number of model classes to segment each image patch into two ground truth classes (water and land), we need to be able to assign any number of model classes to a ground truth class. Moreover, as with many other remote sensing datasets, our datasets contain imbalanced classes, with land pixels being much more numerous than water pixels (details in section 3.1). Taking into account both of the above issues we simply assign each model class to the ground truth class it has the greatest Intersection over Union with. The new algorithm for assigning model classes to ground truth classes can be seen in Listing 2.3.

Listing 2.3: Modified algorithm for Matching self-supervised Segmented Map to the Reference Map

Input: Reference image and set of its constituent 2 classes $\{\mathcal{S}_1, \mathcal{S}_2\}$
Input: Obtained segmented image and set of its constituent K classes $\{\mathcal{T}_i\}_{i=1}^{K}$
for $k \leftarrow 1$ to K **do**
    Find $\mathcal{S}_{\mathbb{D}^k}$ as the class in $\mathcal{S}$ having highest intersection over union with $\mathcal{T}_k$
    Assign $\mathcal{T}_k$ as match for $\mathcal{S}_{\mathbb{D}^k}$
end for

2.3.6 Implementation of an ensemble model

Since our method uses no annotated data, we find that the performance of each model can vary significantly between runs, even when using hyperparameters are identical, due to the random nature of the classes that the model creates. We observe that the classes generated by our model are sometimes not suitable for the water segmentation task, since the classes that contain the water also contain significant areas of land, resulting in very poor performance. In order to overcome this problem, we use an ensemble of models. Since we have no annotated data, and therefore no way to assess our models' performance during

training, we opt for a basic bagging method where each model is independently trained with the full training dataset. Since the classes each model predicts do not have any correlation between the different models, we combine the model predictions after we transform them into water/land predictions by using per-pixel simple majority voting.

2.4 Discussion

Of the various available algorithms that perform self-supervised semantic segmentation, we chose the one presented in "Unsupervised Single-Scene Semantic Segmentation for Earth Observation" for a number of reasons. The most important one is that the algorithm is specifically designed for remote sensing datasets and takes advantage of the spatial properties of remote sensing images. Moreover, the method achieved state-of-the-art results in a number of different remote sensing datasets, suggesting that it can be effectively applied to a variety of remote sensing problems. This is further supported by the fact that the algorithm combines two different self-supervised training methods, namely deep clustering and negative sampling.

Despite those factors we found that we had to make significant changes to the algorithm in order for it to perform well for our datasets. Unlike the original paper we found out that we had to make the model output significantly more classes than the two in our datasets in order to produce useful segmentations for our purpose, which in turn necessitates a different class assignment algorithm. Moreover, the high variance of the model performance between different runs forced us to implement an ensemble version of the algorithm in order to obtain consistent results.

We should also mention that we attempted to use classic deep clustering by replacing the prediction module with a k-means clustering algorithm that clusters the representations of the pixels that the projection modules produce. However the large amount of computational power required to cluster so many data points forced us to abandon this avenue of exploration.

Chapter 3

Experiments

In this section we will present the experiments we used to gauge the performance of our algorithm on the task of water detection from satellite images, with an emphasis on detecting water under vegetation. We conduced a number of experiments to not only compare the performance of the self-supervised algorithm with that of other methods, but also to examine the effect different loss functions, number of model classes and model architectures have on the performance of the algorithm. Before presenting the experiments however we will present the datasets that the algorithm will be applied to, the algorithms that will be used as a baseline for the performance of our algorithm and the metrics that will be used to evaluate the performance of all methods.

3.1 Datasets

In order to test the ability of our model in the task of water segmentation from satellite radar images, we performed experiments with the following datasets:

3.1.1 Örebro SAR dataset

The Örebro SAR dataset[3] consists of SAR satellite images of wetlands in the county of Örebro in Sweden, along with corresponding surface water masks obtained by using the NDWI method on optical satellite images of the same area. The images were captured on the date 04/07/2018. These images have a pixel resolution of 10 meters. Some example images from the dataset can be seen in Figure 3.1. The percentage of water pixels in the dataset is 9.42%. After splitting the Örebro area into 639 tiles of 512x512 pixels, 80% of the tiles

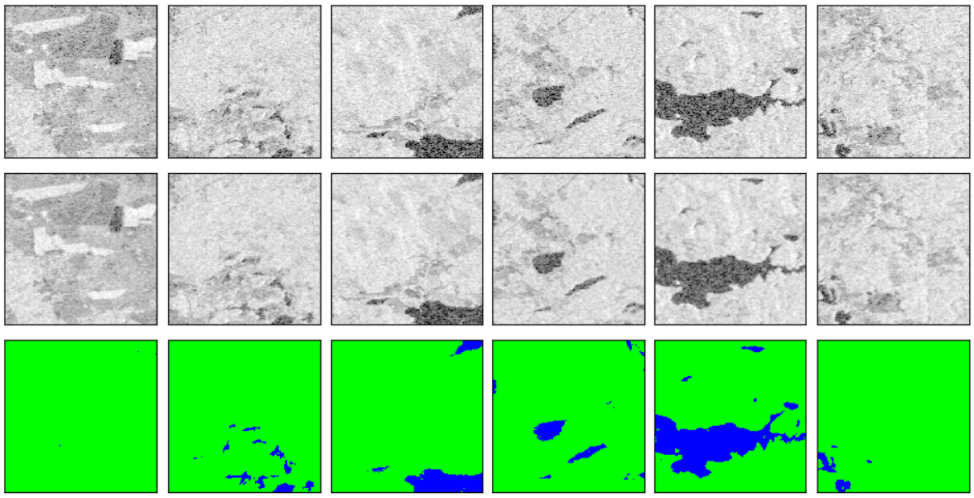


Figure 3.1: Examples of images from the Örebro SAR dataset. The top row shows the original SAR images, the middle row shows the SAR images augmented with Gaussian blur and the bottom row shows the ground truth images.

are randomly selected as the training dataset for our model and the remaining 20% are used as the validation dataset.

3.1.2 Swedish Wetlands SAR dataset

The Swedish Wetlands SAR dataset consists of 336 SAR satellite images of four different Swedish wetlands (Hjalstaviken, Hornborgarsjon, Mossatrask and Svartadalen) at different dates along with manually generated water segmentation masks of those areas. These images are captured at various dates from 12/10/2014 to 04/11/2022. In order to avoid images that contain a lot of snow, that can be difficult to differentiate from water in SAR images, we do not include any pictures taken during the months from December to March. These images have a pixel resolution of 10 meters. The percentage of water pixels in the dataset is 22.18%. Some example images from the dataset can be seen in Figure 3.2. This dataset was created by the author and two other KTH students. As with the Örebro SAR dataset, the images are split into 1203 tiles of 512x512 pixels that are used as the test dataset for our model.

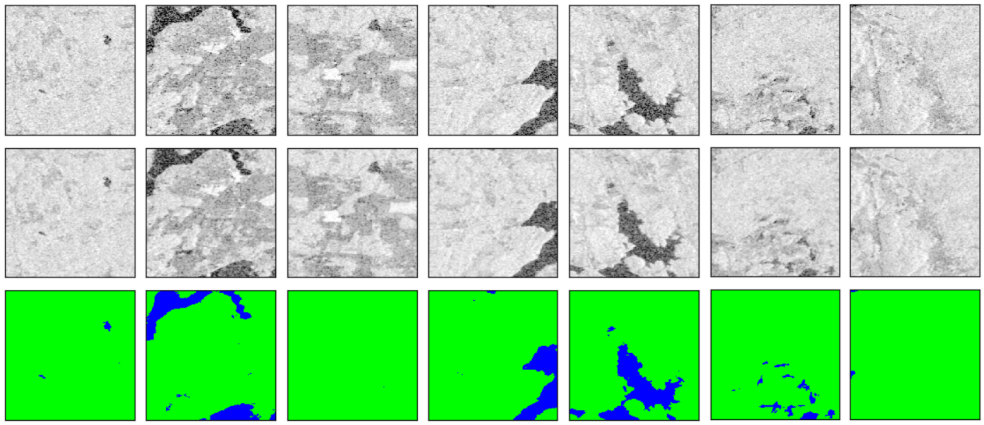


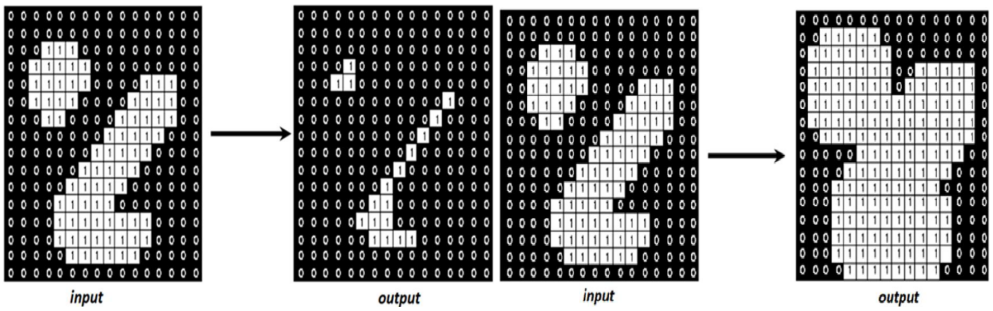
Figure 3.2: Examples of images from the Swedish Wetlands SAR dataset. The top row shows the original SAR images, the middle row shows the SAR images augmented with Gaussian blur and the bottom row shows the ground truth images.

