UNIVERSITÄT HAMBURG

MASTER’S THESIS

Using Collocated Satellite Data for Ice Water Path Retrieval - A Reimplementation of SPARE-ICE

Author:
John MRZIGLOD

Supervisors:
Prof. Dr. Stefan BÜHLER
Dr. Akós HORVATH

Radiation and Remote Sensing
Meteorological Institute

Topic: Using Collocated Satellite Data For Ice Water Path Retrieval - A Reimplementation of SPARE-ICE

ABSTRACT
This thesis is about the reimplementation of SPARE-ICE, an ice water path (IWP) retrieval trained with collocations amongst active and passive remote sensing instruments. The original toolkit was hardly documented, based on an experimental code and slow on large datasets. Furthermore, the retrieval underestimates the IWP at mid-latitudes systematically in comparison to its training target. Hence, I reimplemented it with more efficient workflows and object-oriented programming standards. The collocator, shipped with the new implementation, can be used a standalone toolkit to find collocations between geo-referenced data efficiently. I retrained the retrieval with more training data and additional inputs such as a land sea mask and the standard deviation of the passive radiometer channels. After training, the mid-latitude bias was reduced significantly and the overall performance improved. The reimplemented SPARE-ICE toolkit is publicly available via the python package typhon.

Contents

Chapter 1

Introduction

Ice clouds have a huge impact on the Earth's climate system and global energy budget due to their radiative properties in the short- and long-wave spectra. However, whether ice clouds cool or heat up the atmosphere strongly depends on their composition, height as well on their vertical and horizontal extent. The complexity of the radiative transfer problem and the lack of representative observation data make the accurate modelling of ice clouds for GCMs very challenging. According to Waliser et al. (2009), the Ice Water Path (IWP), a key quantity to measure the accumulated amount of ice particles per atmospheric column, differs in various Global Climate Models (GCM) by several magnitudes. The IPCC classifies our understanding of ice clouds and their climate feedback as *low* (IPCC 2013). Better and more measurements of the IWP might improve the understanding of ice cloud physics. Hence, many efforts were taken to develop retrievals that provide globally covering IWP observations. Since in-situ measurements of ice clouds are not practical and ground-based instruments do not provide data with a global coverage, many retrievals are based on data coming from remote sensing instruments on satellites (Waliser et al., 2009).

There are two types of instruments the retrievals are based on: *active* and *passive* ones. Active instruments (such as RADARs) send out a signal and measure the backscattered radiation, while passive instruments only measure the incoming radiation without sending signals on their own. The IWP retrieved from an active instrument is likely more accurate than from a passive one since the active instrument gives one the strategic advantage of knowing the amplitude of the out-going signal. Furthermore, active instruments allow to retrieve the Ice Water Content (IWC) for different vertical layers. In contrast, retrievals based on passive instruments mostly allow only column-accumulated products (such as the IWP). However, active instruments are usually nadir-looking, i.e. they measure only a narrow area directly under the satellite. This constrains their spatial coverage drastically in comparison to passive instruments which typically scan a larger area. Hence, both instrument types have their strengths and limitations.

Holl et al. (2014) introduced the Synergistic Passive Atmospheric Retrieval Experiment-ICE (SPARE-ICE) product, which is one of the first retrievals based on data coming from active and passive instruments to combine their strengths: accurate IWP measurements with a high spatial coverage. It is build upon collocations[1] amongst the 2C-ICE product (Deng et al., 2010), which provides IWP values retrieved from RADAR and LIDAR measurements, and brightness temperatures measured by passive microwave and infrared instruments. These collocations and further auxiliary

[1]Collocations are measurements from different sources that were taken roughly at the same time and location.

data were used to teach a neural network how to calculate the IWP from brightness temperatures. The code is written in MATLAB and publicly available via the atmlab package[2]. I am going to refer to this software package as SPARE-ICE (atmlab).

While showing great potential, especially in regions where passive IWP retrievals are difficult, SPARE-ICE (atmlab) has some drawbacks. A major problem is a strong mid-latitude bias between its zonal mean and its training data 2C-ICE discovered by Li (2015) (see figure 1.1). SPARE-ICE seems to underestimate the IWP in comparison to 2C-ICE in mid-latitudes regions systemically. This is counter-intuitive since SPARE-ICE should show the same statistics as 2C-ICE when trained correctly.

