Master's thesis

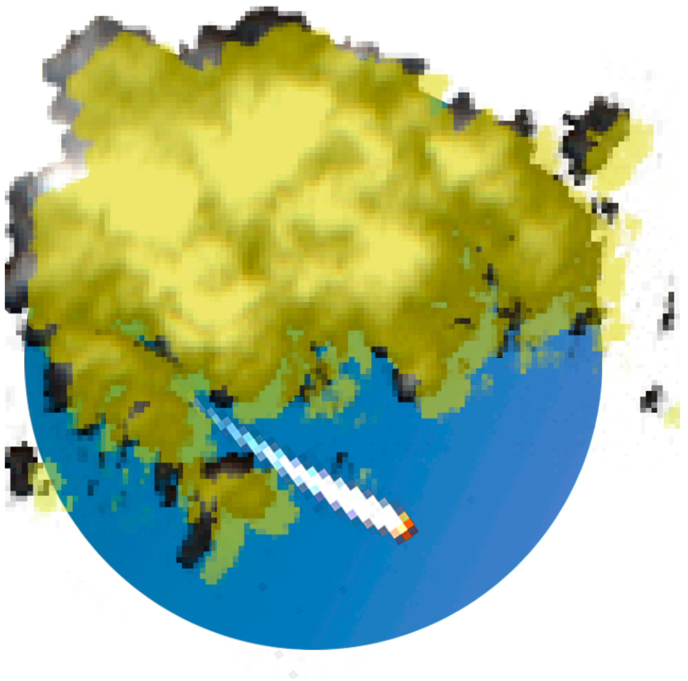
NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Computer Science

Lars Moe Ellefsen

Cloud Recognition and Masking of Earth Observation Imagery

An Optimized Approach for Automatic Labeling of Sentinel-2 Imagery for Object Detection

Master's thesis in Master of Science in Informatics
Supervisor: Svein-Erik Bratsberg
June 2019



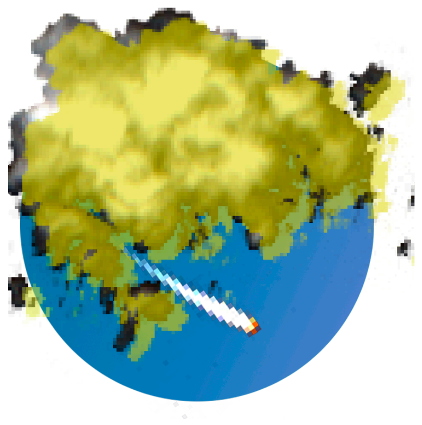
NTNU
Kunnskap for en bedre verden

VAKE

Lars Moe Ellefsen

Cloud Recognition and Masking of Earth Observation Imagery

An Optimized Approach for Automatic Labeling of Sentinel-2 Imagery for Object Detection



Master’s thesis in Master of Science in Informatics
Supervisor: Svein-Erik Bratsberg
June 2019

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Computer Science

Preface

This master's thesis is written for the Department of Computer Science at The Norwegian University of Science and Technology, as part of the Informatics Master Programme. The work took place over one year, from august of 2018 to june 2019 under the guidance of Professor Svein-Erik Bratsberg.

The idea for this work originated through a cooperation with Vake, whose founders had the previous year written their master's thesis on the topic of identifying ships in satellite imagery. Through our collaboration we identified the problem area for this thesis which would complement and further their previous and current work.

This work is an interdisciplinary study, combining machine learning, computer vision and remote sensing data to create the methods proposed in this thesis. The work is written to be understood by computer scientists, geologists and laymen with interest in the topic at hand. All the needed background is presented, but a good understanding of both computer science and remote sensing data is highly recommended to fully appreciate the work.

Trondheim, 2019–06–03

....

Lars Moe Ellefsen

Acknowledgement

I would like to dedicate a special thanks Adrian Tofting and all the people at Vake for great support and guidance, as well as allowing me to further the work done in their nominal masters thesis.

Also a big thank you to my supervisor Svein-Erik Bratsberg for great guidance and positive outlook through the whole project.

And a great thank you to Ida Cathrine Skogvoll for creating various illustrations used in this thesis.

Lastly I would like to thank my friends and family for support and letting me talk about clouds and deep learning for hours on end.

L.E

Abstract

In this project we have developed three different methods for analyzing Sentinel-2 satellite images containing ships, and determining whether the ship is obscured by a cloud or not.

Our motivation for this work is to find the optimal method for correctly labeling images for use as training data in a ship detection machine learning model. By creating three different approaches we are able to compare and evaluate each approach with each other and existing solutions, as well as create a baseline and detailed outline for further work.

We present the background and theory for our work to specify why and how we are doing this, as well as presenting an overview of what constitutes a correctly labeled image.

The three different methods proposed in this work is a fuzzy logic reasoning method for determining the probability that the ship is obscured, an image segmentation method for calculating the cloud cover over the ship, and lastly a deep learning model capable of producing cloud masks on a given image.

Through evaluation and analysis we uncovered that image segmentation was the most accurate method in correctly detecting when a ship is covered by a cloud, and we present the case for deep learning semantic segmentation being the optimal choice for further work given a better data set.

Sammendrag

I dette prosjektet har vi utviklet tre forskjellige metoder for å analysere Sentinel-2 satellitt-bilder som inneholder skip, med mål om å avgjøre om skipet er skjult av en sky.

Vår motivasjon for denne oppgaven er å finne den optimale metoden for å kunne klassifisere bilder av skip med høyest mulig nøyaktighet. Målet med bildene og klassifiseringen er å kunne bruke de som et treningssett for å lære en maskinlærings-algoritme å gjenkjenne og spore skip i Sentinel-2 bilder. Ved å utvikle tre forskjellige metoder tillater det oss å sammenligne og evaluere de forskjellige metodene med hverandre og eksisterende løsninger, og også legge et grunnlag og detaljert beskrivelse for videre arbeid på oppgaven.

Vi presenterer bakgrunnen og teorien for oppgaven for å gi leseren den nødvendige kunnskapen for å forstå hva vi gjør og hvorfor vi gjør det. Leseren vil også få en forståelse for hva som utgjør en korrekt og nøyaktig klassifisering av et Sentinel-2 bilde.

De tre metodene presentert i denne oppgaven er fuzzy logikk metode for å avgjøre sannsynligheten for at skipet er tildekket, en bildesegmenteringsmetode for å regne ut det potensielle skydekket over en båt, og til slutt en dyp lærings-metode som er i er stand til å produsere skysegmenter på et gitt bilde.

Gjennom evaluering og analyser har vi fastslått at bildesegmentering er den mest nøyaktige metoden for å finne ut om skip er tildekket av en sky eller ikke. Vi presenterer også argumentet for bruk av vår dyp læring metode som det optimale valget for videre arbeid, gitt et bedre datasett av høyere kvalitet.

Contents

Chapter 1

Introduction

In this chapter we will introduce the background and motivation for this work, as well as the objectives and how we aim to complete them. This chapter will give the reader an understanding of what we are doing and why.

1.1 Background

In the recent years there have been a growing investment in free and open earth obervation (EO) data through the Copernicus Programme, led by the European Space Agency (ESA) and the European Commission. The programme aims to deliver full, open and free-of-charge earth observation data for the public to freely use.

With the launch of the first Sentinel-2 satellite in 2015, the Copernicus programme delivers open high-resolution multi-spectral imaging for land services. With this wealth of highly detailed and free data there is a lot of untapped potential. Examples applications of this data is monitoring land and sea change, measuring water quality and natural disaster monitoring [1].

It is through Vake that I first became aware of these opportunities, as the founders of Vake had previously written a master thesis on this topic [2]. Vake is currently working on a solution combining machine learning, Sentinel-2 imaging data and Automatic Identification System (AIS) messages to provide automatic ship identification. By providing the machine learning algorithm with Sentinel-2 images based on the corresponding AIS ship messages, the algorithm can be trained to recognize ships through satellite imagery. This solution could provide detection of ships that are not broadcasting AIS signals, which could provide useful help for search-and-rescue operations, detection of illegal coastal activities and detecting and tracking of refugees traveling by sea without the proper equipment and preparations, amongst other.

Through the cooperation with Vake I learned that one of the challenges the project is facing is cloud recognition and cloud masking. The machine learning algorithm relies heavily on a consistent, large, clean and highly accurate training set of images containing ships. The problem arises when a ship is fully covered by a cloud, which could lead to the machine learning algorithm training on the wrong data, causing low accuracy and precision, and in the worst case, not working at all (e.g recognizing a cloud as a ship).

There are some pre-existing solutions to this problem, most notably from ESA themselves, which will be covered in Chapter 2.

However, even though these solutions exist, they tend to have a rather large error margin and are not directly applicable to this problem. In most circumstances this error margin would be adequate, but this error margin could propagate in the training set, which would greatly impact the quality if the machine learning model.

With this problem in mind, I structured my master thesis around this problem area, with the main goal of providing a way to detect and discard images where ships are fully covered by clouds, or if the image is in some way unusable as training data. The developed method(s) should also be able to correctly label images in situations where the ship is visible but partially obscured, as this variation will greatly help the machine learning model detect ships in difficult environments.

To achieve this we will be developing different methods of image validation, which can be compared to the existing solutions, as well as other methods developed in this work. With this we aim to not only provide a solution capable of generating a varied training set of high quality, but also a detailed evaluation of different approaches for this problem as a basis for further research

1.2 Objectives

Here we will present the main objectives which this work aims to complete:

1. Develop different methods for validating and correctly labeling ship images for use in ship detection.
2. Evaluate and compare the developed solutions.
3. The methods should be fast and optimized for ship level labeling.

In order to meet these main objectives, we also define a set of objectives which should be completed in order to reach these goals:

1. Generate images used for developing and evaluating the methods.
2. Develop a method for evaluating our proposed methods in comparison with existing solutions.
3. Define what makes an accurate label

1.3 Approach

In this work we will approach the problem by developing different methods within different fields. The fields we will be examining in this work are fuzzy logic reasoning, image segmentation and deep learning.

By comparing and evaluating these approaches together we can determine which approach is best suited for our objectives. This also allows us to compare, evaluate and discuss the different fields in the context of our problem, by looking at strengths and weaknesses. As the problem of cloud detection and masking in EO-images is a highly discussed problem, our evaluation of the different approaches will serve as the basis for further research.

In order to minimize the delay from the images are captured until we are able to analyze them, the ship detection will be done on the Sentinel-2 1C product. This is the earliest available product provided by ESA, and is the least processed level for a Sentinel-2 product. As such, level 1C will be the basis for all our methods in this work, ensuring that our developed approach is capable of correctly generating a training set on the 1C level. More information about the different processing levels can be found in chapter 2.1.2

Chapter 2

Background

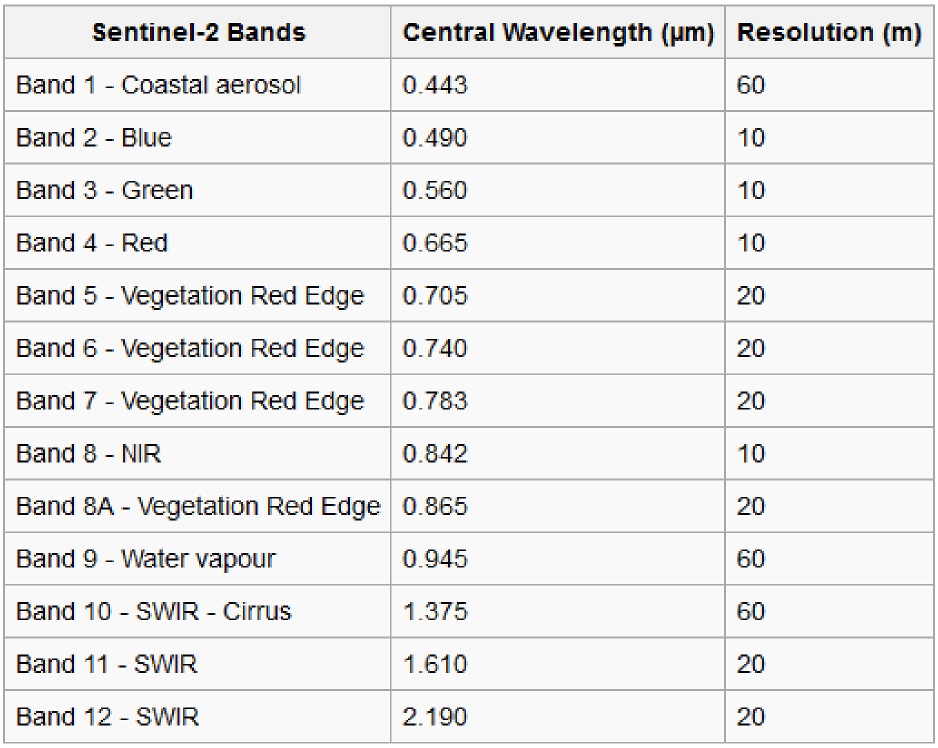
In this chapter we will present the relevant background theory needed for the work presented in this thesis. The reader will get an understanding of the theory behind our developed solutions as well as the theory in regards to our motivation.

2.1 Sentinel-2

As of 2019, the Sentinel-2 mission is composed of two satellites: Sentinel-2A, launched in 2015, and Sentinel-2B, launched in 2017. These twin polar-orbiting satellites are in the same orbit phased at 180° to each other. These satellites orbit with a 10 day cycle, which together give a 5 day temporal resolution at the equator. At higher latitudes, the temporal resolution can be as low as 2-3 days [3].