3.2 Compared Methods

3.2.1 Otsu thresholding

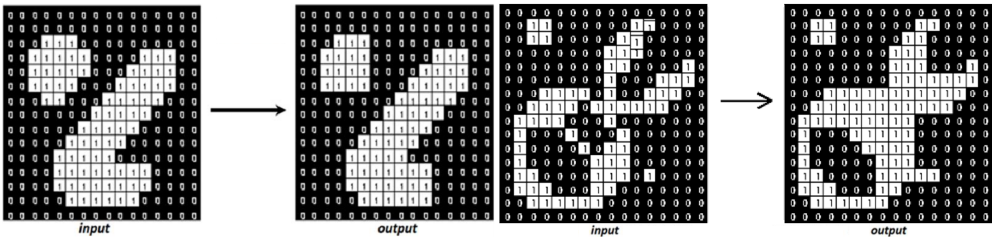
Otsu thresholding[44] is a non-parametric statistical method used to binarize a grayscale image by splitting the pixels into two classes based on whether their value is higher or lower than the threshold. The Otsu threshold is the one that minimizes the variance of the values in each of the two classes and maximizes the variance of the values between the two classes. Since water surface is usually much darker than ground surfaces in SAR images, the Otsu thresholding method has been used for semantic segmentation of water in satellite images[3]. We use Otsu thresholding as a baseline method that doesn't use any training data. In order to improve the output of this method, we first follow the example of the paper "DeepAqua: Semantic Segmentation of Wetland Water Surfaces with SAR Imagery using Deep Neural Networks without Manually Annotated Data"[3] and apply Gaussian blur to the images before using the Otsu method in order to reduce the impact of the noise inherent in SAR images on the method. The Gaussian blur is applied using a 5×5 size kernel. Unfortunately, since the Otsu method classifies each pixel independently from its neighbours, it tends to produce segmentations with lots of noise.

In order to improve the results of the Otsu method, we apply the morphological operations opening and closing to reduce noise from the



(a) Example of erosion in binary image. (b) Example of dilation in binary image.

Figure 3.3: Examples of erosion and dilation in binary images. Figures adapted from [45] used under CC BY-NC-SA 4.0 Deed licence.



(a) Example of opening in binary image. (b) Example of closing in binary image.

Figure 3.4: Examples of opening and closing in binary images. Figures adapted from [45] used under CC BY-NC-SA 4.0 Deed licence.

segmentation. The term morphological operations refers to image processing techniques that affect the shape of objects in an image[45]. Both the opening and closing operations are achieved by combining two more basic operations, namely erosion and dilation. The operation erosion, when applied to a binary image, replaces each pixel's value with the minimum of all pixel values in it's neighbourhood, shrinking shapes in an image. Similarly the dilation operation replaces each pixel's with the maximum of all pixel values in it's neighbourhood, expanding shapes in an image. Examples for those two operations can be seen in Figure 3.3. The opening operation is achieved by eroding an image and then dilating. This has the effect of smoothing contours and removing small objects and thin lines from the image. The closing operation is the opposite, a dilation followed by an erosion. This has the effect of smoothing contours and filling small holes in the image. Examples for those two operations can be seen in Figure 3.4. We apply 7 iterations each of opening and closing morphological operations using a 3×3 size kernel. The effect of these pre- and post-processing measures can be seen in section 3.6.1. The

parameters for pre- and post-processing were chosen based on their effect on the performance of the method on the training and validation datasets.

3.2.2 Supervised training

In order to compare the performance of the self-supervised algorithm to that of traditional supervised training, we train a U-Net model on the same data in a supervised manner. The U-Net model is the same as the projection module of the self-supervised training method with the addition of a convolutional layer with a 1x1 size kernel and a sigmoid activation layer at the end of the model. These layers transform the generated per-pixel features into one value between 0 and 1 for each pixel. Pixels with values greater than 0.5 are interpreted as water pixels and the rest as land pixels. This model is trained using the Dice loss function[46] which has been found to perform well for binary segmentation problems[47]. Dice loss is defined as

$$L_{dice} = 1 - \frac{2 \cdot \sum_i p_i \cdot g_i + \epsilon}{\sum_i p_i + \sum_i g_i + \epsilon}$$

In the above equation p_i and g_i represent the predicted and the ground truth pixels respectively, $\sum_i$ represents a summation of all pixels in the image and ϵ is a small positive constant. The value of each of the pixels p_i and g_i is 1 for the class of interest (in our case water) and 0 for the background (in our case land). All other parameters of the model architecture and training algorithm are identical to those of the self-supervised model. In order to determine the effect of the number of training data to the performance of the model, we repeated our experiments using 100%, 50% and 10% of the training dataset.

3.3 Evaluation metrics

In order to evaluate the performance of all methods in the task of semantic segmentation we use two different evaluation metrics: per-pixel-accuracy and Intersection Over Union. Per-pixel-accuracy is simply the number of pixels that have been assigned the correct class (in our case, water or land) divided by the total number of pixels. While this is a simple and extensively used metric, it is not suitable for datasets that have a significant class imbalances, which is quite common in land cover datasets. As an example, in a dataset that contained significantly more land pixels than water pixels, a model could achieve a high per-pixel-accuracy score by simply assigning each pixel to the land class. In order to avoid this problem we also use the IOU evaluation

metric. This metric is very often used in semantic segmentation tasks as it maps the overlap between the predicted area of the class of interest (in our case water) and the actual area. More specifically, the formula for IOU is:

$$IOU = \frac{W_{pred} \cap W_{gt} + \epsilon}{W_{pred} \cup W_{gt} + \epsilon}$$

where W_{pred} is the area classified as water by our model, W_{gt} is the ground truth water area and ϵ is a small constant. We compute IOU over all the images in each batch and take the average of the batch IOUs as the dataset IOU.

3.4 Model architecture configuration

Considering the fact that our input images have only one channel and our final goal is to distinguish between 2 different classes, water and land, we opt to use a U-Net model with a small depth and a small number of channels. After some initial experiments we settled on an architecture of depth 2, meaning it has 2 max-pooling layers and 2 up-sampling layers. We choose to use convolutional layers with only 8 channels at the top layer, meaning that the bottom layer has convolutional layers with 32 channels. Some experiments on the effect of those parameters in the performance of the algorithm can be seen in sections 3.6.7 and 3.6.8. We initialize the weights of our models using He initialization, following the example of the original paper.

3.5 Training configuration

The training was performed in the language python using the PyTorch machine learning library. The model was trained on an computer node provided by Stockholm University in the Alvis cluster and on a NVIDIA T4 GPU. We train the model for 100 epochs using the AdamW[48] optimizer algorithm with a learning rate of $lr = 0.001$, coefficient for gradient moving average $\beta_1 = 0.9$, coefficient for gradient squared moving average $\beta_2 = 0.999$, small constant $\epsilon = 10^{-8}$ and a weight decay of $w = 0.01$. During training the learning rate changes every epoch according to the cosine annealing with warm restarts algorithm[49] with parameters of minimum learning rate $\eta_{min} = 0$, number of iterations for the first restart T_0 equal to the number of training batches and factor multiplied with iterations number at each restart $T_{mult} = 2$. The augmented image patches $\hat{\mathcal{X}}$ are created by applying a Gaussian filter with a 5×5 size kernel and a standard deviation randomly sampled uniformly

from between the values of 1 and 2. Moreover, we apply further augmentation on all images by randomly flipping them horizontally and vertically with a probability each of 0.5. After training, the model's weights at the final epoch are used to obtain the final model segmentations. For all our experiments we perform 10 runs with each configuration and present the mean and variance of the resulting metrics.