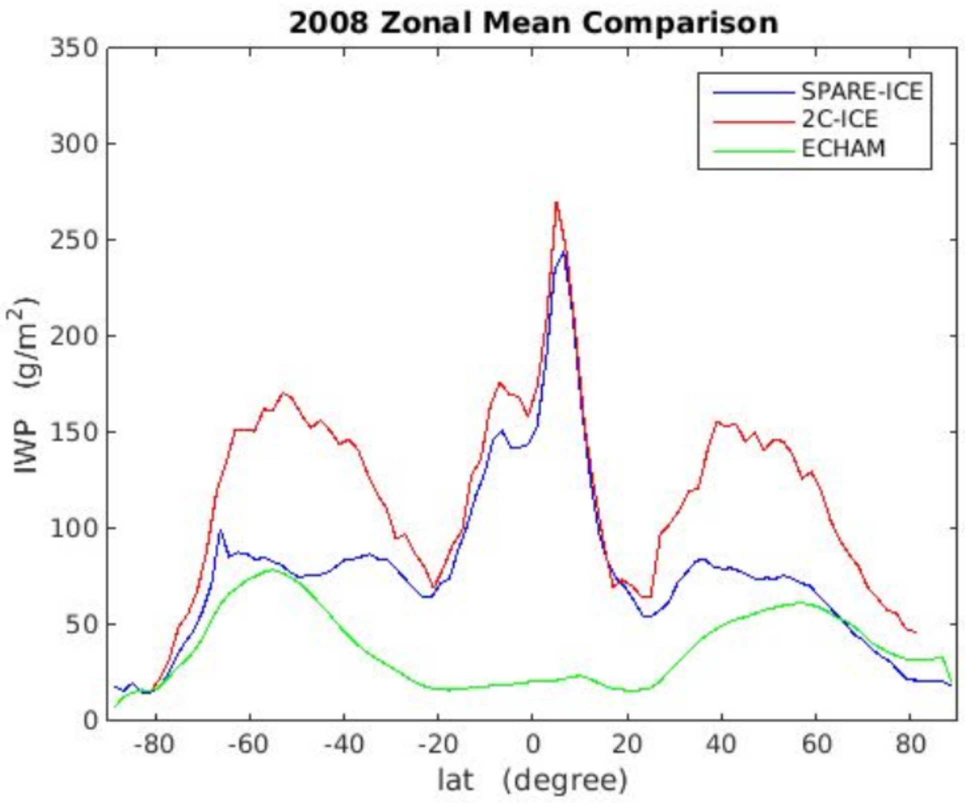


FIGURE 1.1: The zonal mean of the SPARE-ICE, 2C-ICE and ECHAM product of the year 2008. There is a large difference between SPARE-ICE and its training target 2C-ICE in the mid-latitudes (Li, 2015, Fig. 14)

While it is not clear where this bias comes from, other problems are caused by the design of the underlying software toolkit. SPARE-ICE requires a collocator for finding data to train its neural network and to calculate the IWP. This collocator was introduced by Holl et al. (2010) and was updated in John et al. (2012). It has shown its potential in many other studies (Eliasson et al., 2013; Holl, 2013; Kottayil et al., 2016). However, its code base is at a very experimental stage, is not automatically tested and is hardly documented. This makes its usage complex and hampers further development. It also leads to less transparency for experiments that are conducted with it. Additionally, it does not use parallel processing, which makes collocating of larger data volumes very slow.

[2]Get the stable version of atmlab here: http://www.radiativetransfer.org/misc/download/stable/2.2/.

The aim of this thesis is to reimplement SPARE-ICE and to solve these problems. Hence, it deals with two very different topics:

1. The **reimplementation** of SPARE-ICE (atmlab) and its underlying collocator is a software development task. The goal is to implement a new toolkit which provides the same functionality as SPARE-ICE (atmlab) but is easier to use, maintain and develop. It allows parallel processing support and better performance with large datasets. Its code base is developed by using object oriented programming principles and modern standards.

2. The **retraining** of the SPARE-ICE neural network. Here, I focus on the remaining training error of SPARE-ICE to 2C-ICE including the mid-latitude bias and try to reduce it.

The reimplemented toolkit, referred to as SPARE-ICE (typhon), is written in python and publicly available via the typhon package[3].

The thesis is structured as follows. In the next chapter, I summarize background knowledge from the literature to which I refer in later chapters. I present known collocation methods and summarize the basics of neural networks and decision tree classifiers. I give an overview about SPARE-ICE (atmlab) and its used datasets. Furthermore, I explain technical terms and concepts of programming that I am going to use for the reimplementation.