2.1.1 Spectral bands

The Sentinel-2 Multispectral Instruments samples 13 different spectral bands: four bands at 10 metres, six bands at 20 metres and three bands at 60 metres spatial resolution [4]. The spatial resolution is the measure of the smallest object that can be resolved by the sensor, which translates to the linear dimension on the ground represented by each pixel [5]. These 13 bands range from visible and near-infrared (VNIR) to Short-Wave Infrared (SWIR), giving each band different characteristics as shown in figure 2.1



Sentinel-2 Bands	Central Wavelength (µm)	Resolution (m)
Band 1 - Coastal aerosol	0.443	60
Band 2 - Blue	0.490	10
Band 3 - Green	0.560	10
Band 4 - Red	0.665	10
Band 5 - Vegetation Red Edge	0.705	20
Band 6 - Vegetation Red Edge	0.740	20
Band 7 - Vegetation Red Edge	0.783	20
Band 8 - NIR	0.842	10
Band 8A - Vegetation Red Edge	0.865	20
Band 9 - Water vapour	0.945	60
Band 10 - SWIR - Cirrus	1.375	60
Band 11 - SWIR	1.610	20
Band 12 - SWIR	2.190	20

Figure 2.1: The spectral bands of Sentinel-2 [6]

2.1.2 Data Products

The Copernicus Programme currently offers two different products to end-users, level 1C and level 2A. These two products are different levels of processing available for each image. There are multiple ways to access and download these products. ESA offer a web-based client as well as an API for downloading images programmatically, which are collectively called the "Copernicus Open Access Hub" [7]. As of 2019, products are also hosted on Amazon AWS, but with a slight delay of a few hours behind the Copernicus Hub. [8]

Level 1C

Level 1C is the least processed data product available to the public, and is achieved by processing the raw Sentinel-2 ground segment several times. Compressed raw data (level-0) is decompressed into level 1A where a geometric model is developed, allowing any pixel in the image to be located. Level 1B is then achieved by performing radiometric corrections on the level 1A product, and finally 1C is reached by resampling level 1B into orthorectified Top-Of-Atmosphere (TOA) reflectances together with computed land/water and cloud masks. The final 1C product is composed of orthorectified 100x100 km^2 tiles in UTM/WGS84 projection [4]. A detailed look into the computation of the 1C cloud mask can be in Chapter 2.2.

Level 2A

The 2A product is achieved by applying atmospheric corrections to the TOA 1C product, resulting in a Bottom-of-Atmosphere (BOA) corrected reflectance product. Additionally, a scene classification map, a water vapour map and Aerosol Optical Thickness map is computed for the 2A product [9]. A condensed overview of the scene classification algorithm is shown in figure 2.2, and an example of the scene classification map is seen in figure 2.3

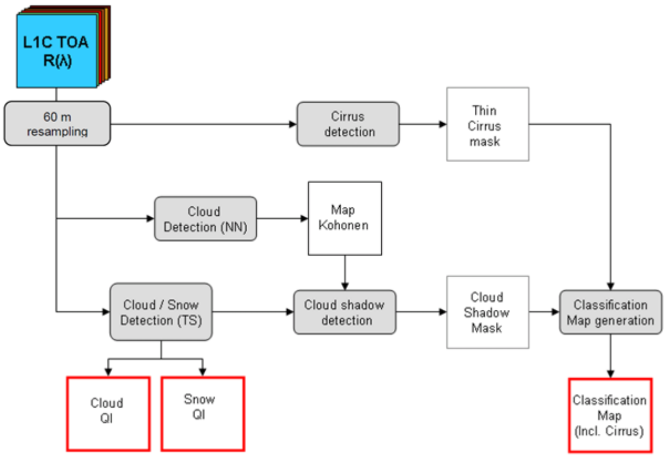


Figure 2.2: An overview of the Scene Classification algorithm for level 2A [9]

Metadata

All the Sentinel-2 products also come with a wide selection of metadata, which is queryable both through the API and the Copernicus Open Access Hub. The available metadata ranges from technical specifications regarding the onboard instruments to image related metadata such as when sensing started and ended, as well as geographic-specific data. This also includes a calculated cloud cover, detailing how much of the current tile is considered cloudy on a 0-100 percentage scale. Since the Sentinel-2 products are relatively large in size (500-700MB in average), it is often preferred to query and review the returned metadata before committing to a rather lengthy download. For training and experimenting purposes it will be beneficial to filter based on cloud cover and geographical position (e.g not over landmasses), resulting in suitable data products for our purposes.

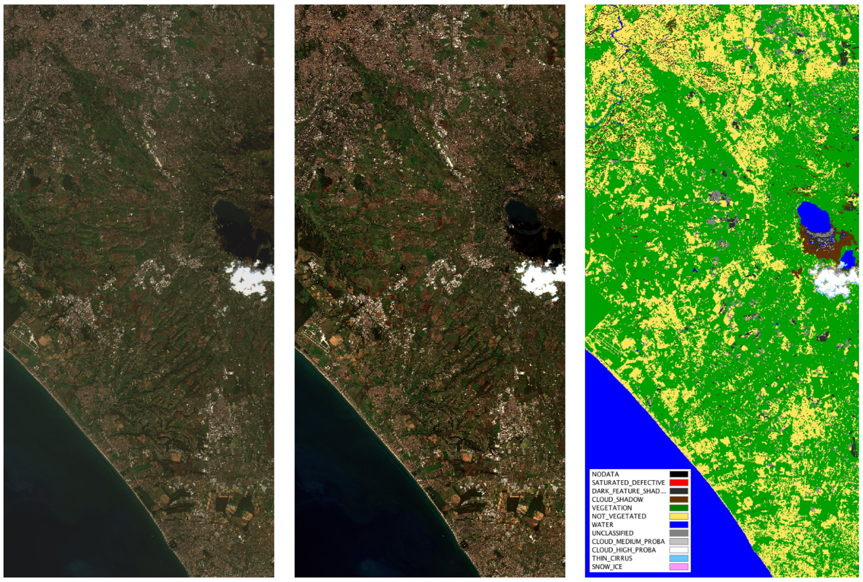


Figure 2.3: Left to right: Level 1C, level 2A and 2A scene classification map

2.2 Level 1C cloud masks

Level 1C products come with a pre-computed cloud mask in a Geographic Markup Language (GML [10]) vector format, which is computed on data sampled at 60m spatial resolution.

The cloud mask differentiates between dense clouds (also called opaque clouds) and thin, semi-transparent clouds called cirrus clouds. The 1C cloud masks include an indicator specifying the type of cloud identified. Some of the more transparent cirrus clouds (and other atmospheric haze) are naturally removed during the Atmosphere Correction process when going from level 1C (Top-of-Atmosphere) to 2A (Bottom-of-Atmosphere).

Dense clouds are characterised by a high reflectance in band 2 (Blue spectral region), thus the method of identifying dense clouds make use of a reflectance threshold in the blue band. Since snow and high altitude ice clouds are also highly reflectant in the blue band, the algorithm utilizes the Short-Wave Infrared (SWIR) reflectance in bands 10, 11 and 12 to differentiate between snow and clouds. Snow has a low SWIR reflectance, whereas clouds have a high SWIR reflectance.

Cirrus clouds on the other hand, due to the low opacity and density, has very little reflectance in the blue spectral region, but a high SWIR reflectance in band 10 due to their high altitude. Since band 10 is a high atmospheric absorption band only high altitude clouds are detected. Thus the algorithm can differentiate between dense clouds (High reflectance in band 2, band 10 and band 11), cirrus clouds (low reflectance in band 2, high reflectance in band 10) and snow (high reflectance in band 2, low reflectance in band 10, band 11 and band 12).

To limit false detection, the algorithm applies a filter using morphology-based operations to perform erosion (remove isolated pixels) and dilatation (fill gaps and extend clouds). After these operations, pixels are set to one of three values, where 0 is a cloud-free pixel, 1 is a dense cloud pixel and 2 is a cirrus cloud pixel. If measurements are not available in one or several bands that are needed to calculate the cloud masks, the mask value is set to NODATA.

After all filtering steps, the cloud mask is available at a spatial resolution of 60 m, but is resampled to 10m and 20m spatial resolution through a radiometric transformation [11].

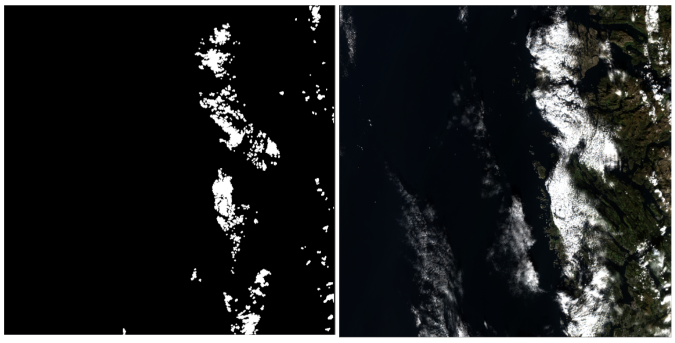


Figure 2.4: 1C product with corresponding cloud mask. Left: 1C cloud mask. Right: 1C RGB product

2.3 AIS

Automatic Identification System (AIS) is an automated system broadcasting information about the vessel to other vessels and coastal authorities. The current regulation requires that all ships over 300 gross tonnage and upwards engaged on international voyages, cargo ships of 500 gross tonnage and upwards not engaged on international voyages and all passenger ships irrespective of size be equipped with AIS-capable equipment [12].

AIS messages are required to be broadcast every 2 to 10 seconds (depending on the speed) while underway, and every 3 minute while the vessel is anchored. These messages includes:

- The vessel's Maritime Mobile Service Identity (MMSI) - A unique 9 digit code
- Navigation status
- Rate of turn
- Speed over ground
- Positional accuracy
  - Longitude – to 0.0001 minutes.
  - Latitude – to 0.0001 minutes.
- course over ground
- true heading
- True bearing at own position
- UTC Seconds when the data was generated

Additional data is broadcast every 6 minutes, which includes dimensions of the ship, type of ship, destination and an optional high precision time stamp amongst other.

2.4 Deep Learning

Deep learning is part of the broader term "Machine Learning", in which the model is based on an Artificial Neural Network (ANN). In this section the term "machine learning" and "deep learning" will be used interchangeably for the sake of simplicity.

Deep learning is often categorized into several broad categories: Unsupervised learning, Supervised learning and reinforcement learning. Our motivation for this work is to label and generate training data for object detection, i.e detection of ships, which shows promising results within the area of supervised learning. We will cover the concepts needed to understand our motivation and the qualities of a good training set, as well as the concepts needed for our proposed method.

2.4.1 Neural Networks

In order to understand how a deep learning model is able to learn, we will present a general overview of the underlying ANN structure. A more in-depth look and implementation of a neural network is presented in chapter 3.4.3.

A neural network is is composed of connected nodes (Often referred to as a 'Neurons') arranged in layers, where the intermediate layers are called the 'hidden layers'. If the topology of the network is a Directed Acyclic Graph (DAG), the network is known as a Feedforward Neural Network, which is what we will be examining in this section. The edges between the nodes are weighted and the nodes are given biases. When a node receives input from the preceding node, it puts this weighted input through an **activation function** together with the bias to produce an output. Based on the network and the activation function this output can either dictate if the node should fire or not, as is the case when using a Step function, or it can be used as input for the ensuing nodes. Most commonly used in modern neural networks is the Rectified Linear Unit function (ReLu), which will output 0 if the produced value is negative, otherwise output the value as-is. ReLu can be written as:

$$f(x) = max(0, x)$$

When data has passed through the network, the final layer will produce an output. During the training phase this output is compared against the ground truth and the error is calculated using a loss function (Sometimes referred to as a *Cost function* or an *Error function*). This loss is a measurement of "how wrong" the prediction is. The loss is then **backpropagated** through the network, adjusting all the parameters accordingly in order to minimize the error.

How the parameters are adjusted is through an **Optimizer**, which aims to calculate the minimum for the loss function. An optimizer uses learning rate and momentum to adjust how much the parameters should be changed given a loss value, where slower means longer training is required, but will result in higher accuracy.

2.4.2 Convolutional Neural Networks

A Convolutional Neural Network (CNN) is a specific type of neural net that is suited for images. The input layer of the CNN takes in a tensor with shape $(Depth \times Width \times Height)$, where the depth is the amount of channels in the input image, and the width and height are the dimen-

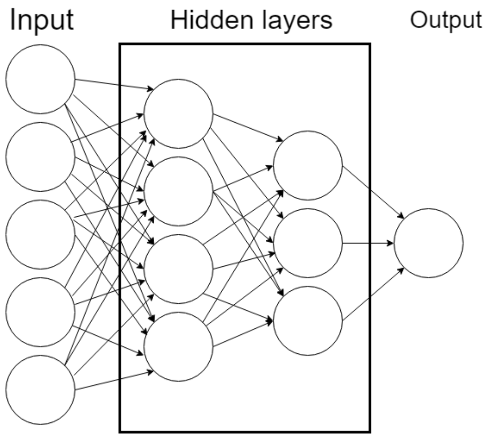


Figure 2.5: A neural network with two hidden layers

sions of the image. Typical channels for an image is the red, blue and green (RGB) channels. The image is represented as a 3D array of pixel values which is then put through the network.

A CNN is composed of **Convolutional layers** that uses learnable filters to detect certain features from the image. A filter is a NxN matrix (Kernel) that slides across the image, computing the dot product between the matrix and each NxN block of pixels, resulting in a new image. This process is repeated a set number of times, with each iteration being able to identify more abstract features. A pooling layer is used to down-sample the image in order to save memory, reduce the parameters and save computational time. Most commonly used is *max pooling* [13], described in Appendix B.2.

Once the data has passed through the convolutional and pooling layers, the data is sent through a fully connected neural network, producing an output. By back-propagating the loss, the network adjusts the parameters as before, but the values in the filter kernels are also adjusted accordingly. This is what makes the CNN able to learn different features embedded in the training set.