3.6 Results

3.6.1 Otsu thresholding performance

Table 3.1: Otsu thresholding performance

Pre-processing	Post-processing	Val IOU	Val acc.	Test IOU	Test acc.
No	No	0.57	0.954	0.456	0.818
Yes	No	0.652	0.966	0.484	0.833
Yes	Yes	0.827	0.988	0.594	0.897

In Table 3.1 and Figure 3.5 we can see the performance of the Otsu thresholding algorithm with and without pre- and post-processing. We can see that both the pre-processing (Gaussian blurring) and the post-processing (Opening and Closing morphological operations) result in less noise in the resulting segmentation. On the other hand, the post-processing also results in significantly cruder borders between the segmented regions. Despite that fact, using both pre- and post-processing outperforms the other methods. We can see that the differences in performance between the different methods is much more pronounced for the IOU metric than for the per-pixel accuracy. This is due to the fact that most of the pixels in the validation and test datasets are land. This shows that IOU is a more appropriate metric for our datasets than simple accuracy. For this reason we will focus on the IOU metric in future experiments. We also see that the algorithm performs worse on the test dataset, which might be because the parameters for pre- and post-processing were fine-tuned for the training and validation datasets.

3.6.2 Supervised training performance

We performed supervised training on portions of the Örebro SAR dataset as a second baseline method to compare the performance of the self-supervised

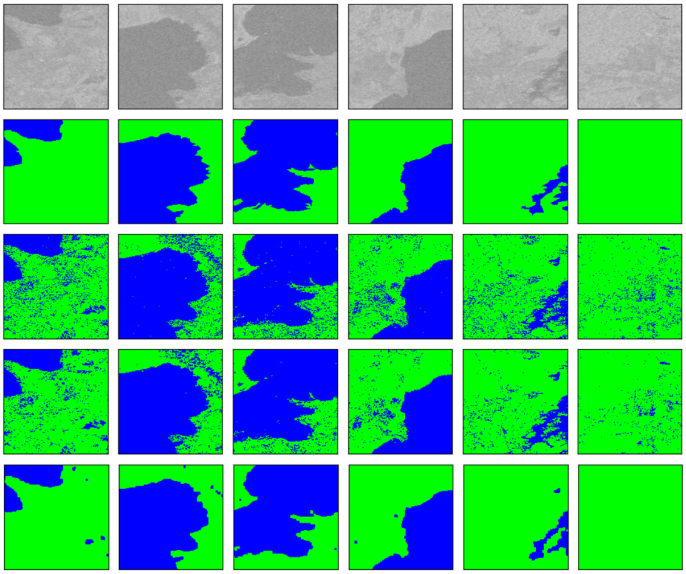


Figure 3.5: Examples of semantic segmentation of water from land using the Otsu thresholding algorithm on our validation dataset. The top row shows the input SAR image, the second row shows the ground truth labels, the third row shows the segmentation of the Otsu algorithm without pre- and post-processing, the fourth row shows the segmentation of the Otsu algorithm with Gaussian blurring as pre-processing and the fifth row shows the segmentation of the Otsu algorithm with Gaussian blurring as pre-processing and the morphological operations Opening and Closing as post-processing.

Table 3.2: Supervised model performance

Perc. training data	Max val IOU mean±var	Test IOU mean±var
1.0	$0.885 \pm 3.9 \times 10^{-6}$	$0.642 \pm 3.4 \times 10^{-3}$
0.5	$0.879 \pm 1.4 \times 10^{-5}$	$0.566 \pm 1.0 \times 10^{-2}$
0.1	$0.87 \pm 4.3 \times 10^{-5}$	$0.583 \pm 7.9 \times 10^{-3}$

training method to. We use the same training-validation split an in the self-supervised training algorithm, except that only a fraction of the training data is used. We performed 10 runs each on 100%, 50% and 10% of the training dataset. The models' performance on the test dataset are shown in Table 3.2. As expected, the supervised model tends to perform better when trained with more data, although the performance for 50% and 10% of the training dataset is very similar. Comparing the results with those of the Otsu method, we can see that while the supervised model performs better when trained on

the whole dataset, in other cases the Otsu thresholding method with pre- and post-processing actually outperforms the machine learning model on the test dataset.

3.6.3 Base self-supervised model performance

Table 3.3: Self-supervised model performance

Val final IOU mean±var	Val final acc. mean±var	Test IOU mean±var	Test acc. mean±var
$0.78 \pm 3.2 \times 10^{-3}$	$0.98 \pm 7.1 \times 10^{-5}$	$0.533 \pm 4.0 \times 10^{-2}$	$0.878 \pm 1.8 \times 10^{-3}$

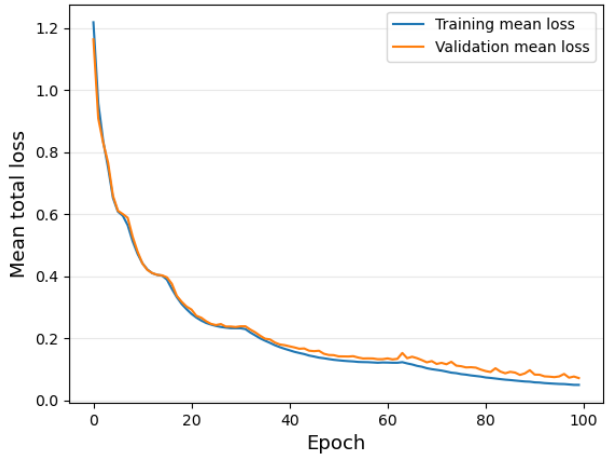
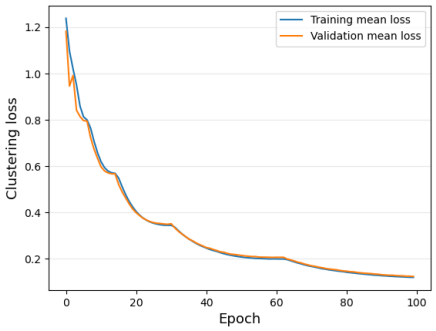
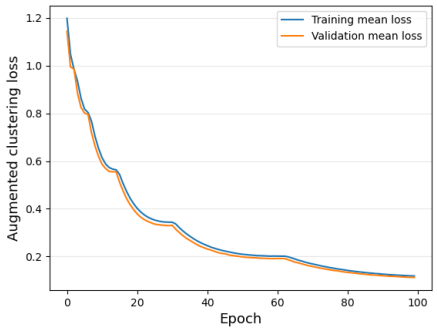


Figure 3.6: Self-supervised model mean training and validation total loss.

We performed 10 runs of the self-supervised training method using the training parameters described in 3.5. The mean training and validation losses for the runs can be seen in Figures 3.6, 3.7 and 3.8. We observe that all losses except $\mathcal{L}_s$ are almost identical for the training and validation dataset, while validation $\mathcal{L}_s$ seems to diverge from it's training counterpart as the training epoch increases. Moreover, while $\mathcal{L}_p$, $\hat{\mathcal{L}}_p$ and $\mathcal{L}_s$ seem to plateau after a point, the value of the dissimilarity loss $\mathcal{L}_c$ continues to decrease at a steady pace. This could be a problem since during longer training epochs $\mathcal{L}_c$ might come to dominate the other losses. However for the 100 epochs that we train the model the scaling factor of 0.1 that we use for $\mathcal{L}_c$ does not allow it to overly impact the total loss. The validation and test metrics for the runs are shown in Table

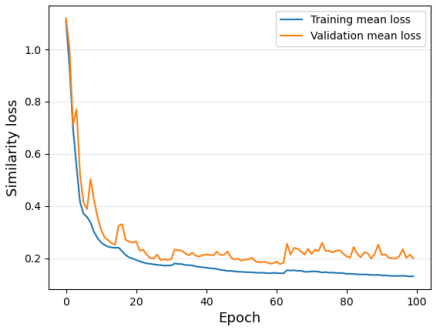


(a) Self-supervised model mean training and validation clustering loss.

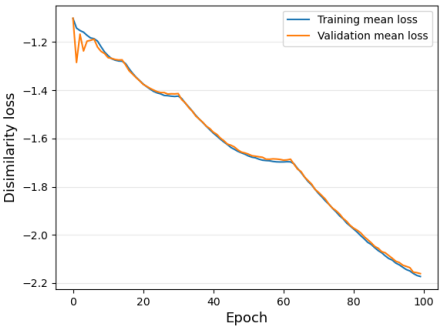


(b) Self-supervised model mean training and validation augmented clustering loss.

Figure 3.7: Self-supervised model mean training and validation deep clustering losses.