Owing to the nature of this thesis and to facilitate the reading flow, I separate the technical reimplementation from the retraining. Chapter 3 only deals with the technical implementation. Here, I analyse the flaws in the SPARE-ICE (atmlab) design and work out a new architecture to counter these issues. At the end of that chapter, I also evaluate the reimplemented collocator and show its further applications outside of the scope of SPARE-ICE.

In chapter 4, I address the retraining of the new implemented toolkit, SPARE-ICE (typhon). I analyse the IWP retrieval performance from SPARE-ICE (atmlab) to 2C-ICE including the mid-latitude bias and propose new training experiments. Finally, I present the results for the retraining and discuss them. In the last chapter, I conclude the achievements in this thesis and give a further outlook on how to improve SPARE-ICE (see chapter 5).

[3]Get the latest version of typhon here: https://github.com/atmtools/typhon.

Chapter 2

Background

In this chapter, I summarize different approaches for efficient collocation search. Furthermore, I explain the concept of machine learning techniques such as neural networks and decision trees. I present the SPARE-ICE product including the used datasets. Reimplementing SPARE-ICE requires profound knowledge of parallel and object-oriented programming (OOP). Hence, I name the most important terms and principles.

2.1 Collocation Methods

Several definitions of the term *collocations* exist; it mostly depends on the application or field of research in which it is used. In this thesis, I refer to the defintion by Holl et al. (2010): collocations or match-ups are events when two or more instruments measure almost the same location at almost the same time. Thus, two thresholds are important: the maximal spatial distance ΔS_{max} and the maximal temporal interval ΔT_{max}. Only if the distance and interval between the measurements are smaller than these thresholds, they are considered to be a collocation. Since this definition is very general and applicable to a broad range of events, it is also valid for instruments on satellites, ships or radiosondes. Another popular term is Simultaneous Nadir Overpasses (SNO), which is especially used in Earth science and constraint to the context of satellite applications. It describes collocations where both satellites were looking nadir (Cao et al., 2004). Collocations are used for a variety of applications: e.g. for the inter-calibration of satellite instruments (John et al., 2012; Yan and Weng, 2008, i.a.) or to combine different satellites products with each other (such as in SPARE-ICE or in Deng et al. (2010)). It depends on the instruments and the later application of the collocations which values one should choose for ΔS_{max} and ΔT_{max} . Normally, they guarantee that the observed atmosphere state has not changed significantly between the collocated measurements.

Measurements of remote sensing instruments do not represent points but rather areas. The form and size of these areas also vary with the viewing angle, e.g. nadir-pixels are smaller than off-nadir-pixels. Moreover, the sensor spatial response is not regular for the whole area but might rather have a Gaussian shape. However, the most collocation algorithms neglect these facts to simplify the search and reduce it to a range-query problem (see e.g. Holl et al. (2010)). A range-query or search denotes the problem where one has to find the neighbours of points in a set A with the points in another set B within a given radius. The time complexity[1] of the solution

[1]Time complexity denotes the dependence of the algorithm run time on the number of points N to process. $\mathcal{O}(1)$ means the run time is independent of the number of points, $\mathcal{O}(N)$ describes a linear relationship. An algorithm with quasi-linear time complexity scales efficiently with N. Other time complexities (such as $\mathcal{O}(N^2)$) would be dramatically slow with a large N.

depends mainly on the number of distances that have to be calculated.

Figure 2.1 shows a naive point-to-point search: calculating the distances amongst each point from A with B. All pairs of points with distances below ΔS_{max} are considered to be a collocation. This approach would not scale efficiently with the size of the sets since its time-complexity would be $\mathcal{O}(N_{\text{A}} \cdot N_{\text{B}})$.

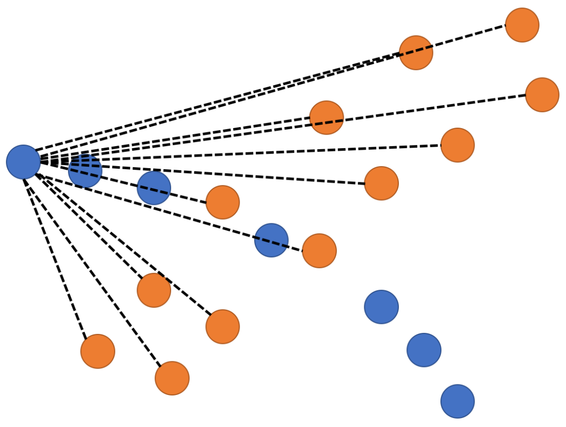


FIGURE 2.1: **Point-to-point search**: all distances for each point of A (blue) to each point of B (orange) must be calculated. This figure only shows the distances between the first point of A and all points of B as dashed lines.