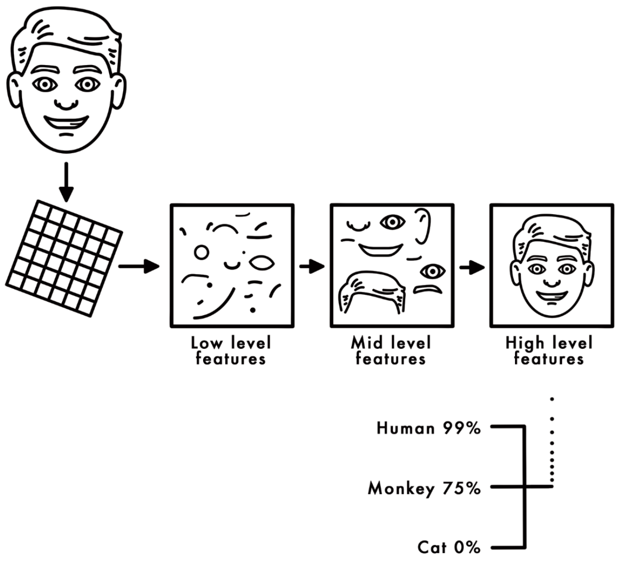


Figure 2.6: Abstract example showing how a CNN processes increasingly more complex and abstract features in images

2.4.3 Training data

A supervised learning model takes a set of data that contains both the input and the desired output [14]. For object detection this will be an image and a label which represents the ground truth for the image, which in the case of ship detection will be labels representing the class of the image, i.e *"ship"* and *"no ship"*. How well these labels match the input images is called the **accuracy** of the data, and is arguably one of the most important aspects of the training data in relation to the final correctness of the machine learning model [15]. The effect of a mislabeled image can be offset by having a large volume of data, as this lessens the effect of each mislabeled image. However it is important that the overall accuracy of the data set is as high as possible, as a machine learning model will only ever be as accurate as the data it trained on, which is a considerable bottleneck. Thus it is also important to have a good **variety** in the data set.

With this work we not only aim to provide precise labeling (Accuracy), but also a good variety. Our proposed methods aim to to correctly label ships in a variety of situations, such as ships in clouded environments that are still visible. This introduces not only good **representation** of real-life scenarios, but also a good variety for the model to train on. Further augmentation techniques can be applied to the data set to provide even more variety, such as random cropping, rotation and scaling. The line between obscured and not obscured is not always easy to differentiate, and as such it is not an easy task to define what makes an accurate label. In this work we try to define an unobscured ship as when ship features are *easily* distinguishable by a human. The idea behind this is that if a human can see and process the information required to distinguish the boat, the machine should be able to also. As such we will manually label all images used for evaluation in this work.

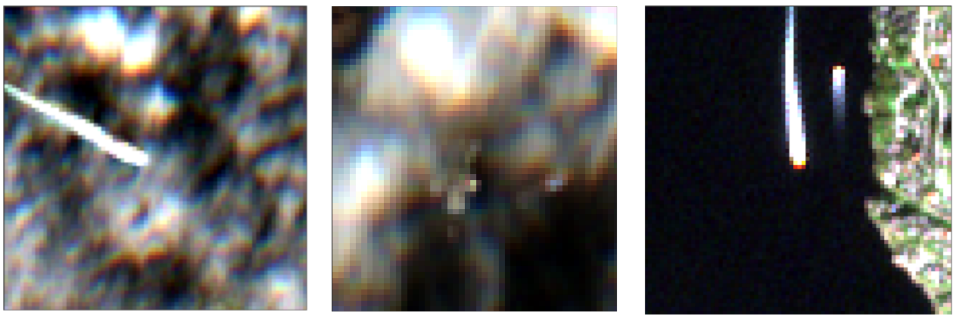


Figure 2.7: Different examples of variety in ship training images, from clear to nearly obscured.

Volume, or the size of the data set, is another aspect that is important for deep learning. There is no one-size-fits-all as different tasks requires different amount of data. For multiclass classification there should be a relatively balanced amount of images per class, so for each class another set of *N* images are required. This number might be lower for binary classification, but in general a large data set is always to be preferred.

The goal of a neural network is to achieve **generalization**, in which the model can make accurate predictions based on unseen data. In the case of the model scoring great accuracy on test images, but lower accuracy score on unseen images, we say the model is **overfitted**, which is an unwanted effect we want for our model. In a way we can say that our model has "memorized" the training set instead of actually learning to generalize. This can be caused by not having a large enough data set, or if the model is overtrained on the data set. A common method to avoid overtraining is to continually check the prediction accuracy on a set of unseen validation images between training rounds (Often called epochs) and stopping the training when the accuracy on the validation images reach a peak. A clear sign of overtraining is when the accuracy continually increases for training images, but stagnates/converges for validation images, or getting gradually worse validation accuracy.

A data set is often split into three different parts:

- Training set
  - A set of images and corresponding labels that the model will train on.
- Validation set
  - A set of images and corresponding labels that the model will use to check the prediction accuracy on between training sets.
- Test set
  - A set of images and corresponding labels that will be used as a final benchmark after testing has completed.

2.4.4 Training

The training phase is where all the previous theory comes together to learn and produce outputs. During the training phase we often define a couple of parameters:

- Epoch
  - One epoch is a pass of all the training data
- Batch size
  - The number of inputs passed through the network before parameters are updated

These parameters are called **Hyperparameters**, and should not be confused with the trainable parameters in the model. There is no one way to correctly set these hyperparameters, and is often set through continuous testing, tweaking and iteration. Batch size defines the number of inputs the network should process before adjusting parameters in the model, and as such can have a noticeable impact on the accuracy of the model. Since whole batches are kept in memory during training, it is not feasible to process the whole data set before updating weights. A batch size of one input might be memory efficient, but the model might not adjust parameters correctly if the images are highly varied. How to set the optimal batch size is a highly debated topic and is left for the designer tweak accordingly.

The number of epochs required until you reach convergence (I.e a desired accuracy) varies depending on the data and network implementation. **Early stopping** is employed here to avoid overfitting, stopping the training if the model shows signs of overfitting to the data set.

Algorithm 1 shows a basic outline of a simple training process:

Algorithm 1: Training process

```
1  $Epochs \leftarrow$ Number of epochs
2  $batchsize \leftarrow$ Number of inputs for each batch
3  $Trainingset \leftarrow$ Set of inputs and corresponding labels
4  $Validationset \leftarrow$ Set of inputs and corresponding labels
5  $Net \leftarrow$ The neural network
6  for $i \leftarrow 0$ to $Epochs$ do
7      $Batches \leftarrow$ getBatches(Training set, batch size)
8      foreach Batch in Batches do
9          $Output \leftarrow$ Net(batch)
10         $Loss \leftarrow$ lossFunction(output)
11         $Net.Backwards(Loss)$
12     end
13     $Batches \leftarrow$ getBatches(Validation set, batch size)
14     foreach Batch in Batches do
15         $Output \leftarrow$ Net(batch)
16         $Loss \leftarrow$ lossFunction(output)
17     end
18 end
```

Chapter 3

Proposed Methods

In this chapter we will be presenting the technical implementation, analysis and experiments done for the work. The end goal for all the proposed methods is to correctly identify when a ship is obscured by a cloud, and label it as such. We also want our methods to be able to identify ships that are semi-obscured but still distinguishable from clouds, as we want our methods to be able to label a data set with a high variety. This should not come at a cost of mislabeling obscured ships however, as this could have a major negative impact when used as training data for a machine learning model.

The approach for this work involves three different methods in different fields. The methods will be developed in an iterate fashion, building on analysis done in previous methods. This chapter presents the implementations used for each of these methods, as well as the evaluations. All the following solutions will be written using Python.

3.1 Pre-processing

First off we break down our main objective into "Are there any clouds in this image" and "If so, do they obscure the ship in any way?". In order to answer these questions, the first step is to reduce the problem area and focus only on parts of the image that we know for certain contain ships. In order to do this we need to know where the longitude/latitude positions of the ships in the image.

By combining AIS time stamp and longitude/latitude position, we can find areas with a large amount of boats in the same time frame. This allows us to find the corresponding Sentinel-2 1C data product by querying metadata based on the time stamp and geographical location from the AIS messages. Figure 3.1 shows a 1C product with 121 identified ships, as a result of this process.

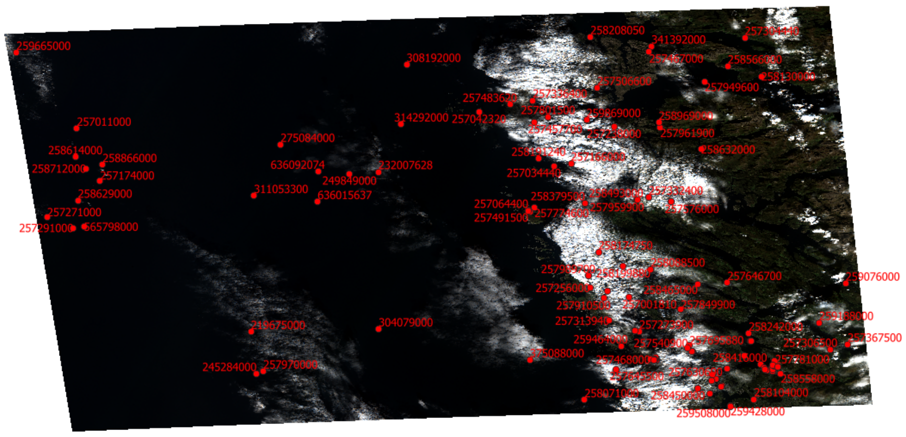


Figure 3.1: A 1C product with ship positions (Marked in red) found by combining AIS messages and Sentinel-2 metadata.

With the ships identified and the correct data product retrieved, we can cut out three sets of images for each ship:

- One set of 78x78 px images containing all the 10m bands (Band 02, 03 and 04)
- One set of 39x39 px images containing all the 20m bands (Bands 05, 06, 07, 08A, 11 and 12)
- One set of 13x13 px images containing all the 60m bands (Bands 01, 09, and 10)

Since the spatial resolution for the different bands vary between 10, 20 and 60, these raster sizes allows us get the same size for each set of image in actual metres, which is $780m^2$. These images are produced by using the Geospatial Data Abstraction Library (GDAL) Python wrapper, outputting GeoTiff files. For this work we will be using both uint16 and float32, especially as float32 is better suited for deep learning later on in the work. This gives us rasters where each pixel value is represented by an unsigned integer value with a range of 0-65535 in the case of uint16. Each raster is originally greyscale, so these pixel values represents the intensity of the pixel, where higher equals brighter.

Figure 3.2 shows a $78 \times 78px$ cutout produced with this method. The image is shown using a colormap (As it is originally greyscale), where brighter colors indicate higher intensity values:

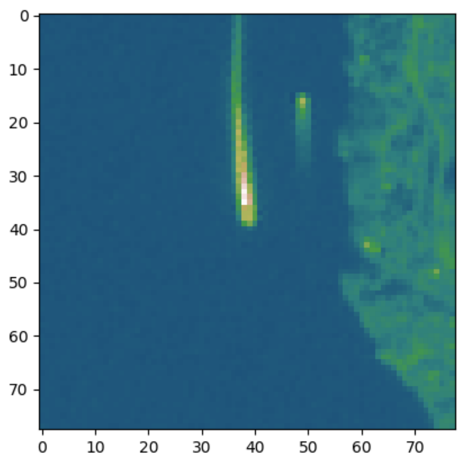


Figure 3.2: Example of a 78x78px cutout produced by translating the lat/long coordinates of the ship to raster x/y coordinates. Image is shown using a colormap where brighter equals higher intensity.

3.2 Fuzzy logic reasoning

With isolated pictures of ships, we can begin engineering a solution to test whether a ship is obscured by a cloud or not. Firstly we need to find a threshold for when a pixel should be considered a cloud. This value varies slightly based on what band we're currently processing. As previously seen, dense clouds are characterised by a high reflectance in the blue band (band 2), so band 2 will theoretically yield a bigger difference in values between cloud pixels and non-cloud pixels. By calculating the average value of pre-determined pixels we know to be dense, opaque clouds, it averages out to a value of roughly 1400 (uint16), or 5.0 for float32. We can use this value as a starting threshold for cloud cutoff, but different values can be used when testing for accuracy.

We can utilize the other bands to filter out false-positive pixels that exceed this threshold. Urban cityscape and certain vegetation near shore can be highly reflectant in all bands, and as such can negatively impact the calculation. To help minimize these potentially falsely identified pixels, we can use a combination of the SWIR (band 10 through 11) and NIR (Near-Infrared band 8) bands. The NIR band will yield considerably higher values for land and vegetation than the blue band, so we will disregard pixels that have considerably higher reflectance in the NIR band than in the blue band as this in most cases indicate land cover.

We can also apply this filtering technique with the 20m SWIR bands. As SWIR bands will reflect off high-altitude clouds, high SWIR will be indicative of higher-altitude clouds, thus pixels that exceeds both the SWIR and dense cloud threshold can safely be assumed to be clouds [16]. It is important to set conservative thresholds for the SWIR and NIR bands, as we don't want to potentially filter out actually clouded pixels. This filtering should only catch pixels where these properties are clearly distinguishable. Thus there may still be cases of falsely identified pixels, such as in urban areas and shorelines with a lot of unpredictability, but these extra measures should minimize the potential false-positives.

In order to calculate the cloud coverage of the image, one could simply iterate over the pixel value and count how many pixels are identified as clouds. This value can then be compared to a set threshold in order to determine if the image is obscured or not, but this approach raises some interesting questions and obstacles. This output from this approach could be assumed correct if nearly all the pixels are identified as clouds, but for lower counts this approach is less reliable. Consider these cases:

- The ship is completely obscured by a small cloud, but the rest of the image is cloud free, resulting in a low cloud count. The ship would be falsely marked as unobscured.
- The image has a lot of clouds around the ship, but the ship itself is unobscured. The ship would be falsely marked as obscured.

One interesting approach may be to simply check if the ship pixel exceeds the threshold, which would imply that the ship is behind a cloud.

This will not be sufficient for our work, as slight inaccuracies in the coordinates can cause the point to drift from the actual ship. The ship itself may also be highly reflectant and may exceed the cloud threshold. As such it is necessary to check surrounding pixels as wells.