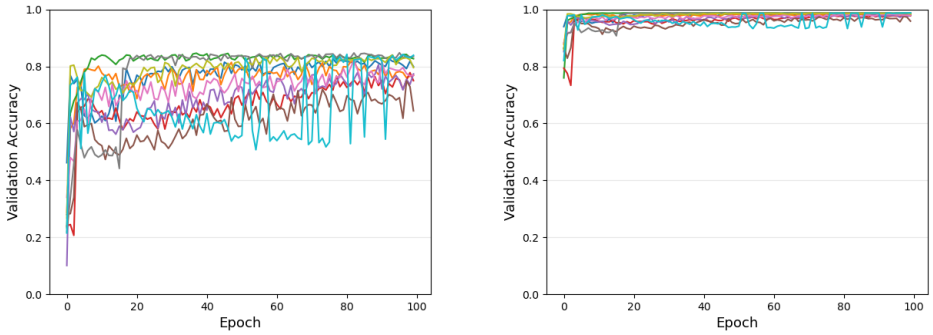
(a) Self-supervised model mean training and validation similarity loss.



(b) Self-supervised model mean training and validation dissimilarity loss.

Figure 3.8: Self-supervised model mean training and validation negative sampling losses.

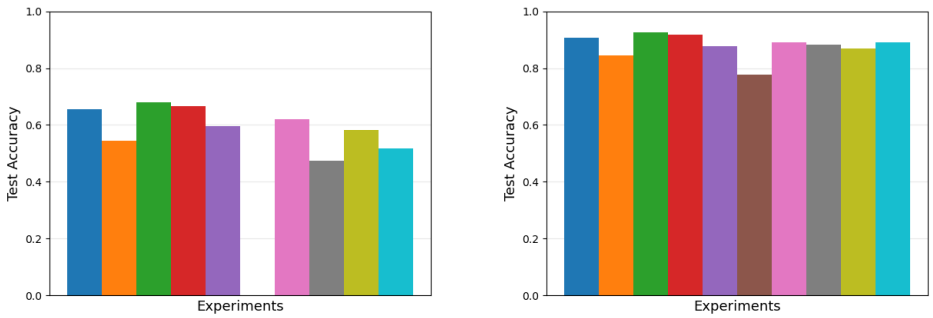
3.3 and in Figures 3.9 and 3.10. We can see that while the IOU and accuracy metrics for each model appear to be heavily correlated, the IOU values have a much greater range in comparison to the accuracy values. Moreover, we can see that the model that achieves the lowest final validation IOU achieves a test IOU of 0, while still having a test accuracy of almost 80%. This is due to the fact that the segmentation classes produced by that model on the test dataset all have a higher IOU with the land ground truth class than the water one, which results in all of the model classes being assigned the land label. Figure 3.9a shows that there is a wide variance between different runs



(a) Self-supervised model validation IOUs.

(b) Self-supervised model validation accuracies.

Figure 3.9: Self-supervised model validation metrics. Each color represent a different run.



(a) Self-supervised model test IOUs.

(b) Self-supervised model test accuracies.

Figure 3.10: Self-supervised model test metrics. Each color represent a different run.

in the number of epochs that the model performance needs to plateau and in the final validation IOU. Moreover we can see that some models show large variation in the validation IOU between consecutive epochs. This happens when a model class that was previously assigned the water label is in the next epoch assigned the land label (or vice versa). Comparing the results with those of the supervised model, we can see that the supervised model outperforms the self-supervised one even when using only 10% of the training data. Moreover, the Otsu thresholding method performs better than the self-supervised model. The high variance we observed in the performance of the model lead us to implement the ensemble version of this algorithm.

3.6.4 Effect of weighted cluster loss

Table 3.4: Weighted clustering loss performance

Clustering loss type	Val final IOU mean±var	Test IOU mean±var
Uniform	$0.696 \pm 1.9 \times 10^{-2}$	$0.438 \pm 6. \times 10^{-2}$
Weighted	$0.78 \pm 3.2 \times 10^{-3}$	$0.533 \pm 4.0 \times 10^{-2}$

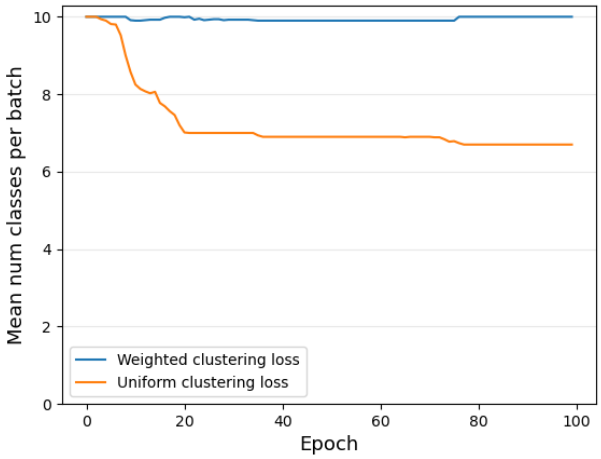
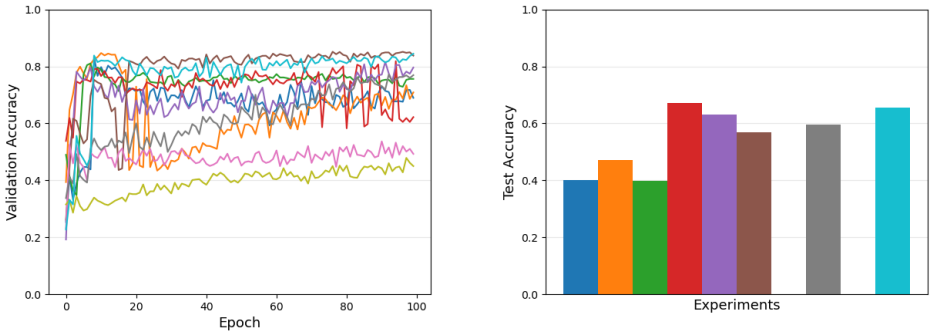


Figure 3.11: Mean number of different classes per batch.



(a) Validation IOUs.

(b) Test IOUs.

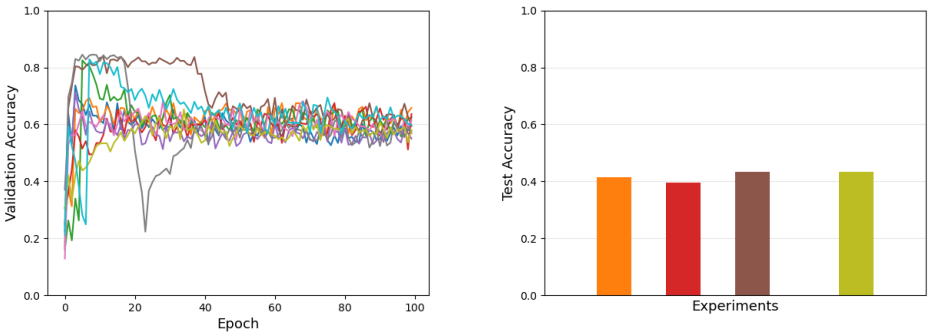
Figure 3.12: Uniform weight loss model validation and test IOUs. Each color represent a different run.

In this ablation experiment we examined the effect of weighting the clustering loss for each class inversely proportionally to the number of pixels predicted to be that class in each batch by training 10 models using classic Cross-entropy instead of weighted Cross-entropy for the clustering losses. Looking at Figure 3.11 we can see that while models that use weighted loss assign all or almost all possible classes in each batch during all epochs, models that don't use weighted loss seem to assign fewer unique classes as the epoch number increases. As we can see in Table 3.4 and Figure 3.12 not using weighted loss results in a significant increase in the variance of the models' performance and a big reduction in the mean test IOU, indicating that maintaining a sufficiently high number of classes during training is important for the performance of the algorithm.

3.6.5 Effect of number of classes

Table 3.5: Effect of K in model performance

K	Val final IOU mean±var	Test IOU mean±var
6	$0.6 \pm 1.1 \times 10^{-3}$	$0.167 \pm 4.7 \times 10^{-2}$
10	$0.78 \pm 3.2 \times 10^{-3}$	$0.533 \pm 4.0 \times 10^{-2}$
14	$0.655 \pm 6.7 \times 10^{-3}$	$0.255 \pm 1.5 \times 10^{-2}$

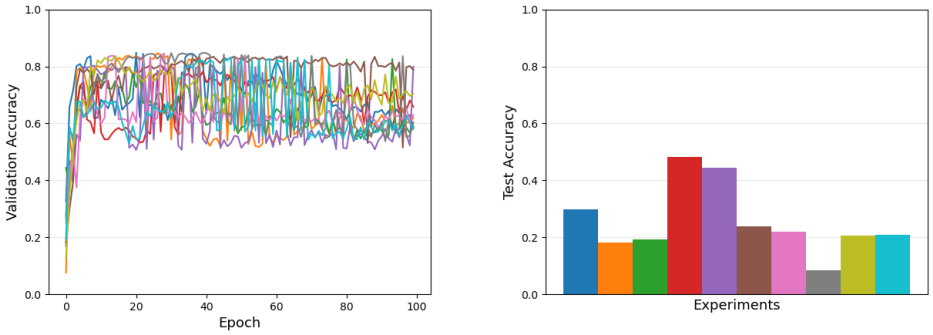


(a) Validation IOUs.