2.1.1 Temporal Binning

A first optimization can be applied if a temporal criterion such as ΔT_{max} is given. The computation of spatial distances is more expensive than of temporal intervals because a spatial distance needs to be calculated in a higher dimensional space and needs a transformation of coordinates. One can avoid the unnecessary calculation of spatial distances by grouping the points in the sets according to their temporal coordinate. The groups (bins) of A should have a minimum period of ΔT_{max}. The corresponding bins of B are then expanded by ΔT_{max} and may overlap (see figure 2.2). Only the spatial distances between the points that are in the same bin will be calculated. This changes the time complexity of the point-to-point search to $\mathcal{O}(\frac{3}{N_{\text{bins}}} N_{\text{A}} \cdot N_{\text{B}})$ if the bin size is ΔT_{max}. However, the search for spatial collocations remains the crucial part. Therefore, there are different approaches to optimize it.

2.1.2 Prediction of Path Crossings

To reduce the amount of data to process, Cao et al. (2004) used the orbital viewing geometry of satellites to preselect the data (see figure 2.3). With the help of an orbital model, they predicted where the paths of two satellites will cross, i.e. where SNOs should exist. They performed the point-to-point search only for points near the predicted crossings. This reduces the amount of distance calculations. Anyhow,

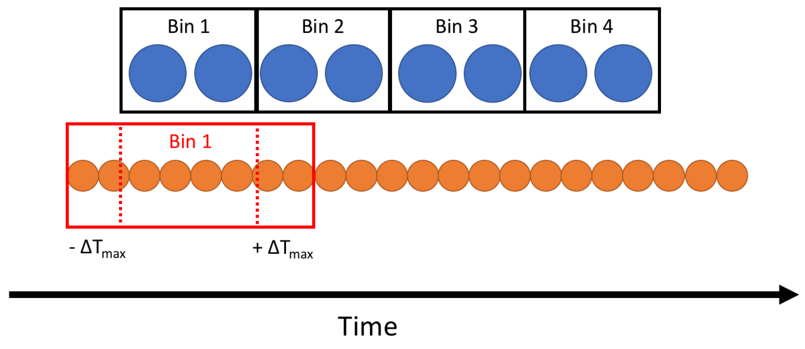


FIGURE 2.2: **Temporal pre-binning:** all points of A (blue) are grouped into bins. Only the points of B (orange) that fall in the same bin expanded by ΔT_{max} are searched for collocations. For example, bin 1 consists of two A points and all B points that are in the red bin.

it requires the accurate knowledge of the orbital parameters and limits the collocation search to instruments where predictions are possible. It is not applicable to other instruments which move in a more "chaotic" way such as ships or radiosondes. Furthermore, it does not find collocations between off-nadir pixels.

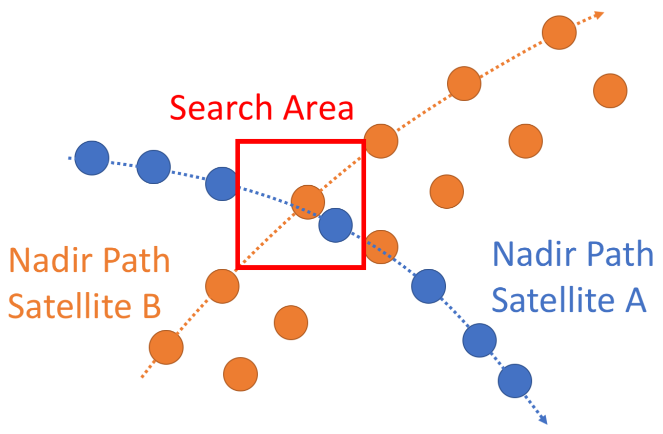


FIGURE 2.3: **Using path crossings**: the paths of the satellites are calculated by an orbital model. Only the area where the paths cross is searched for collocations.

Holl et al. (2010) uses a similar approach. It is even applicable to data where the orbital parameters are not known. In a first step, N sample points from A are drawn and the spatial distances of them to all points in B are calculated. All sample points with distances smaller than ΔS_{max} to any point of B define super-intervals with their neighbour sample points. Only the points in the super-intervals are considered for the point-to-point search. The value for N is crucial for the performance of this method. If it is too small, too many distances must be calculated or some collocations might even be missed. If N is too large, the method will get less efficient. Whether all collocations are found also depends on the structure of the data. This method might work better with regular satellite paths than with chaotic radiosondes tracks.