In order to counter this, we need to know roughly where the pixels are in relation to the ship. We can transform the longitude/latitude coordinates of the ship into the corresponding pixel in the raster by using GDAL to map pixel/line coordinates into georeferenced space. This gives us six coefficients we can use to calculate which pixel the coordinates results in, thus giving us "ship pixel". We can calculate the distance from each pixel to the ship pixel by using the Euclidian distance (see appendix B.1), where a lower number equals closer to the ship.

With a spatial reference we can begin to calculate a score for each pixel exceeding the cloud threshold. By using the current pixel value, the current distance to the ship pixel and the total distance we can construct a basic decreasing function:

$$score = intensity * \left(\frac{a}{ln(totalDistance+1)}\right) * ln(distance+1) + a$$

where a is a constant, $intensity$ is the value of the pixel and $totalDistance$ is the max distance from the ship that a pixel can be.

This function gives us a weighting coefficient that decreases from a to 0 non-linearly as the distance increases to $totalDistance$. We can also create a further bias by multiplying all pixels within a certain distance with another weight coefficient, making our decreasing function non-linear favoring closer pixels. With this weighting function we can lessen the value of far-away pixels without disregarding them completely which should result in a more precise result. By summing the resulting intensity scores, we can then calculate a confidence score for our ship.

Another question that arises is how to determine the threshold for the

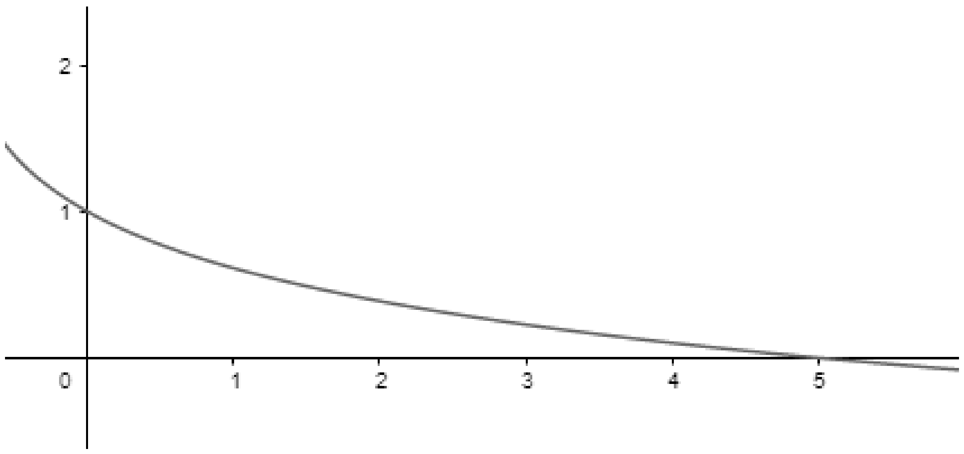


Figure 3.3: A graph of the decreasing function where $a = 1$ *and* $totalDistance = 5$

result. There is no clear-cut answer here and requires testing different threshold to find the best cutoff for our purposes. We can approximate different values by creating different models we can run our algorithm on.

We test the following models:

- Where all pixels within $totalDistance$ are at the threshold value
- Where all pixels within $totalDistance$ have a value greater than the threshold
- Where half of all the pixels within $totalDistance$ are at the threshold value
- Where half of all pixels within $totalDistance$ have a varying value greater than the threshold.
- Where a third of all pixels within $totalDistance$ are at the threshold value.

As dense, opaque clouds tends to be coherent, the pixels are set relative to each other as to best simulate real world scenarios. This gives us a theoretical upper threshold, as well as some intermediate values. But due to the nature of the images, values can vary significantly above our threshold, so we treat these values as starting points. By combining this with real world examples of clouds, as well as constructed test images, we can narrow down a suitable threshold value that should hold in most scenarios.

Comparing our model result with different parameters to clouded 1C images, we see a correlation between our model with a third of the pixels at the threshold and actual clouded images. Images with dense, highly reflectant clouds gives a significantly higher value, but for images with a moderate amount of cloud cover we see a clear correlation with our model results. This gives us a baseline value for the threshold.

We test this threshold on a set of constructed test scenarios. By creating greyscale images with either 16 or 32-bit depth, we can simulate different scenarios to see if our threshold is adequate for certain edge-cases. Some of the results from these simulations can be seen in Figure 3.4 (below).

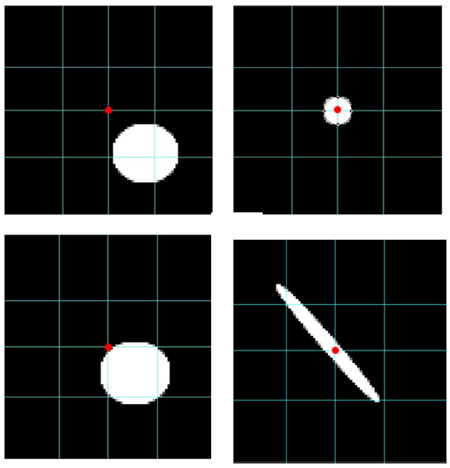


Figure 3.4: Constructed scenarios correctly identified using the threshold values uncovered from our experiments. The red dot represents our ship positions, while the white masks represent clouds.

3.2.1 Fuzzy logic

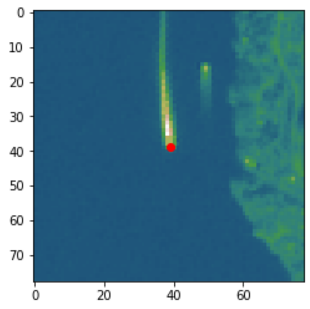
Our current solution utilizes a fuzzy logic approach, where there are no clear-cut true or false values. Pixel values for clouds can differ significantly based on a number of different factors, thus knowing when a pixel goes from transparent to opaque is not a straight-forward task in most cases. Problem also arises when defining when a ship is properly obscured, or if it is still visible through the clouds. Thus deciding when and how a ship is completely obscured is not only hard to decide, but can also be hard to measure. In our approach exploit these properties by using pixel values to represent a range between "completely false" and "completely true", where a higher value represents a higher confidence that the pixel is opaque and is indeed a cloud pixel. This also enables us to use distance to further modify the truth values of each pixel.

Our algorithm can be seen as a "Defuzzification" process that aims to convert these fuzzy problems into crisp logic/boolean logic, in our case the output values 'true' or 'false'.

3.2.2 Evaluation

We can test our current implementation and values on a couple of known images and compare the results. This initial evaluation should show us how the algorithm works in practice, and if the values from our experiments holds for real images. The implementation returns ***true*** if the ship is obscured, and ***false*** if the ship is unobscured. The following tests uses 1 as the constant a for calculating cloud scores:

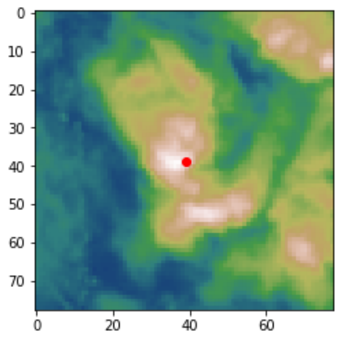
First a completely cloud free image:



```
[87]:  False
```

Figure 3.5: Cloud free image where $a = 1$ and $totalDistance$ = 20. Correctly labeled as unobscured (False)

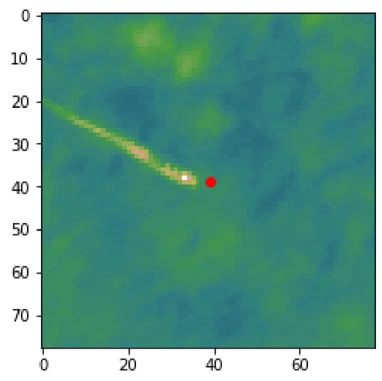
A clouded, obscured ship:



```
[90]:  True
```

Figure 3.6: Obscured ship image where $a = 1$ and $totalDistance$ = 20. Correctly labeled as obscured (True)

A partially obscured ship/Semi-transparent clouds:



```
[92]: False
```

Figure 3.7: Semi-transparent clouds where a = 1 *and $totalDistance$ = 20, correctly labeled as unobscured (False)*

For these three known test images, the algorithm is able to correctly identify the images.

In order to properly evaluate our solution, we can compare our results with other existing solutions. For this we have chosen to compare with the 1C cloud mask that accompanies the 1C data product, and a cloud mask algorithm F-mask [17]. F-mask is one of the more popular cloud-masking algorithms for Sentinel-2 images, initially developed for Landsat-8 products, but in the recent years have been developed for use in Sentinel-2 products as well. Finally we will also compare our solution to the cloud classification masks from the 2A product.

Since the 1C cloud masks are given as polygons we can check if each point falls within the polygon or not. This is achieved by using a ray-casting algorithm, and for this work we're using the Shapely python implementation of the algorithm.

For F-Mask we need a different approach to judge whether the point constitutes a cloud or not. Since a F-Mask image is composed of pixels values ranging from [0-4], where 4 is a cloud pixel, we need to check if the coordinates translate to a pixel of value 4. In order to get a more precise result we also look at the adjacent pixels, and if all pixels are of value 4, the point is safely assumed to be within the F-Mask cloud mask.

The comparison is done with several different parameters for our solution.

Comparison where total distance = 20 and a = 1.0

The following results are from a comparison where our solution looks at pixels within a distance of 20, with a weight coefficient decreasing nonlinearly from 1 to 0 as the distance increases.

Of the 121 ships, 66 are marked as ***False*** (unobscured) and 10 are marked as ***True*** (obscured) by all three solutions. By manually reviewing these two sets of results, we can see that they are all correctly identified.

This leaves us a data set of 45 ships that have conflicting results between our solution and the two others. A quick overview shows that the 1C cloud mask identifies all of the 45 ships as unobscured;***False***, whereas the F-Mask cloud mask identifies all the 45 ships as obscured; ***True***.

By manually reviewing each ship, we get the following results:

- 16 images correctly identified by our algorithm, that F-Mask cloud mask incorrectly identified (Falsely marked as obscured). Some examples of these images can be seen in Figure 3.8.
- 27 images correctly identified by our algorithm that the 1C cloud mask incorrectly identified (Falsely marked as unobscured).
- 2 images with uncertain verdict. These are images where the GPS error margin makes it difficult to correctly judge the validity of the results. This is often due to the ships being near shore or at anchor, and the ship position lands on nearby land cover.

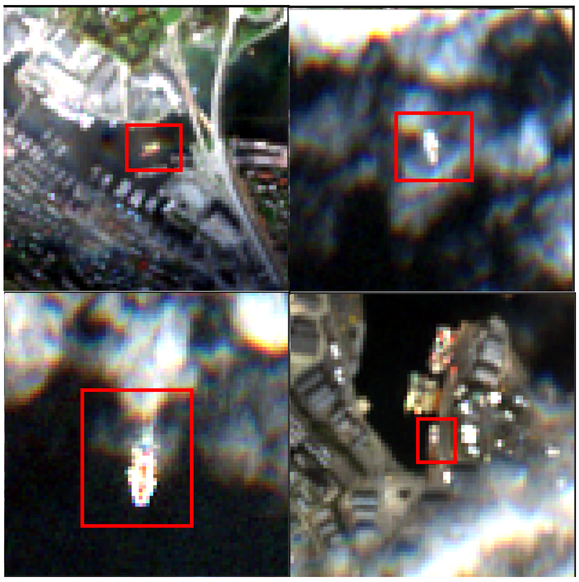


Figure 3.8: Four examples of ships incorrectly identified when using F-Mask cloud masks.

Thus, for 119 boats (if we remove the uncertain images), where we know that 38 ships are obscured by clouds and 81 ships are not obscured:

Results			
	1C cloud mask	F-Mask masks	Our method
Obscured ships	10 of 38 correctly identified	38 of 38 correctly identified	37 of 38 correctly identified
Unobscured ships	81 of 81 correctly identified	65 of 81 correctly identified	81 of 81 correctly identified
Total	91 of 119 in total (76.47%)	103 of 119 in total (86.55%)	118 of 119 in total (99.16%)

Table 3.1: The results of 1c cloud masks, F-Mask cloud masks and our method when evaluatiing 119 known images with distance = 20 and a = 1.0

Our method has lowest error margin so far, with only one ship that is incorrectly labeled (Figure 3.9). Examining this image, we can see that the mislabeling occurs because of the ribbon-like structure and semi-transparent clouds. This can be remedied by either lowering the threshold (which may cause other false positives) or further tweak the weighting algorithm.

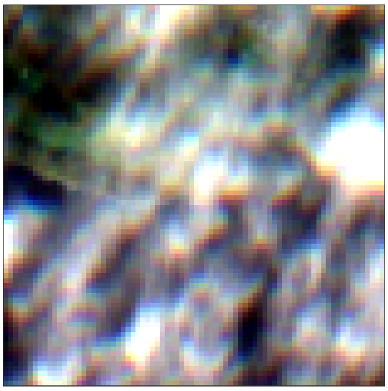


Figure 3.9: Image incorrectly labeled as unobscured by our solution.

Comparison with a steep decreasing function where totalDistance = 20.0

The following results are from a comparison where our solution uses an aggressive cutoff for our decreasing function, highly valuing pixels with a distance less than 10.

Of the 121 ships, 66 are marked as ***False*** (unobscured) and 10 are marked as ***True*** (obscured) by all three solutions. As before, these ships are all correctly labeled. This leaves us a data set of 45 ships that have conflicting results between our solution and the others.

By manually reviewing each ship, we get the following results:

- 12 images correctly identified by our algorithm, that F-Mask incorrectly identified (Falsely marked as obscured).
- 27 images correctly identified by our algorithm that the 1C cloud mask incorrectly identified (Falsely marked as unobscured).
- 2 images with uncertain verdict.
- 4 images incorrectly identified by our algorithm.