(b) Test IOUs.

Figure 3.13: Self-supervised model validation and test IOUs for number of classes K=6. Each color represent a different run.

We also performed experiments to determine the effect that using different amounts of classes K has to the performance of the algorithm. More



(a) Validation IOUs.

(b) Test IOUs.

Figure 3.14: Self-supervised model validation and test IOUs for number of classes K=14. Each color represent a different run.

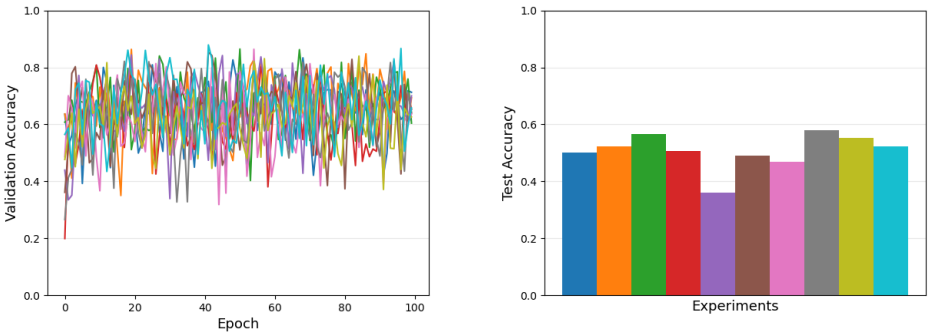
specifically, we performed 10 runs each with $K = 6$ and $K = 14$ and compare the results with those of the basic training setup of $K = 10$. The results of those runs can be seen in Table 3.5 and in Figures 3.13 and 3.14. We see that, when using only 6 classes, a few of the models briefly achieve a performance on the validation dataset equal to that of the models using 10 classes. However all models eventually converge at a significantly lower validation IOU. Moreover most of the models have a test IOU of zero, suggesting that all the model classes have been assigned to land, which leads us to believe that 6 classes are not enough to produce segmentations that are conductive to the task of water detection. On the other hand, when using 14 classes all of the models achieve a maximum validation IOU similar to that of the models using 10 classes, but there is significantly more variation in their performance for different epochs of training. These sudden changes in validation IOU mean that one or more of the model classes often shift from having more overlap with water to land or vice versa. This again suggests that the segmentations produced by the model are not a good fit for the task of water detection. From these experiments we conclude that choosing the correct number of classes depending on the task at hand is very important fo the performance of this algorithm.

3.6.6 Effect of batch size

In order to determine the effect different batch sizes have on the performance of the model we performed 10 runs each using batch sizes 2 and 8. The validation and test IOUs are still computed in batches of size 16 so that the results can

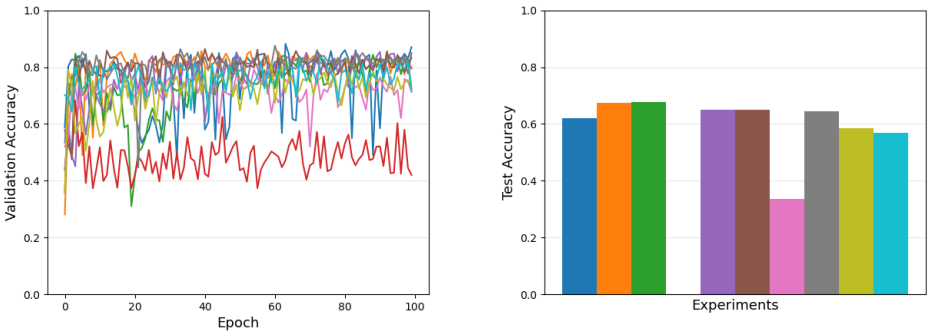
Table 3.6: Effect of batch size in model performance

Batch size	Val final IOU mean±var	Test IOU mean±var
2	$0.656 \pm 1.2 \times 10^{-3}$	$0.506 \pm 3.9 \times 10^{-3}$
8	$0.744 \pm 1.6 \times 10^{-2}$	$0.54 \pm 4.6 \times 10^{-2}$
16	$0.78 \pm 3.2 \times 10^{-3}$	$0.533 \pm 4.0 \times 10^{-2}$



(a) Validation IOUs. (b) Test IOUs.

Figure 3.15: Self-supervised model validation and test IOUs for batch size 2. Each color represent a different run.



(a) Validation IOUs. (b) Test IOUs.

Figure 3.16: Self-supervised model validation and test IOUs for batch size 8. Each color represent a different run.

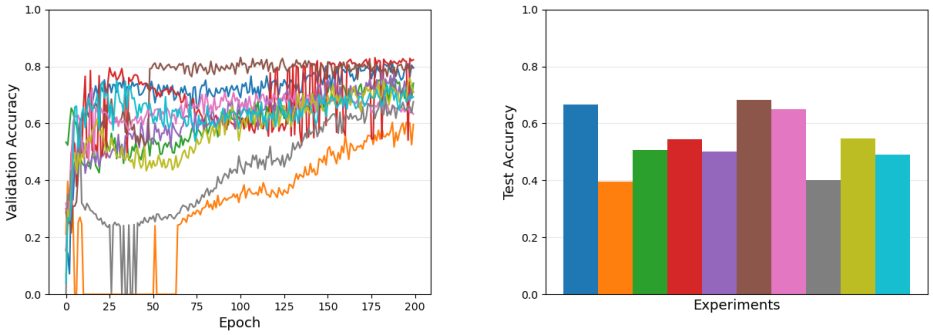
be compared with our other experiments. The results can be seen in Table 3.6 and in Figures 3.15 and 3.16. We observe that for a batch size of 2 there is a lot of variance in the validation IOU between different epochs, and the test IOUs of the models are on average slightly lower than those of our base model. On

the other hand for a batch size of 8 the model seems to perform similarly to the base model, suggesting that a batch size of 8 is sufficient for the performance of the algorithm in these datasets.

3.6.7 Effect of number of channels

Table 3.7: Effect of number of channels in model performance

Number of channels in first layer	Batch size	Val final IOU mean±var	Test IOU mean±var
4	16	$0.717 \pm 5.4 \times 10^{-3}$	$0.538 \pm 1.0 \times 10^{-2}$
8	16	$0.78 \pm 3.2 \times 10^{-3}$	$0.533 \pm 4.0 \times 10^{-2}$
16	8	$0.741 \pm 1.9 \times 10^{-2}$	$0.522 \pm 3.3 \times 10^{-2}$

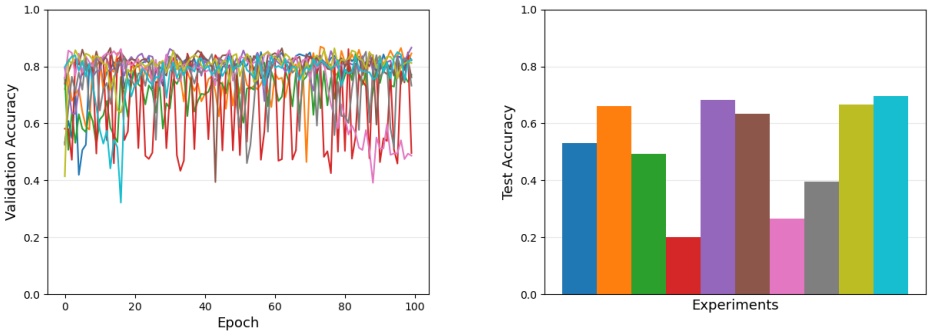


(a) Validation IOUs.

(b) Test IOUs.

Figure 3.17: Self-supervised model validation and test IOUs for 4 channels in model's first layer. Each color represent a different run.

In order to determine the effect the number of channels in the model's convolutional layers has on it's performance, we performed 10 runs each using models with 4 and 16 channels in the first convolutional layers (which are doubled each time the depth increases). Due to the larger size of the model we were forced to halve the batch size for the experiments with the models using 16 channels in the first layer. The results can be seen in Table 3.7 and in Figures 3.17 and 3.18. We can see that all models achieve similar mean test IOUs to those of the baseline model. However the models with 16 channels in the first layer show large variation in validation IOU between different epochs. On the other hand, the models with 4 channels seem to require on average



(a) Validation IOUs.

(b) Test IOUs.

Figure 3.18: Self-supervised model validation and test IOUs for 16 channels in model's first layer. Each color represent a different run.

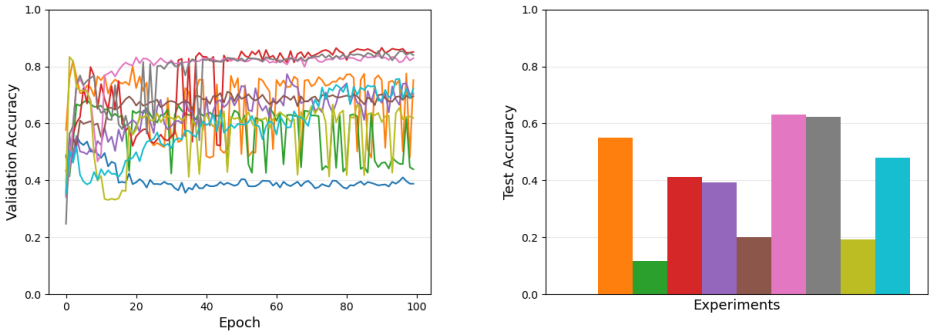
more epochs to reach the plateau of their performance. This suggests that in some cases longer training can compensate for smaller feature size.

3.6.8 Effect of model depth

Table 3.8: Effect of model depth in model performance

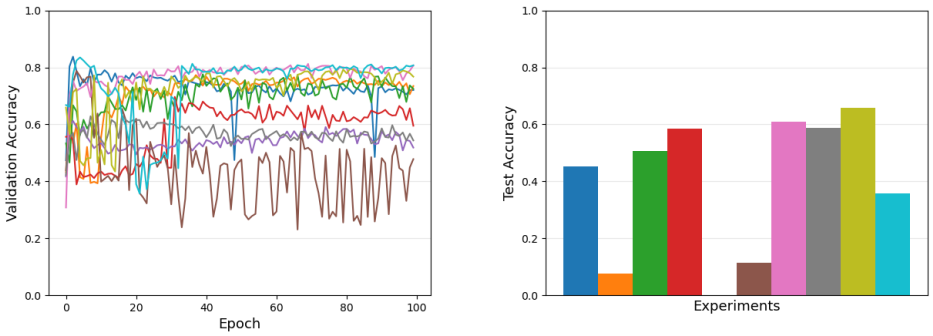
Model depth	Val final IOU mean±var	Test IOU mean±var
2	$0.78 \pm 3.2 \times 10^{-3}$	$0.533 \pm 4.0 \times 10^{-2}$
3	$0.683 \pm 2.6 \times 10^{-2}$	$0.36 \pm 4.9 \times 10^{-2}$
4	$0.669 \pm 1.5 \times 10^{-2}$	$0.394 \pm 6.0 \times 10^{-2}$

In order to determine the effect the model's depth has on it's performance, we performed 10 runs each using models with depth 3 and 4. The results can be seen in Table 3.8 and in Figures 3.19 and 3.20. Curiously, we observe that increasing model depth results in a much bigger variance in the validation IOU value that each model plateaus at. While the top of that range is similar to that of the baseline model, the bottom is much lower, resulting in significantly lower test IOUs. These different plateaus suggest that deeper models can produce a wider variety of segmentations, although they don't seem to be conductive to our task. Moreover, we observe that using a model with depth 3 or 4 results in similar performance.



(a) Validation IOUs. (b) Test IOUs.

Figure 3.19: Self-supervised model validation and test IOUs for model depth 3. Each color represent a different run.



(a) Validation IOUs. (b) Test IOUs.

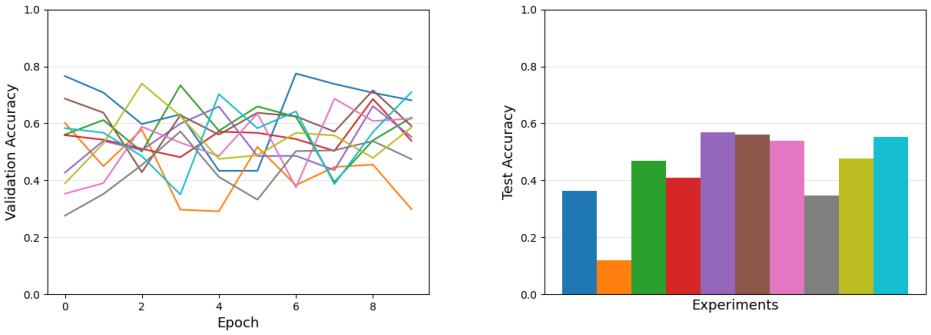
Figure 3.20: Self-supervised model validation and test IOUs for model depth 4. Each color represent a different run.

3.6.9 Effect of model architecture

Table 3.9: Model architecture performance

Model	Num. parameters	Batch size	Val. final IOU mean±var	Test IOU mean±var
Original	889630	2	$0.566 \pm 1.3 \times 10^{-2}$	$0.44 \pm 1.9 \times 10^{-2}$
U-Net	13310	2	$0.656 \pm 1.2 \times 10^{-3}$	$0.506 \pm 3.9 \times 10^{-3}$

In order to examine the effect the model architecture has on performance, we compare the results of our U-Net-based model with the results of the original model from [2]. As seen in Table 3.9, our model contains much fewer



(a) Validation IOUs.

(b) Test IOUs.

Figure 3.21: Self-supervised model validation and test IOUs for original model architecture. Each color represent a different run.

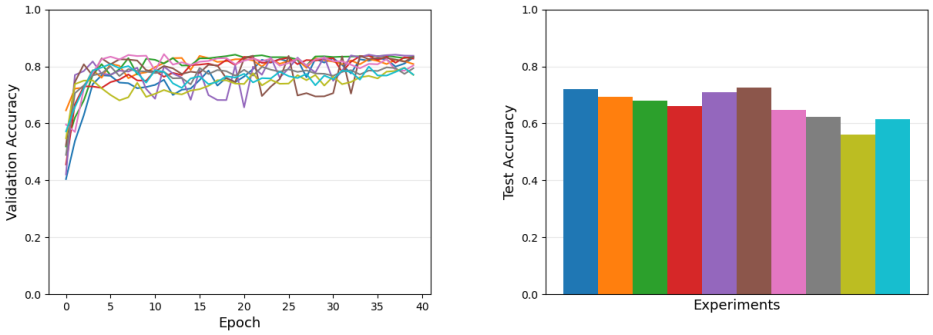
trainable parameters than the original one due to it's small depth, low number of channels and its use of upsampling layers in place of reverse convolutional ones. The large size of the original model restricts us to a maximum batch size of 2 when performing this experiment. We can see in Figure 3.21 and in Table 3.9 that the original model achieves significantly lower validation and test IOU results than U-Net. Moreover, while it requires very few epochs to reach it's plateau, it also shows a very high variance between epochs.

3.6.10 Comparison with ensemble model

Table 3.10: Ensemble self-supervised model performance

Max val IOU mean±var	Test IOU mean±var
$0.81 \pm 6.7 \times 10^{-4}$	$0.662 \pm 2.8 \times 10^{-3}$

As we mentioned before, the high variance in the performance of our models lead us to implement an ensemble model to achieve more stable results. After some experiments we determined that an ensemble of 5 models achieves the best balance between performance and computational resources needed. We performed 10 runs with the ensemble model and the results are shown in Table 3.10 and in Figure 3.22. As we had hoped, the ensemble model has much more stable performance that the individual models and achieves a consistently high validation IOU unlike any of the models in our previous experiments. Moreover, the ensemble model achieves a significantly higher



(a) Validation IOUs.

(b) Test IOUs.

Figure 3.22: Ensemble self-supervised model validation and test IOUs. Each color represent a different run.

mean test IOU of 0.662, compared to the mean test IOU of 0.533 of the base model and outperforming both the Otsu segmentation method and the supervised supervised models, even when they use the whole training dataset.

3.7 Discussion

In this section we will discuss the results of our experiments. The first observation we can make is that all of our self-supervised models, with the exception of the ensemble model, demonstrate considerable variance in their performance between different runs. We think that this is mainly due to the high degree of randomness that is inherent when using as pseudo-labels the labels produced by a completely untrained model, as is the case in the first epoch of the training algorithm. This initial randomness can have a very large impact to the rest of the training, although in most of our experiments all models tended to converge toward similar validation performance towards later iterations. Many models also present high variance in validation performance for different epochs, caused by model classes often being switched from land to water or vice versa. This leads us to conclude that often some of the classes created by the models are not conductive to our task of detecting water. We were able to bypass these problems by using an ensemble of 5 self-supervised models, although of course this increases the cost of training fivefold. Perhaps more concerning is the fact that changes in model depth, number of classes or model architecture result in a significant decrease in the performance of the algorithm. This need for

extensive hyperparameter search suggest that this training method and model might not transfer well to other datasets or tasks. Despite the above, it is noteworthy that our ensemble model outperforms the fully supervised one, proving that detecting water from SAR satellite images without the use of annotated data is possible.