As in the last section, the results for F-Mask and the 1C-mask is the same. Thus, for 119 boats, where we know that 38 ships are obscured by clouds and 81 ships are not obscured:

- Our solution: 115 of 119 ships correctly identified. (97.47%).
  - Obscured ships: 38 of 38 correctly identified
  - Unobscured ships: 77 of 81 correctly identified

With a more aggressive weighting of nearby pixels, we are able to correctly label all obscured ships, but at the cost of four new incorrectly labeled images.

3.2.3 Further testing

We can test this further by doing the same analysis on different sets of ships, using parameters with the lowest error margin. In order to avoid overfitting our solution to the set, we can introduce some variety and randomness by having the ship position be anywhere in the cut image. In the previous test we have mainly been cutting our images with the ship in the center, but we can generate some randomness by having multiple cut images of ships in non-static positions. As seen in figure 3.10, different positions introduces new information in the image. With this we can test if our results are deterministic, i.e if all images for a ship have the same results, or if the results vary depending on the position and information available. Our new data set consists of 108 new ships, each with 5-10 images with the ship in different positions.

The same algorithm is then applied to each image. In the interest of avoiding false-positives, a ship and all its images are marked as obscure if a single of the images is marked as obscured.

For 108 new ships, where only 1 is completely obscured, we get the following:

- 1C cloud mask: 106 of 108 ships correctly identified in total. (98.15%).
  - Obscured ships: 0 of 2 correctly identified
  - Unobscured ships: 106 of 106 correctly identified
- F-Mask: 102 of 108 ships correctly identified. (94.44%)
  - Obscured ships: 2 of 2 correctly identified
  - Unobscured ships: 102 of 106 correctly identified
- Our solution: 108 of 108 ships correctly identified. (100%).

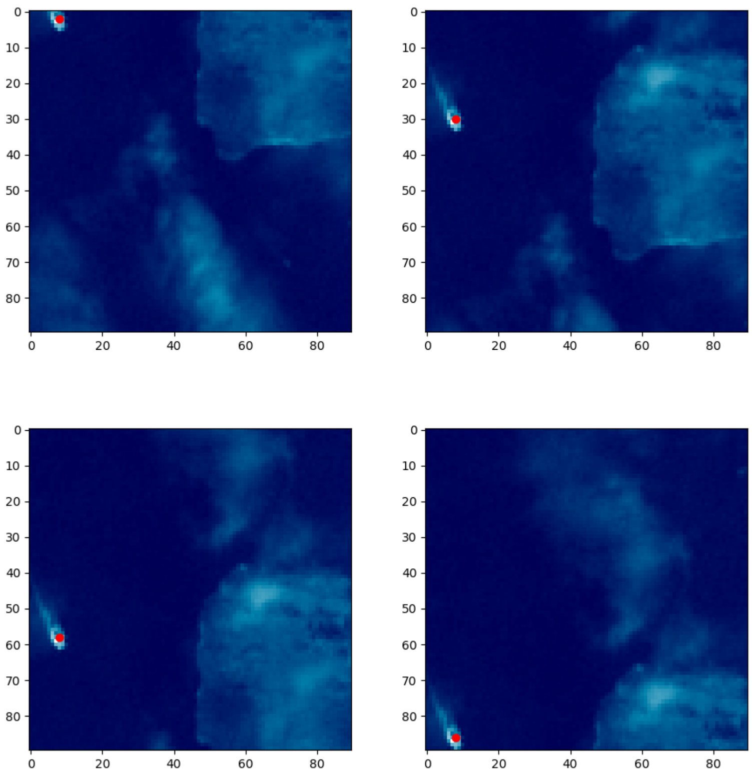


Figure 3.10: Same ship in four different positions

3.2.4 Conclusions

From this we can see that the 1C cloud masks are highly inaccurate and that F-mask is significantly more accurate. A ship can however be safely assumed to be obscured if it falls within the 1C masks, as they are considerably more conservative than other solutions. Although F-mask is highly accurate for most use cases, it is a very computationally expensive algorithm (With computation time ranging from 10 to 40 minutes depending on the image, cloud cover and hardware), so for our solution we can both reduce the computational power needed as well as provide fine-grained accuracy on a ship level. Closer inspection of F-Mask also shows some interesting results:

F-Mask sometimes incorrectly masks large, highly reflectant ships as clouds, as seen in figure 3.11.

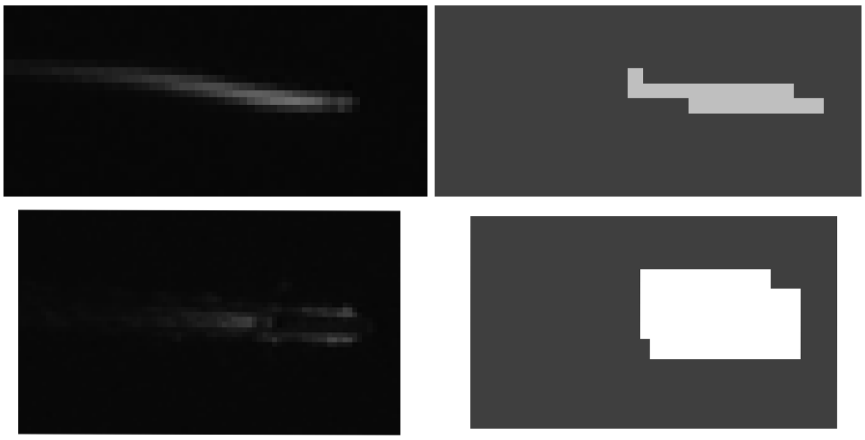


Figure 3.11: F-Mask incorrectly masking larger ships

This is the opposite of the properties displayed by 1C, both of which would be inadequate for our usage.

Our solution proves to be quite effective, with a lower error margin than both F-mask and 1C cloud masks. With an aggressive weighting of pixels a short distance from our ship we can in most cases guarantee that a ship is indeed obscured, and we can guarantee that a ship is not obscured if there are little to no clouds present in the image.

3.3 Image Segmentation

In the previous examples we have used a fuzzy logic approach to labeling images, combining several factors to output a Boolean value based on a threshold. This method proved to be effective, but relies heavily on changing factors and initial guesswork. Another popular approach within the field of computer vision is image segmentation. Image segmentation is the process of partitioning an image into multiple segments based on some common characteristic. Using what we learned from our previous results we can construct an image segmentation algorithm to eliminate some of the variables and guesswork previously needed. With a segmentation approach we aim to do a binary classification of each pixel, i.e cloud region pixel or clear pixel.

One of the simplest approaches to this is a threshold-based region-growing algorithm. Region-growing works by growing outwards from an initial seed point, incorporating neighbours based on similarity (threshold). When there are no more similar neighbours to incorporate, the segmentation is done, and the region is the list of neighbouring pixels that matched the given criteria.

Algorithm 2 shows the pseudo-code for a simple region growing algorithm we will base our solution on:

Algorithm 2: Region Growing Pseudo-code

```
$SeedPoint \leftarrow$ Initial Seed point
$Processed \leftarrow$ Empty list of processed pixels
$Queue \leftarrow$ Empty queue
$Queue \leftarrow$ Push seed point
$Region \leftarrow$ Empty region matrix
**while** $Queue \neq Empty$ **do**
    $Pixel \leftarrow$ *Queue.pop()*
    $Region \leftarrow$ *Push* $Pixel$
    **foreach** *Neighbour* **do**
        **if** *Neighbour* $\geq$ *threshold* **then**
            $Region \leftarrow$ *Push* $Neighbour$
            **if** *Neighbour* ***not in*** *Processed* **then**
                $Queue \leftarrow$ *Push Neighbour*
            **end**
        **end**
        $Processed \leftarrow$ *Push Neighbour*
    **end**
**end**
```

The quality and accuracy largely depends on the choice of seed point(s) and the threshold. With this we have a few different approaches to consider:

- Create a cloud segment by using the brightest pixels as seed point(s).
- Create a clear, non-cloud water segment by using the least brightest pixels as seed point(s).
- Use our ship pixel as the initial seed point and grow based on a cloud threshold.

The first approach to be considered is growing our region from the brightest pixels to create a cloud segment. If the ship pixel falls within the region it would be considered obscured. This method introduces some problems however: This would indeed work for images with large, uniform dense clouds, but as we've seen earlier, clouds may take ribbon-like and scattered forms. The brightest pixels is also not guaranteed to be clouds, and as such we cant guarantee that the seed point is a cloudy pixel with our current information.

The second approach is the inverse of our first proposal: Finding clear pixels and checking if the boat does not fall within the segment. We can with a high certainty guarantee that the pixel is water, but this method also sees the same problems. Due to the possibility of scattered clouds our seed points are not guaranteed to produce accurate segments in all scenarios. Choosing several seed points a certain distance away from each other could possibly relieve some of these concerns.

Our final approach involves using the ship pixel as a seed point and growing a region based on a cloud threshold. This alleviates some of the problems we've encountered in the other methods. Since we just want to know if the ship is obscured or not, we don't have to consider the form of the clouds, and we don't have the problem of finding appropriate seed points. The resulting segment will either be the ship itself, due to the high reflectancy property most ships exhibit, or a larger segment of the cloud covering the ship. However, due to the small error margin in the positional accuracy we cannot in all cases guarantee that our seed point is the ship itself. In the scenario of the seed point being non-cloud/water pixel the result will be an empty segment, which will indicate a cloud-free ship. This may cause false positives in the scenario where the ship is unobscured, but the reported position drifts to a nearby cloud.

All approaches provides their own specific problems, but using the ship as the seed point will, except for certain edge-cases, provide the most accurate segments for our use. We can use the threshold values and filtering methods from the previous solution as our threshold method when generating the segments. This proved highly accurate, and should be a good fit for our region growing implementation.

For our implementation we will be using 8-connectivity when generating neighbouring pixels (Figure 3.12). In order to calculate the final result, we need to compare the size of the mask with a given threshold. Since ships are usually relatively small, we say that segments consisting of less than 50 pixels are unobscured (false), while any segment exceeding 50 pixels are obscured (true). Examples of this can be seen in figure 3.13.

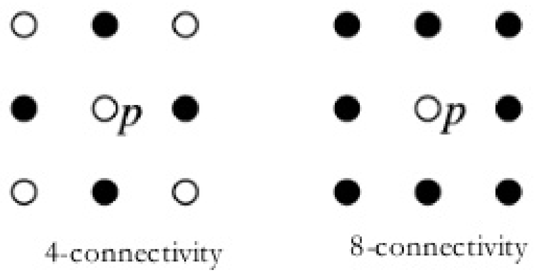


Figure 3.12: 4-connectivity and 8-connectivity when generating neighbouring pixels

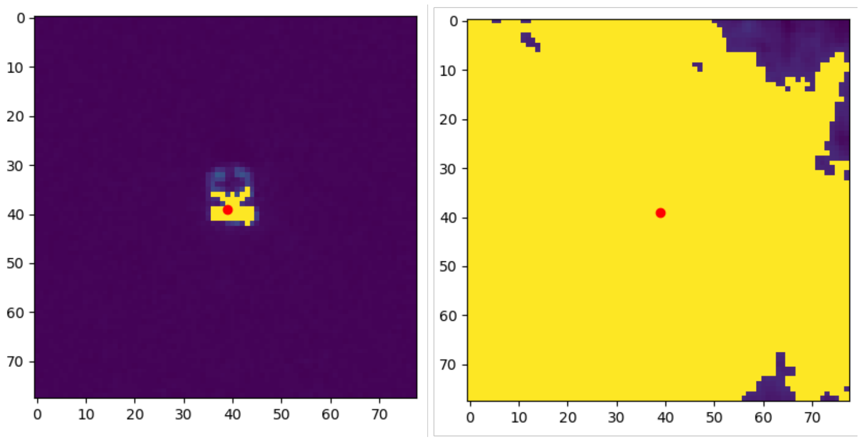


Figure 3.13: Ship segment (left) and a cloud segment (right), where the seed point is denoted with a red dot.

We do note that this way of calculating the size of the mask is not ideal for situations where the ship is obscured and near an edge or a corner. This approach would not work in the case of a ship being covered by the edge of a cloud appearing from outside the boundaries of the images, as this could lead to the resulting segment being below the size threshold and yet be a cloud. These specific cases are not covered in our data set, and for our work we can eliminate this by mosaicing corresponding images together to restore information from outside the original image. This might not be ideal for images where this is not possible, so alternative ways to validate the resulting mask is a topic for discussion. A further discussion of this can be found in chapter 4.2.

3.3.1 Evaluation

To evaluate our segmentation method we will be using the same set of ships as before:

- Region growing segmentation: 227 of 227 ships correctly identified in total. (100%).
  - Obscured ships: 40 of 40 correctly identified
  - Unobscured ships: 187 of 187 correctly identified

Our region growing implementation is able to correctly label all images in our data set, making it an improvement over our previous method. The computional time needed for this method is not ideal howerver. Depending on hardware, image size and cloud cover, one image can take upwards of 1 minute to process in a worst case scenario. For cases where there are little to no clouds, the process is fast. Although this method is our most accurate so far, it is a trade-off in terms of speed.

3.4 Deep Learning

Our final approach is examining the possibility of a Deep Learning model for labeling images. There have been a growing interest in the application of Deep Learning on EO data, especially when it comes to cloud masking and detection. Due to the unpredictable nature of clouds, we aim to eliminate any guesswork and unpredictability needed by delegating it to a deep learning model. Deep learning networks excel in finding patterns and meaning in large data sets, which in theory should make such an approach more generalizeable than our previous methods. However, there have been uncovered some limitations to this approach, outlined in chapter 3.4.1.

In designing a method for our objective we need to decide on what the goal of the network is. In order to properly detect where the clouds are in the image, a standard image classification or object detection approach would not be sufficient. A promising approach that has been growing in popularity the recent years is **Semantic Segmentation**, which is a pixel-level classification method. Semantic segmentation is the task of clustering parts of image together which belong to the same object class [18]. This method of segmentation should give us fine-grained precision in the same vein as F-Mask and 1C masks.

The input is an image, and the label is the corresponding mask. Since our problem is a binary classification problem, our labels will be cloud masks. The network then trains on the input data and finally produces an output mask.