Chapter 4

Conclusions and Future work

In this project we explored the application of self-supervised machine learning methods in the task of detecting water from satellite radar images.

4.1 Conclusions

We based our approach on the training method proposed by the paper ”Unsupervised Single-Scene Semantic Segmentation for Earth Observation” but we modified it in three key ways. Firstly, we replaced their convolutional model with U-Net, one of the most successful deep networks in the field of semantic segmentation. The U-Net model we used performed better that the model used in the original paper while having over 66 times less trainable parameters, meaning it requires significantly fewer resources and time to train. Secondly, we weighted the clustering losses by the inverse of each classes frequency to encourage the model to use all possible classes. This change forced the model to use all the clustering pseudo-classes, while the original training algorithm tended to consolidate different classes together after a few epochs. While those changes improved the performance of the model on our datasets, we observed the performance of a single model can vary a lot based on the random model weight initializations and the shuffling of the training data. This is due to the fact that the model uses as a pseudo-label for each pixel the label it gives maximum probability, which during the first epoch is very dependant on the above two factors. Moreover, there was significant variance between different epochs within runs as some pseudo-classes switched from being assigned from land to water and vice-versa. In order to solve this problem we used an ensemble of 5 models, each trained independently of each other and aggregated their results by a simple vote of land or water

for each pixel. The ensemble model not only significantly reduced the per-run and per-epoch variance of our model, but also achieved a better overall performance. Our work shows that self-supervised machine learning models can be used to detect water from satellite SAR images, with our ensemble model outperforming the Otsu thresholding method.

4.2 Limitations

This project highlights the potential of self-supervised machine learning methods but also some of their limitations. More specifically, we observed that the performance of the model is very sensitive to different model architectures and hyperparameters, meaning that in order to achieve good results extensive hyperparameter search is required. Moreover, even when using the same hyperparameters, the high variance resulting from the deep clustering training method requires an ensemble of models to achieve stable results. These limitations work against the main benefit of self-supervised learning, which is the lack of need for expensive and time-consuming data annotation.

Another limitation of our work is the somewhat limited number of runs that were performed for each experiment. Due to the large number of different training methods, models and hyperparameters that were tested, we were only able to perform 10 runs for each of the different configurations. Considering the high performance variance that we observed in many of those experiments, 10 is probably an insufficient number of runs for us to be very confident in our observations.

4.3 Future work

There are several avenues worth exploring in order to improve our method for detecting water from satellite SAR images using self-supervised learning. One of the simplest ones would be applying our method to wetlands from different areas. All the wetlands from our dataset are located in the Nordics, and since wetlands from different areas of the world show high heterogeneity, applying our method to wetlands from different regions would provide very important information regarding the generalization abilities of our model.

Since the training algorithm we use combines a deep clustering loss with a contrastive loss, a natural extension of our work would be to try including more self-supervised losses in the training algorithm. One such possible extension would be to add a second upsampling hand to our U-Net model, parallel to the

original one, whose output is trained to mirror the input image. In this way, a generative loss could be added to the model.

Another direction for future research would be trying to use a different model architecture. In the last few years a variety of models specifically focused on semantic segmentation of remote sensing images have been designed, such as ResUnet[50], which adds residual connections to the U-Net architecture, or the attention-incorporating MA-UNet[51]. We believe that using models specifically designed for remote sensing images could yield significant improvements.

One final direction for future work would be to try applying different methods to reduce the variance in the performance of our model. Since using an ensemble of models significantly improved the stability and performance of our model, we believe that exploring more complex ensembling methods is a promising avenue.

4.4 Ethics and Sustainability

In this section we will discuss the environmental and ethical implications of our work. Our work seeks to improve our ability to monitor water hidden under vegetation, a task that is vital for the monitoring of wetlands from satellite images. Since wetlands provide a wide variety of services to nearby areas and contain a large amount of biodiversity, any improvement in our ability to monitor them has the potential for significant environmental impact in the surrounding area. Moreover, since wetlands contain a large amount of greenhouse gasses, their monitoring is incredibly important for the fight against global warming. Finally, we believe that our work also contributes to the task of ecologically sustainable development, as it provides scientists and other decision makers with the information needed to act in a sustainable way when considering development in and near wetlands. In opposition to the above factors, the training and testing of machine learning models can consume large amount of energy, with the energy requirements increasing rapidly as the size of models and datasets balloons[52]. In order to reduce the energy consumption of our model we used a model architecture with 66 times less trainable parameters than the original paper's architecture, thereby massively reducing the energy required for training and testing it. For the above reasons we conclude that our work has a significant positive environmental impact.

One of the biggest ethical considerations regarding machine learning is it's impact in the job market. In the last few years the rapid advancements

in areas of machine learning such as computer vision and natural language processing have resulted in many tasks, which were previously performed by humans, being automated. Our work specifically could be seen as endangering jobs in two ways. The first is that it seeks to automate the task of remote water monitoring, while the second is by seeking to reduce the need for manual annotation of data through the use of self-supervised learning. We believe that the importance of the task as described above, along with the immense difficulty that manual annotation and monitoring would entail, make the automation of water monitoring necessary.

Finally, the recent advances in computer vision have created significant concerns about the effect the technology can have in surveillance and other privacy-invading applications. However, since our model uses satellite images and is trained to detect water bodies, we don't believe that our work contributes to these problems in any way.

References

[1] 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 1.]

[2] S. Saha, M. Shahzad, L. Mou, Q. Song, and X. X. Zhu, "Unsupervised single-scene semantic segmentation for earth observation," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 60, pp. 1–11, 2022. doi: 10.1109/TGRS.2022.3174651 [Pages 1, 17, 18, 19, 20, 21, and 48.]

[3] 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/S156984322300448X [Pages 2, 3, 12, 31, and 33.]

[4] N. C. Davidson, "How much wetland has the world lost? long-term and recent trends in global wetland area," *Marine and Freshwater Research*, vol. 65, no. 10, pp. 934–941, 2014. [Page 2.]

[5] P. Goyal, M. Caron, B. Lefaudeux, M. Xu, P. Wang, V. Pai, M. Singh, V. Liptchinsky, I. Misra, A. Joulin, and P. Bojanowski, "Self-supervised pretraining of visual features in the wild," 2021. [Page 3.]

[6] 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. [Page 4.]

[7] G. Kaplan and U. Avdan, "Mapping and monitoring wetlands using sentinel-2 satellite imagery," *ISPRS Annals of the photogrammetry, remote sensing and spatial information sciences*, vol. 4, pp. 271–277, 2017. [Page 4.]

[8] D. J. Lary, A. H. Alavi, A. H. Gandomi, and A. L. Walker, "Machine learning in geosciences and remote sensing," *Geoscience Frontiers*, vol. 7, no. 1, pp. 3–10, 2016. doi: https://doi.org/10.1016/j.gsf.2015.07.003 Special Issue: Progress of Machine Learning in Geosciences. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1674987115000821 [Page 4.]

[9] K. H. Thamaga, T. Dube, and C. Shoko, "Advances in satellite remote sensing of the wetland ecosystems in sub-saharan africa," *Geocarto International*, vol. 37, no. 20, pp. 5891–5913, 2022. doi: 10.1080/10106049.2021.1926552. [Online]. Available: https://doi.org/10.1080/10106049.2021.1926552 [Page 4.]

[10] J. Gu, Z. Wang, J. Kuen, L. Ma, A. Shahroudy, B. Shuai, T. Liu, X. Wang, G. Wang, J. Cai *et al.*, "Recent advances in convolutional neural networks," *Pattern recognition*, vol. 77, pp. 354–377, 2018. [Page 5.]

[11] S. Sharma, S. Sharma, and A. Athaiya, "Activation functions in neural networks," *Towards Data Sci*, vol. 6, no. 12, pp. 310–316, 2017. [Page 5.]

[12] A. Eltner, P. O. Bressan, T. Akiyama, W. N. Gonçalves, and J. Marcato Junior, "Using deep learning for automatic water stage measurements," *Water Resources Research*, vol. 57, no. 3, p. e2020WR027608, 2021. doi: https://doi.org/10.1029/2020WR027608 E2020WR027608 2020WR027608. [Online]. Available: https://agupubs.onlinelibrary.wiley.com/doi/abs/10.1029/2020WR027608 [Page 5.]

[13] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," 2015. [Page 6.]

[14] 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: 18th International Conference, Munich, Germany, October 5-9, 2015,*

Proceedings, Part III 18. Springer, 2015, pp. 234–241. [Pages 6, 25, and 26.]

[15] M. Mahdianpari, B. Salehi, M. Rezaee, F. Mohammadimanesh, and Y. Zhang, "Very deep convolutional neural networks for complex land cover mapping using multispectral remote sensing imagery," *Remote Sensing*, vol. 10, no. 7, p. 1119, 2018. [Page 6.]

[16] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE conference on computer vision and pattern recognition*. Ieee, 2009, pp. 248–255. [Page 6.]

[17] Z. Jiang, K. Von Ness, J. Loisel, and Z. Wang, "Arcticnet: A deep learning solution to classify arctic wetlands," *arXiv preprint arXiv:1906.00133*, 2019. [Page 6.]

[18] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778. [Page 7.]

[19] B. Cui, Y. Zhang, X. Li, J. Wu, and Y. Lu, "Wetlandnet: semantic segmentation for remote sensing images of coastal wetlands via improved unet with deconvolution," in *Genetic and Evolutionary Computing: Proceedings of the Thirteenth International Conference on Genetic and Evolutionary Computing, November 1–3, 2019, Qingdao, China 13*. Springer, 2020, pp. 281–292. [Page 7.]

[20] H. N. Pham, K. B. Dang, T. V. Nguyen, N. C. Tran, X. Q. Ngo, D. A. Nguyen, T. T. H. Phan, T. T. Nguyen, W. Guo, and H. H. Ngo, "A new deep learning approach based on bilateral semantic segmentation models for sustainable estuarine wetland ecosystem management," *Science of The Total Environment*, vol. 838, p. 155826, 2022. [Page 7.]

[21] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017. [Page 7.]

[22] B. Hosseiny, M. Mahdianpari, B. Brisco, F. Mohammadimanesh, and B. Salehi, "Wetnet: A spatial–temporal ensemble deep learning model for wetland classification using sentinel-1 and sentinel-2," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 60, pp. 1–14, 2021. [Page 7.]

[23] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997. [Page 7.]

[24] Y. Wang, C. M. Albrecht, N. A. A. Braham, L. Mou, and X. X. Zhu, "Self-supervised learning in remote sensing: A review," 2022. [Online]. Available: https://arxiv.org/abs/2206.13188 [Pages 8 and 9.]

[25] J. Ho, A. Jain, and P. Abbeel, "Denoising diffusion probabilistic models," *Advances in neural information processing systems*, vol. 33, pp. 6840–6851, 2020. [Page 9.]

[26] I. Kobyzev, S. J. Prince, and M. A. Brubaker, "Normalizing flows: An introduction and review of current methods," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 43, no. 11, pp. 3964–3979, 2021. doi: 10.1109/TPAMI.2020.2992934 [Page 9.]

[27] G. S. Hartnett and M. Mohseni, "Self-supervised learning of generative spin-glasses with normalizing flows," *arXiv preprint arXiv:2001.00585*, 2020. [Page 9.]

[28] F. Bordes, R. Balestriero, and P. Vincent, "High fidelity visualization of what your self-supervised representation knows about," *arXiv preprint arXiv:2112.09164*, 2021. [Page 9.]

[29] M. Caron, P. Bojanowski, A. Joulin, and M. Douze, "Deep clustering for unsupervised learning of visual features," in *Proceedings of the European conference on computer vision (ECCV)*, 2018, pp. 132–149. [Page 10.]

[30] L. Scheibenreif, M. Mommert, and D. Borth, "Contrastive self-supervised data fusion for satellite imagery," *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. 3, pp. 705–711, 2022. [Page 11.]

[31] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, "A simple framework for contrastive learning of visual representations," 2020. [Page 11.]

[32] L. Scheibenreif, J. Hanna, M. Mommert, and D. Borth, "Self-supervised vision transformers for land-cover segmentation and classification," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 1422–1431. [Page 11.]

[33] O. Manas, A. Lacoste, X. Giró-i Nieto, D. Vazquez, and P. Rodriguez, "Seasonal contrast: Unsupervised pre-training from uncurated remote sensing data," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 9414–9423. [Page 11.]

[34] K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick, "Momentum contrast for unsupervised visual representation learning," 2020. [Page 11.]

[35] Z. Cai, Z. Jiang, and Y. Yuan, "Task-related self-supervised learning for remote sensing image change detection," in *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2021, pp. 1535–1539. [Page 12.]

[36] C. Wu, B. Du, and L. Zhang, "Fully convolutional change detection framework with generative adversarial network for unsupervised, weakly supervised and regional supervised change detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2023. [Page 12.]

[37] O. Sagi and L. Rokach, "Ensemble learning: A survey," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 8, no. 4, p. e1249, 2018. [Page 12.]

[38] C. J. Vörösmarty, P. Green, J. Salisbury, and R. B. Lammers, "Global water resources: Vulnerability from climate change and population growth," *Science*, vol. 289, no. 5477, pp. 284–288, 2000. doi: 10.1126/science.289.5477.284. [Online]. Available: https://www.science.org/doi/abs/10.1126/science.289.5477.284 [Page 13.]

[39] N. Ghajarnia, G. Destouni, J. Thorslund, Z. Kalantari, I. Åhlén, J. A. Anaya-Acevedo, J. F. Blanco-Libreros, S. Borja, S. Chalov, A. Chalova, K. P. Chun, N. Clerici, A. Desormeaux, B. B. Garfield, P. Girard, O. Gorelits, A. Hansen, F. Jaramillo, J. Jarsjö, A. Labbaci, J. Livsey, G. Maneas, K. McCurley Pisarello, S. Palomino-Ángel, J. Pietroń, R. M. Price, V. H. Rivera-Monroy, J. Salgado, A. B. K. Sannel, S. Seifollahi-Aghmiuni, Y. Sjöberg, P. Terskii, G. Vigouroux, L. Licero-Villanueva, and D. Zamora, "Data for wetlandscapes and their changes around the world," *Earth System Science Data*, vol. 12, no. 2, pp. 1083–1100, 2020. doi: 10.5194/essd-12-1083-2020. [Online]. Available: https://essd.copernicus.org/articles/12/1083/2020/ [Page 13.]

[40] J. Leifeld and L. Menichetti, "The underappreciated potential of peatlands in global climate change mitigation strategies," *Nature Communications*, vol. 9, no. 1, p. 1071, Mar. 2018. doi: 10.1038/s41467-018-03406-6. [Online]. Available: https://doi.org/10.1038/s41467-018-03406-6 [Page 13.]

[41] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," 2015. [Page 20.]

[42] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *CoRR*, vol. abs/1502.03167, 2015. [Online]. Available: http://arxiv.org/abs/1502.03167 [Page 22.]

[43] A. Odena, V. Dumoulin, and C. Olah, "Deconvolution and checkerboard artifacts," *Distill*, 2016. doi: 10.23915/distill.00003. [Online]. Available: http://distill.pub/2016/deconv-checkerboard [Page 27.]

[44] N. Otsu, "A threshold selection method from gray-level histograms," *IEEE transactions on systems, man, and cybernetics*, vol. 9, no. 1, pp. 62–66, 1979. [Page 33.]

[45] S. Bhutada, N. Yashwanth, P. Dheeraj, and K. Shekar, "Opening and closing in morphological image processing," *World Journal of Advanced Research and Reviews*, vol. 14, no. 3, pp. 687–695, 2022. [Page 34.]

[46] L. R. Dice, "Measures of the amount of ecologic association between species," *Ecology*, vol. 26, no. 3, pp. 297–302, 1945. [Page 35.]

[47] T. A. Soomro, A. J. Afifi, J. Gao, O. Hellwich, M. Paul, and L. Zheng, "Strided u-net model: Retinal vessels segmentation using dice loss," in *2018 Digital Image Computing: Techniques and Applications (DICTA)*, 2018. doi: 10.1109/DICTA.2018.8615770 pp. 1–8. [Page 35.]

[48] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," *arXiv preprint arXiv:1711.05101*, 2017. [Page 36.]

[49] ——, "Sgdr: Stochastic gradient descent with warm restarts," 2017. [Page 36.]

[50] Z. Zhang, Q. Liu, and Y. Wang, "Road extraction by deep residual u-net," *IEEE Geoscience and Remote Sensing Letters*, vol. 15, no. 5, pp. 749–753, 2018. [Page 55.]

[51] Y. Sun, F. Bi, Y. Gao, L. Chen, and S. Feng, "A multi-attention unet for semantic segmentation in remote sensing images," *Symmetry*, vol. 14, no. 5, 2022. doi: 10.3390/sym14050906. [Online]. Available: https://www.mdpi.com/2073-8994/14/5/906 [Page 55.]

[52] R. Schwartz, J. Dodge, N. A. Smith, and O. Etzioni, "Green ai," *Communications of the ACM*, vol. 63, no. 12, pp. 54–63, 2020. [Page 55.]

TRITA-EECS-EX-2024:32