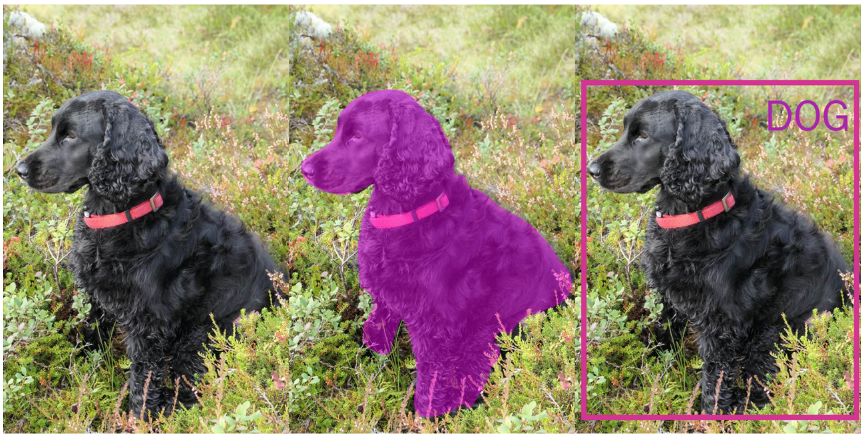


Figure 3.14: Difference between segmentation (Middle) and Object Detection (Right). The semantic segmentation mask is denoted with a pink mask, while the object detection bounding box is shown with a pink box together with the identified class.

3.4.1 Limitations

As of this papers date there exists no publicly available data set of 1C clouds with corresponding cloud masks. There exists segmentation data sets for other satellite platforms, such as DSTL[19], Airbus[20] and a Landsat-8, but none of which are directly applicable to our work. Previous semantic segmentation work has been done on Sentinel-2 2A images, but the authors found it necessary to create their own training images and masks by hand [21]. There is work being done on creating a public data set of manually curated masks, but this is unfortunately not available at the time of writing [22].

Therefor we find it necessary to generate our own training data and masks. As previously seen, F-mask is the most accurate masking solution covered in this work, although prone to falsely labeling pixels as clouds. 1C masks would be far too conservative for us to use as a mask, as for our training set we rather want ships to be falsely labeled as obscured rather than introducing falsely labeled unobscured ships.

As the model will only ever be as accurate as the data it trained on, our best hope for this model is to approximate the F-Mask masks. This even includes its faults, such as incorrectly labeling pixels, but with a large enough data set combined with manually ’cleaning’ the data set of especially inaccurate masks we aim to offset these unfortuante properties as much as possible.

Since F-Mask is not accurate enough in comparison with our other proposed methods, this method should be seen as a **"proof-of-concept"**, to prove the viability of such an approach given a large data set of high quality in future work.

3.4.2 Data set

For our image input data we will be using 1C images taken from three different scenes off the coast of Norway. Most CNNs are built with the expectations of three channel RGB input. Although our data has the possibility of 13 channels, we will only be using the red, blue and green spectral bands for the sake of simplicity in this work.

For our masks we will be computing the masks on our three scenes using F-Masks. Since the resulting masks contains 5 different classes (Pixel values), we need to pre-process our masks to fit with our model. F-Mask comes with the following pixel classifications:

- 0 = clear land pixel
- 1 = clear water pixel
- 2 = cloud shadow
- 3 = snow
- 4 = cloud
- 255 = no observation

To reduce this to a binary classification mask, we set all pixels with a value of 4 to 1, and all other pixels to 0, so we have a cloud mask represented by the value 1.

We utilize a sliding window across both images to produce $128x128$ pixel output images with corresponding masks. It is important that the dimensions of the images are even and a power of 2 because of the downsampling and upsampling process in our network (Detailed in chapter 3.4.3. Since 1C images are provided with a data type of uint16 (0-65535), we process each image as they are created by normalizing to [0-255] and converting the data type to float32. Float is useful for our model because of the improved precision, especially when normalizing to the range of [0-1] in the training process.

As seen earlier in this work, F-mask has the unfortunate property of incorrectly masking some ships as clouds, which would be greatly detrimental to our model accuracy. We correct this by manually cleaning our data set for these cases, as well as removing sets of images containing potential noise, such as land cover and urban cityscape. A small amount of noise could in some situations be advantageous [23], but could in our case introduce incorrect masks. Therefor we aim to reduce the potential noise as much as possible.

This leaves us with a data set of around 1000 images. We can apply augmentations to synthetically increase the data set, giving us a theoretical increase in training images. This not only gives us a larger data set to work with, but augmenting images will also help our model avoid overfitting. For this we will be applying random augmentations with a probability P for each image during training. Our augmentations are horizontal and vertical flips, and *Elastic Transform*[24] (illustrated in figure 3.15).

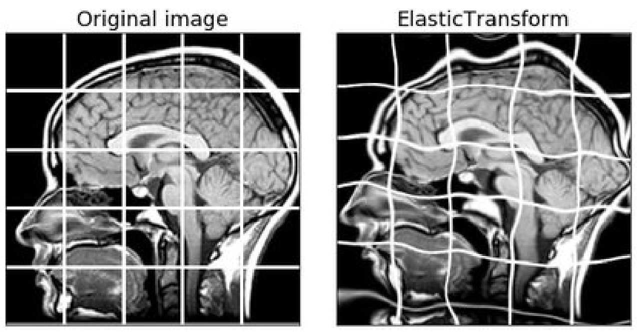


Figure 3.15: Example of elastic transformation in image augmentation [25]

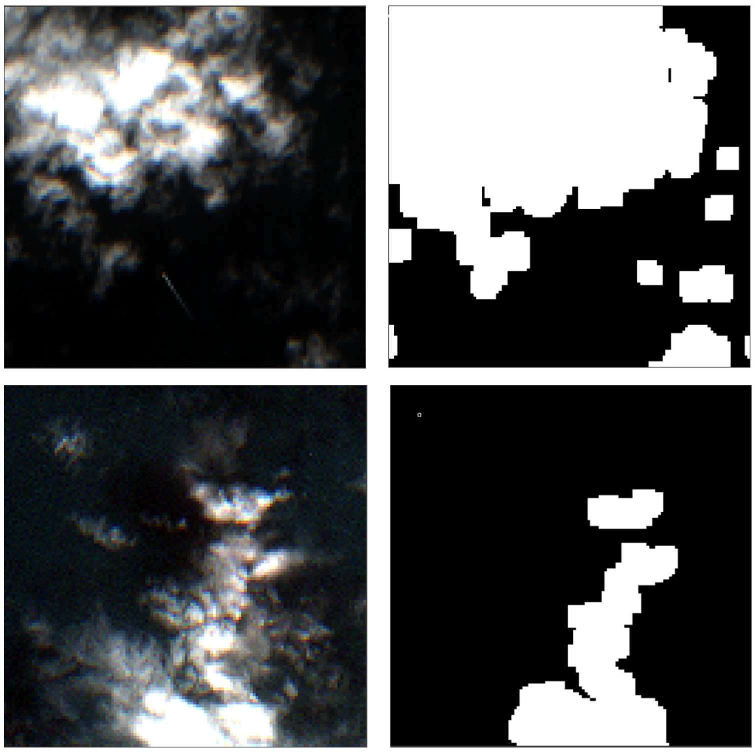


Figure 3.16: Two examples of training set input images with their corresponding processed masks

3.4.3 Network architecture

There exists several networks used for semantic segmentation, such as ResNet, PSPNet and DeepLab, but for our purposes we will be using the U-net architecture [26]. U-net is shown to work well on segmentation tasks for EO data [27], as well as being advantageous when dealing with smaller data sets [28] [26]. U-net was initially developed for segmentation tasks in biomedical images, but has shown great promise in general segmentation tasks and is a widely used architecture today.

For our implementation we will be using PyTorch [29].

U-net is a **Fully Convolutional Network** using an *Encoder-Decoder struc-*

ture, meaning it consists only of convolution layers. The max-pooling layers of a CNN is used for downsampling, which can be seen as an *encoding* process through a stack of pooling and convolutional layers. This process greatly reduces the size of the image and helps the CNN understand what the features in the image, but trades off spatial information, i.e where the features are in the image. U-net uses an up-sampling process through Transposed Convolution that can be seen as a *Decoder* process, restoring the size and spatial information that was removed during the encoding phase. An overview of the U-net network architecture is shown in figure 3.17

The encoding path uses convolution layers and max pooling layers to halve the size of the image and double the number of feature channels for each step.

The decoding path consists of convolution layers up-sampling the feature maps from the encoding phase, halving the number of feature channels for each step. We also concatenate the feature map from the corresponding encoding convolution layer, which are often called the *skip connections* or Residual connections. These connections gives the up-sampling convolutional layers the information needed to undo the max pool downsampling from the encoding path. Each convolution layer in the network is followed by a ReLu. For this implementation we will be using a Leaky ReLu with a leakage factor of 0,01 (Appendix B.3).

The final convolution layer reduces the number of feature channels down to 1. For this last layer we will be using a *Sigmoid activation* (Appendix B.4) resulting in a 128x128 pixel mask with each pixel classified with a value between [0-1]. These values can be seen as a "cloud probability" ranging from 0 (no cloud) to 1 (Cloud). The threshold for what values constitutes a segment can be seen as another hyperparameter. For this work we will be using 0.5 as the threshold for our cloud segment.

We will also be extending the original U-net paper by including batch

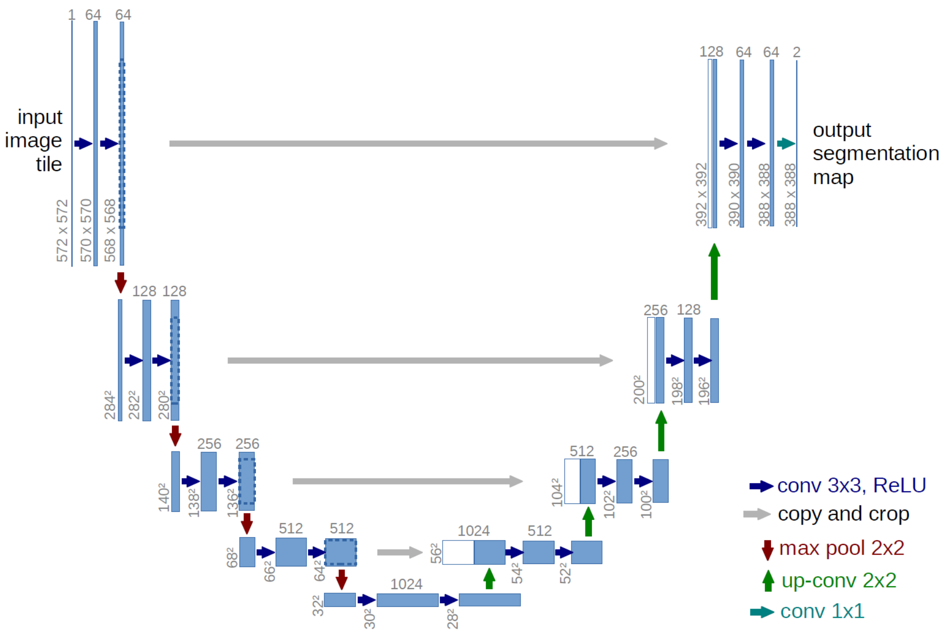


Figure 3.17: Overview of the general U-net architecture. Skip/residual connections are shown with a grey arrow.

normalization and a dropout layer in the middle block. Batch normalization is a technique where the output from each layers in the network is normalized using trainable parameters, which is updated for each batch passing through the network. This is also useful for combating a phenomenon called "Exploding Gradients". This occurs when the layer outputs are not normalized, which can cause the difference between the output and the desired output to grow very large, which can cause saturation in the network when back-propagating the large loss sum.

A dropout layer sets all activations in the layer to zero with a random probability P of activating each forward pass. Although there are discussions regarding the effectiveness of dropout for convolutional layers, it has been shown to an effective technique for some CNNs when used on a deep layer [30] [31] [32].

Loss function

For our loss function we will be using Binary Cross Entropy (BCE) loss, which is a good fit for our [0-1] probability values resulting from the Sigmoid activation. The output from the BCE is a value between [0-1], where lower equals better. BCE can be simplified as:

$$L = -\mathbf{y} \cdot \log(\hat{\mathbf{y}})$$

Where y is our prediction and $\hat{y}$ is the ground truth.

Optimizer

Our choice of optimizer for the network will be Adam, which is shown to work better than the widely used Stochastic Gradient Descent [33], and can be seen as a combination of AdaGrad and RMSProp. Adam computes individual adaptive learning rates each parameter during training and utilizes a *Momentum* technique to further optimize the parameters. Momentum is when a small fraction of previous updates in the network are added to the current step. Repeated updates in one direction will build up a momentum in that direction, increasing the rate at which the parameters change. This will give faster convergence to the minimum of the loss function.

Accuracy

In order to check our model accuracy during the evaluation phase of the training, we need to include a function to calculate the accuracy of the output mask on the given validation set. For this we will be using **Jaccard similarity coefficient**. Jaccard similarity coefficient calculates the *Intersection Over Union* (IoU) between two sets. Essentially this means that we calculate the similarity between our predicted output mask and our target mask. Jaccard is defined as:

$$ioU(A,B) = \frac{|A \cap B|}{|A \cup B|}$$

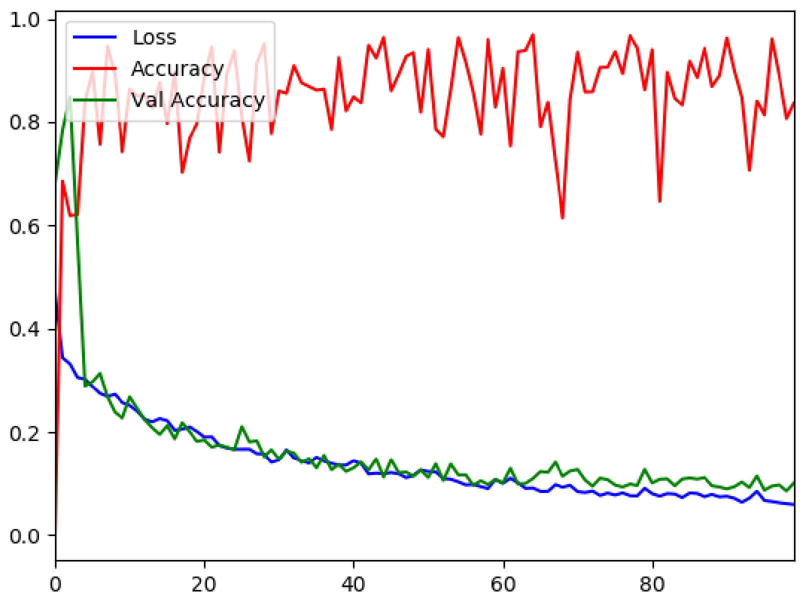
Where A and B are the output mask and target mask respectively. This gives us a value between 0 and 1 that denotes the similarity between the masks, where higher equals better.

3.4.4 Evaluation

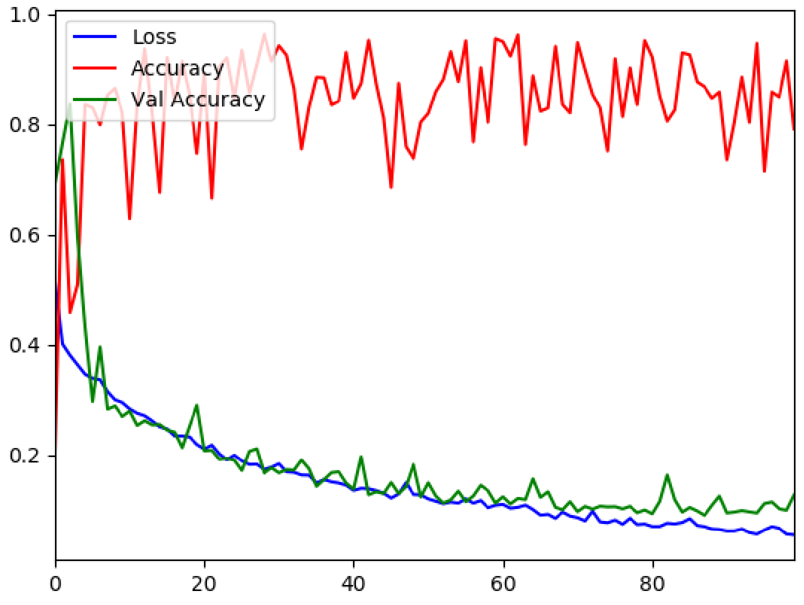
Our training set of roughly ~1000 images will be divided into a test-set and a validation-set randomly, where the split for each set is 67% and 33% respectively. Each input image is normalized to the range of [0-1], with a probability of 50% of being applied a random augmentation.

For our Adam optimizer we will be using a low learning rate of 0.0001, and training will be done over 100 epochs. This will make the training process take longer until convergence, but should yield more accurate results. Training will be done with batch sizes of 16 and 32 for comparison.

The results of our training (Shown in figure 3.18) shows some interesting properties. Although our validation loss converges around ~0.1, the accuracy oscillates after the 10 epoch mark. There could be multiple reasons for this, but it is safe to assume a larger training and validation-set of good quality could smooth this out. Further tweaking of the different hyperparameters could also yield a more stable result, such as a lower training rate, bigger batch sizes and a larger validation set. Our model does show signs of overtraining after the 60 epochs mark, where training loss continues to decrease while our validation loss beings to converge. Thus an early stop between 50 and 60 epochs should yield the



(a) Batch size of 32.



(b) batch size of 16.

Figure 3.18: Training loss (Blue), validation loss (green) and Accuracy (red) for different batch sizes over 100 epochs.

best model.

With this model we're consistently able to get a best accuracy (similarity) score of ~0.98 on our validation set, with a mean accuracy of ~0.90 for our epochs. For fine-grained segmentation purposes this might not be adequate, but for our objective this accuracy should be able to correctly label most images. Figure 3.19 shows a predicted mask created by our model compared with the corresponding ground truth mask.

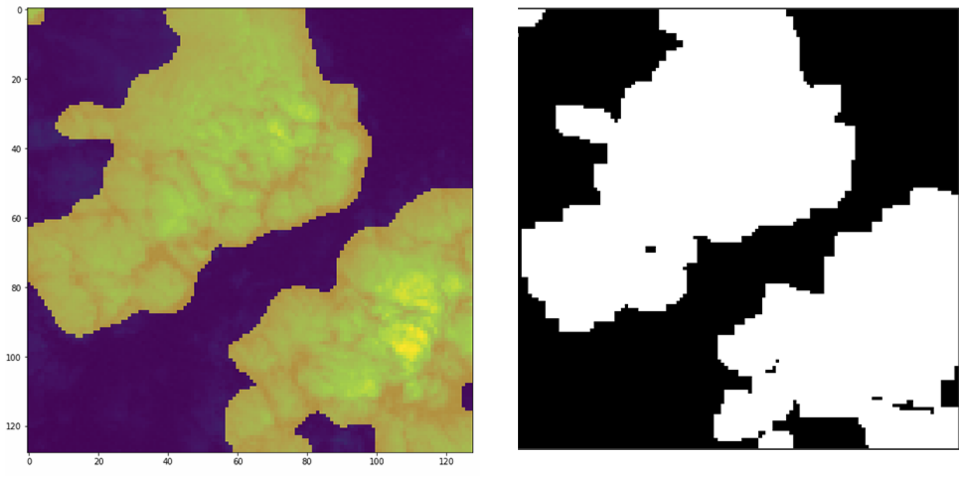


Figure 3.19: Predicted mask shown in yellow (left) and the corresponding ground truth mask shown in white (right)

To get a better grasp of the actual accuracy of our model in relation to our objective, we can use our model on the 221 ship images we have used as validation so far. For this we will be using the same technique as we did to evaluate F-Mask, looking at adjacent pixels near the ship coordinates.

Since our model is trained on F-Mask masks it is reasonable to assume that the model will closely follow the same results as F-Mask (90.3% of all images correctly labeled):

- Semantic Segmentation: 214 of 227 ships correctly identified in total. (94.27%).

- Obscured ships: 40 of 40 correctly identified
- Unobscured ships: 174 of 187 correctly identified

Results shows that our segmentation model actually achieves better results than F-Mask did, by correctly labeling two new ships as unobscured that F-Mask previously incorrectly labeled as obscured. Upon closer inspection we see that this is due to the slight error margin in our model, causing the masks to contract and be slightly off from the F-Mask masks. In a way we can see this as the loss in the model actually making the model an improvement over F-Mask. Example of a previously incorrectly labeled image is seen in figure 3.20.

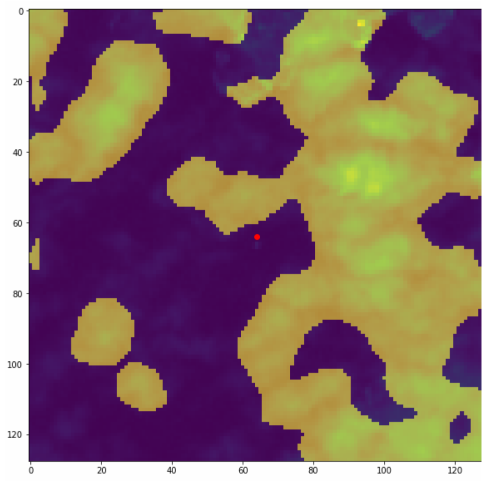


Figure 3.20: Image correctly labeled by our segmentation model that was previously labeled as obscured. Predicted mask is shown in yellow.

Training of the model is a rather time consuming and computationally expensive process, requiring high-end hardware. However, once the model is trained, it is exceptionally fast and scales very well with size and cloud cover percentage. This makes this solution the ideal intersection between speed and accuracy given a model trained on highly accurate training data.

Chapter 4

Conclusions, Discussion, and Recommendations for Further Work

4.1 Summary and Conclusions

In the introduction we presented three main objectives:

1. Develop different methods for validating and correctly labeling ship images for use in ship detection.
2. Evaluate and compare the developed solutions.
3. The methods should be fast and optimized for ship level labeling.

In this section we will be summarizing the methods proposed in this work, compare them and evaluate whether we met our objectives or not. Over the course of this work we developed three different methods for labeling images, each with different approaches to the same problem.

In Chapter 2 we introduced the background theory needed to understand the work and our motivation. Here we presented the qualities for a good data set, as well as what defined an accurate label, in correspondence with out sub-objective presented in the introduction: *Define what makes an accurate label.*

In Chapter 3 we first narrowed down our problem area to smaller images containing boats, presenting a data set of cut images containing boats. This process was in accordance with one of the sub-objectives introduced in the introduction: *Generate images used for developing and evaluating the methods.* By combining the coordinates reported in AIS messages with the corresponding Sentinel-2 images and extracting ship images we have also reduced the domain for each method, making them very fast and optimized in accordance with our third main objective: *The methods should be fast and optimized for ship level labeling.* By further examining the speed and computational time and power needed, we can safely say that our methods satisfy this objective.

Our first proposed method was a fuzzy logic reasoning approach for labeling the ship images. Here we explored the different properties the clouds exhibited in the different spectral bands as well as experiments used to determine a cloud probability threshold. During testing we saw that different parameters impacted had an impact on the results. Using balanced decreasing function we were able to get an accuracy score of **99.16%**, where one obscured ship was falsely labeled as unobscured. By using a more aggressive weighting of pixels close to the ship our overall accuracy dropped to **97.47%**, but we were able to correctly label all obscured ships.

During the evaluation we also saw the different flaws in the existing clouds masks: F-Mask and 1C cloud masks. Both solutions exhibited opposite properties: 1C cloud masks were highly inaccurate and conservative, whereas F-Mask was more accurate but prone to falsely labeling pixels as clouds.

Secondly we presented an image segmentation method based on the region growing algorithm. The goal was to eliminate the guesswork and more volatile factors utilized in our fuzzy logic approach. By using what we learned from the previous experiments we had a solid groundwork for our algorithm, which utilized threshold values and cloud properties previously uncovered. By choosing to use the ship as our seed point we were able to eliminate a lot of the guesswork required, as well as making sure our solution worked optimally despite the volatile and random nature of clouds. With this method we were able to get an accuracy score of **100%**, which is the best accuracy score in this work.

Lastly we introduced a deep learning method for semantic segmentation of clouds. The goal with this approach was to completely eliminate any guesswork by offloading it to a neural network. Through the limitations uncovered we presented a data set based on the F-Mask masks, which after the analysis proved to not be ideal. The network was built with a modified U-net architecture, and trained for 100 epochs. Results showed that the model had a validation loss of **~0.1**, and a best validation accuracy of **~0.98**. Evaluating this approach on our data set gave us an accuracy score of **94.27%**, where our model incorrectly labeled 13 ships as obscured when they were in reality unobscured.

A summary of all the collected results are shown in table 4.1:

Results of all proposed methods	
Method	Best Accuracy
Fuzzy logic reasoning	99.16%
Image segmentation	100%
Semantic segmentation	94.27%

Table 4.1: Results table

In conclusion we can see that we have met our first main objective: *Develop different methods for validating and correctly labeling ship images for use in ship detection.* We have proposed three different methods, all of which yielded a better labeling accuracy on a ship level than the preexisting masking solutions examined in this paper. All methods are viable for use in generating and labeling training set, with **Region growing image segmentation** yielding the best results for our objective. There were some concerns uncovered in regards to images with non-static ship placements, in which case our fuzzy logic reasoning approach is a good alternative. For our motivation we do prefer to use parameters yielding slightly lower overall accuracy, but a high accuracy on obscured ships.

Lastly, our deep learning semantic segmentation approach proved quite successful, but due to the lack of good training data makes it the inferior choice at the moment on a pure accuracy level. However, the potential of this solution in further work is covered in Chapter 4.3.

For our second main objective: *Evaluate and compare the developed solutions*, we will be presenting a more in-depth discussion around our methods in Chapter 4.2.

4.2 Discussion

When evaluating our proposed methods we see that the results differ in terms of mislabeling obscured ships or unobscured ships. As covered in the background chapter, the quality of the machine learning model is highly dependant on the quality of the training set. As such it is an important distinction whether a method is more prone to incorrectly label obscured ships as unobscured or vica versa. For the purposes of object detection, an obscured ship labeled as unobscured will negatively affect the model. Mislabeling an unobscured ship as obscured would not negatively affect the model (Given that we exclude images marked as obscure), but rather give a slightly smaller and potentially less varied training set.

As we saw in the case of the fuzzy logic reasoning method, different parameters yielded different results and different mislabeling. It is arguably a far better choice to minimize the potential of mislabeled images entering our training set than have a larger training with incorrect classifications. A smaller but more accurate data set can easily be augmented to increase in size, as seen in our semantic segmentation method. The fuzzy logic approach is also quite dependant on the pixel values present in the image, as it uses the values to calculate a final score. Seeing how unpredictable and varied EO data can be, it is perhaps the least stable approach in our work, and as such should be tested more vigorously with more varied data in order to properly determine the long-term potential of this approach.

The Region growing algorithm is not dependant on as many variables, only a binary understanding of the pixels in terms of the pixel exceeding the threshold or not. Region growing image segmentation also faced some theoretical limitations in the case of ships appearing near the edges or corners of the image while being obscured by a cloud appearing from outside the borders of the image. This is due to the way we both generate the cloud segment as well as how we calculate the mask size. Seeing as machine learning models greatly benefit from variation in the training set, it is likely that we want to use our proposed methods to label images with ships in random, non-static positions. Although we did not manage to test our methods on an edge case described here, it is important to discuss the potential that it could happen. One approach to tackling this problem is a more sophisticated way of calculating and verifying the resulting cloud segment. A potential solution would be to not only look at the size in terms of the number of pixels in the segment, but looking at the shape and placement as well. If the segment is localized near a corner or edge and the shape indicates extending outside image, it could signal that it is a cloud appearing from outside the boundaries of the current image. Another solution would be to cut the image slightly larger, with the ship in center. After labeling the image with the region growing algorithm, one could proceed to cut the image into smaller sub-images. This ensures that all the resulting sub-images are indeed correctly labeled.

The error margin in the positional accuracy of the ships is also another point of potential concern. Timestamp from the AIS message and the timestamp of the Sentinel-2 data product will not necessarily line up perfectly, which can cause further positional drift. Inaccurate coordinates can cause problems in certain edge cases, where the state of the actual ship position differs from the reported position. It is, however, possible to account for this drift by extrapolating the actual position using the direction the ship is headed together with the speed and timestamp [34].

For our semantic segmentation model U-net showed great potential, achieving good accuracy and a low loss. Although the training set was not ideal, the network architecture shows great potential for further work. Our augmentation techniques also helped greatly in reducing overfitting, and made the model generalize better, which is a good sign that the task of cloud segmentation is possible without an unrealistic data set size. This method also shows promise not only in image labeling, but could be extended to work on a scene level (i.e segmenting whole Sentinel-2 scenes). Although processing whole scenes during training is unrealistic due to memory and hardware constraints, using these smaller images could very well train a model to work on any 1C scene. Extending this method should not be any different than our existing approach, and shows great potential for use in general cloud segmentation on EO-data. Extending the model further we could also introduce multiclass segmentation, allowing our model to not only classify cloud pixels, but vegetation, snow and more.

In conclusion, our region growing approach is currently the optimal method for automatic labeling for use in ship detection. Fuzzy logic reasoning is a viable approach when used with the correct parameters, but the trial-and-error that is potentially required for this to reach the desired accuracy makes this method the least desirable method for automatic ship labeling.

The U-net architecture and deep learning approach shows great potential for our objective, as well as in a broader EO-context, and is the logical continuation of our work.

4.3 Recommendations for Further Work

For our objective in labeling ship images, we have found our work to be satisfactory, but our semantic segmentation model shows potential for further improvements on both labeling and general cloud segmentation. In this section we will cover what we recommend for further work within this topic.

4.3.1 Improved data set for semantic segmentation

The biggest bottleneck for our segmentation model is the quality and availability of the data set needed to train our model. As previously mentioned there is work being done on curating such a data set which would be a perfect fit for our purposes. Alternatively, manually masking images would also be a good alternative, albeit a time-consuming and arduous task. Our method is only concerned with images over water, so a bigger, more diverse data set could easily be introduced to expand the domain of the model as well.

4.3.2 Extending the labeling problem further

In this work we have only focused on clouds as the main concern when labeling images for use in machine learning. However, there are other factors that could affect the image, causing it to not be viable as training images when detecting ships. This main area of interest for further work would be **cloud shadows**. Large, opaque clouds can cause shadows to completely obscure ships in worse-case scenarios. For further work we recommend extending the proposed solutions to also take into account the expected reflectance of the ship. If missing, and the area is found to be of low luminous intensity, the ship could be marked as obscured. For our deep learning model incorporating the cloud shadow mask and extending the model past a binary classification is the recommended next step for this goal.

4.3.3 Further extending the U-net architecture

A logical next step to further increase the accuracy of our semantic segmentation model would be to extend the U-net architecture to accept more bands for each input-image. For each image we have upwards of 13 spectral bands for our disposal all of which could further help the network infer the cloud masks. A closer analysis on what combination of bands yields the best result should be the first step, after which the network can be modified to accept input-images with multiple channels past RGB.

Appendix A

Acronyms

AIS	Automatic Identification System
ANN	Artifical Neural Network
BCE	Binary Cross Entropy
CNN	Convolutional Neural Network
DAG	Directed Acyclic Graph
EO	Earth Obersveration
ESA	European Space Agency
GDAL	Geospatial Data Abstraction Library
GML	Geographic Markup Language
IoU	Intersection Over Union
ReLu	Rectified Linear Unit
TOA	Top-Of-Atmosphere

Appendix B

Formulas and functions

B.1 Euclidian Distance

The distance between two points in the plane with coordinates (x, y) and (a, b) is given by: $dist((x, y), (a, b)) = \sqrt{(x-a)^2 + (y-b)^2}$

B.2 Max pooling

Max pooling defines a NxN matrix that slides across the image, selecting the max value from each submatrix in the window.

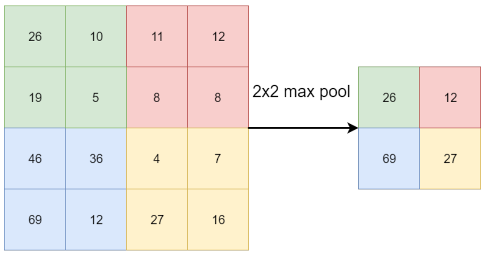


Figure B.1: Max pooling operation with a 2x2 filter with stride = 2

B.3 Leaky ReLu

A leaky ReLu is a Rectified Linear Unit with a small negative slope for values below 0.

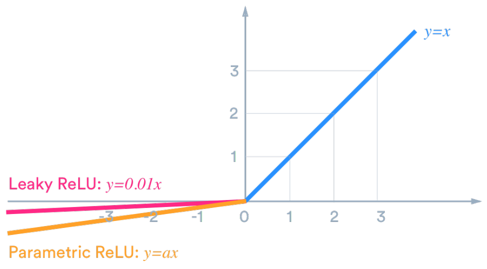


Figure B.2: Leaky ReLu function

B.4 Sigmoid

A sigmoid function is defined by the formula: $S(x) = \dfrac{1}{1+e^{-x}}$

Bibliography

[1] Bianca Hoersch et al. *science-and-applications-with-sentinel-2 @ www.journals.elsevier.com*. URL: https://www.journals.elsevier.com/remote-sensing-of-environment/call-for-papers/science-and-applications-with-sentinel-2.

[2] Adrian Tofting and Lars Henrik Berg-Jensen. 'Detection, A Big Data Approach to Generate Training Data for Automatic Ship'. PhD thesis. Norwegian University of Science and Technology, 2018, p. 109.

[3] ESA GUIDES. 'SENTINEL-2 User Handbook Sentinel-2 User Handbook SENTINEL-2 User Handbook Title Sentinel-2 User Handbook Issue 1 Revision 1 SENTINEL-2 User Handbook'. In: 1 (2015), pp. 1–64. DOI: 10.13128/REA-22658. URL: https://earth.esa.int/documents/247904/685211/Sentinel-2_User_Handbook.

[4] European Space Agency. *Sentinel-2 MSI Overview*. URL: https://earth.esa.int/web/sentinel/user-guides/sentinel-2-msi/overview.

[5] Shunlin Liang, Xiaowen Li and Jindi Wang. 'Learn more about Spatial Resolution A Systematic View of Remote Sensing'. In: (2012). DOI: https://doi.org/10.1016/B978-0-12-385954-9.00001-0. URL: https://www.sciencedirect.com/topics/earth-and-planetary-sciences/spatial-resolution/pdf.

[6] *Sentinel-2A SatelliteSensor | Satellite Imaging Corp.* URL: https://www.satimagingcorp.com/satellite-sensors/other-satellite-sensors/sentinel-2a/.

[7] European Space Agency. *Copernicus Open Access Hub.* 2019. URL: https://scihub.copernicus.eu/.

[8] *Complete Sentinel-2 Archives Freely Available to Users | AWS Government, Education, &amp; Nonprofits Blog.* 2019. URL: https://aws.amazon.com/blogs/publicsector/complete-sentinel-2-archives-freely-available-to-users/.

[9] *Level-2A Algorithm Overview.* 2019. URL: https://earth.esa.int/web/sentinel/technical-guides/sentinel-2-msi/level-2a/algorithm.

[10] OGC. 'OGC® Geography Markup Language (GML) — Extended schemas and encoding rules'. In: *OpenGIS Recommendation Paper* (2010), p. 595. URL: http://www.w3.org/TR/ws-arch/.

[11] *Level-1C Cloud Masks - Sentinel-2 MSI Technical Guide - Sentinel Online.* 2019. URL: https://sentinel.esa.int/web/sentinel/technical-guides/sentinel-2-msi/level-1c/cloud-masks.

[12] *Automatic Identification Systems (AIS).* 2019. URL: http://www.imo.org/en/OurWork/safety/navigation/pages/ais.aspx.

[13] Andrej Karpathy. *Convolutional Neural Networks for Visual Recognition.* 2015. URL: http://cs231n.github.io/convolutional-networks/.

[14] Stuart J Russel and Peter Norvig. *Aritical Intelligence: A Modern Approach.* 3rd. Prentice Hall, 2009, p. 1152. ISBN: 0-13-604259-7.

[15] Valerie Sessions and Marco Valtorta. 'The Effects of Data Quality on Machine Learning Algorithms.' In: 2006, pp. 485–498.

[16] Zhe Zhu, Shixiong Wang and Curtis. E Woodcock. 'Improvement and expansion of the Fmask algorithm: cloud, cloud shadow, and snow detection for Landsats 4–7, 8, and Sentinel 2 images'. In: *Remote Sensing of Environment* 159 (Mar. 2015), pp. 269–277. ISSN: 00344257. DOI: 10.1016/j.rse.2014.12.014. URL: https://linkinghub.elsevier.com/retrieve/pii/S0034425714005069.

[17] Zhe Zhu and Curtis E. Woodcock. 'Object-based cloud and cloud shadow detection in Landsat imagery'. In: *Remote Sensing of Environment* 118 (Mar. 2012), pp. 83–94. ISSN: 00344257. DOI: 10.1016/j.rse.2011.10.028. URL: https://linkinghub.elsevier.com/retrieve/pii/S0034425711003853.

[18] Martin Thoma. 'A Survey of Semantic Segmentation'. In: (Feb. 2016). URL: http://arxiv.org/abs/1602.06541.

[19] *Dstl Satellite Imagery Feature Detection*. 2017. URL: https://www.kaggle.com/c/dstl-satellite-imagery-feature-detection.

[20] Airbus. *Airbus Ship Detection Challenge*. 2018. URL: https://www.kaggle.com/c/airbus-ship-detection.

[21] Cheng-Chien Liu et al. 'Clouds Classification from Sentinel-2 Imagery with Deep Residual Learning and Semantic Image Segmentation'. In: *Remote Sensing* 11.2 (Jan. 2019), p. 119. ISSN: 2072-4292. DOI: 10.3390/rs11020119. URL: http://www.mdpi.com/2072-4292/11/2/119.

[22] Sinergise. *crowdsourcing eo training datasets to improve cloud detection*. 2017. URL: https://medium.com/sentinel-hub/crowdsourcing-eo-training-datasets-to-improve-cloud-detection-4ae134267f23.

[23] Chris M. Bishop. 'Training with Noise is Equivalent to Tikhonov Regularization'. In: *Neural Computation* 7.1 (Jan. 1995), pp. 108–116. ISSN: 0899-7667. DOI: 10.1162/neco.1995.7.1.108. URL:

http://www.mitpressjournals.org/doi/10.1162/neco.1995.7.1.108.

[24] P.Y. Simard, D. Steinkraus and J.C. Platt. 'Best practices for convolutional neural networks applied to visual document analysis'. In: *Seventh International Conference on Document Analysis and Recognition, 2003. Proceedings.* Vol. 1. IEEE Comput. Soc, pp. 958–963. ISBN: 0-7695-1960-1. DOI: 10.1109/ICDAR.2003.1227801. URL: http://ieeexplore.ieee.org/document/1227801/.

[25] Alexander Buslaev et al. 'Albumentations: fast and flexible image augmentations'. In: (Sept. 2018). URL: http://arxiv.org/abs/1809.06839.

[26] Olaf Ronneberger, Philipp Fischer and Thomas Brox. 'U-Net: Convolutional Networks for Biomedical Image Segmentation'. In: (May 2015). URL: http://arxiv.org/abs/1505.04597.

[27] Ruirui Li et al. 'DeepUNet: A Deep Fully Convolutional Network for Pixel-level Sea-Land Segmentation'. In: (Sept. 2017). URL: http://arxiv.org/abs/1709.00201.

[28] Alexander V. Buslaev et al. 'Fully Convolutional Network for Automatic Road Extraction from Satellite Imagery'. In: (June 2018). URL: http://arxiv.org/abs/1806.05182.

[29] *PyTorch*. 2019. URL: https://pytorch.org/.

[30] Haibing Wu and Xiaodong Gu. 'Towards Dropout Training for Convolutional Neural Networks'. In: (Dec. 2015). DOI: 10.1016/j.neunet.2015.07.007. URL: http://arxiv.org/abs/1512.00242http://dx.doi.org/10.1016/j.neunet.2015.07.007.

[31] Yarin Gal and Zoubin Ghahramani. 'Bayesian Convolutional Neural Networks with Bernoulli Approximate Variational Inference'. In: (June 2015). URL: http://arxiv.org/abs/1506.02158.

[32] Nitish Srivastava et al. 'Dropout: A Simple Way to Prevent Neural Networks from OverfittingTitle'. In: *ournal of Machine Learning Research 15 (2014) 1929-1958* (2014), pp. 1930–1958.

[33] Diederik P. Kingma and Jimmy Ba. 'Adam: A Method for Stochastic Optimization'. In: (Dec. 2014). URL: http://arxiv.org/abs/1412.6980.

[34] Van-Suong Nguyen, Nam-kyun Im and Sang-min Lee. 'The Interpolation Method for the missing AIS Data of Ship'. In: *Journal of Navigation and Port Research* 39.5 (Oct. 2015), pp. 377–384. ISSN: 1598-5725. DOI: 10.5394/KINPR.2015.39.5.377. URL: http://koreascience.or.kr/journal/view.jsp?kj=GHMHD9&py=2015&vnc=v39n5&sp=377.