2.1.3 Fixed-Size Grids

To provide a more general approach, one can divide the points of A into grid cells with a minimum side length of ΔS_{max} (see Figure 2.4). When searching for collocations of one point of B, one just has to consider all points of A in the corresponding and adjacent grid cells. The collocator of SPARE-ICE (atmlab) uses this approach (Holl et al., 2014; John et al., 2012). Instead of grid lines one can also use grid planes when transforming the geodetic coordinates to 3-dimensional Cartesian coordinates.

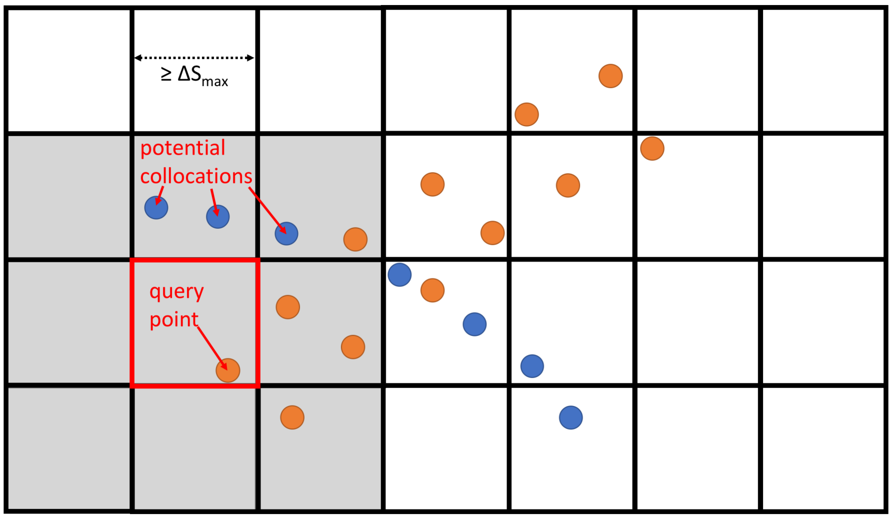


FIGURE 2.4: **Fixed-size grids**: when searching for collocations for all points in one grid cell (red), only the points in its adjacent grid cells (grey) are considered.

Collocating with grids has two advantages. Firstly, it reduces the number of distances that have to be calculated significantly. Secondly, the gridding procedure itself is fast and does not produce much overhead since putting the points into the grid cells has a linear time-complexity. However, there are issues that need to be considered:

- **Unbalanced cells:** If the points are unevenly distributed in the space, some grid cells might contain no points while others contain many of them. This makes the collocation search less efficient.
- **Cell size:** It is not trivial to find the optimal grid cell size. If it is too small, many cells won't be filled and the overhead costs might be too high. If it is too large, a high number of distances calculations remains. When using geodetic coordinates (such as longitudes and latitudes), grid cells at the equator will be larger than at the poles. One might adjust the gridding procedure such as the cell size depends on the latitude but this increases the gridding costs.

Owing to these issues it is difficult to specify the time complexity of this approach exactly. However, it might be considerably faster for a high number of points than applying the point-to-point search directly (Bentley et al., 1977).

2.1.4 Space-Partitioning Trees

Instead of using fixed-size grids, one can use space-partitioning trees, which allow to partition the data points more dynamically. A tree is a structure based on nodes which are stacked in layers. As in a mathematical graphs, the nodes are hierarchically related. A node can have parent nodes (its direct ancestor nodes in the hierarchy), sibling nodes (other nodes with the same parent) and children nodes (inverse to parent nodes). The top node of a tree is called *root* node and all nodes without children are *leaf* nodes.

How does a tree help to solve a range-query problem efficiently? One can use it as a spatial index for one of the datasets (Bentley and Maurer, 1980). For exmaple, all points in A are partitioned recursively into smaller subsets, which are held by the nodes in the tree. Each node n represents a circle (or a ball in a 3-dimensional space) enclosing its points. One starts with the root node. All points are assigned to it. On the next layer, the points are split into two children nodes. They are split as such both children contain the same number of points and their circles are the smallest possible. Each of these children nodes is divided again and this continues until only a few points are left in the leaf nodes in the last layer (see Figure 2.5). An efficient construction algorithm for this kind of ball tree is described in Dolatshah et al. (2015).

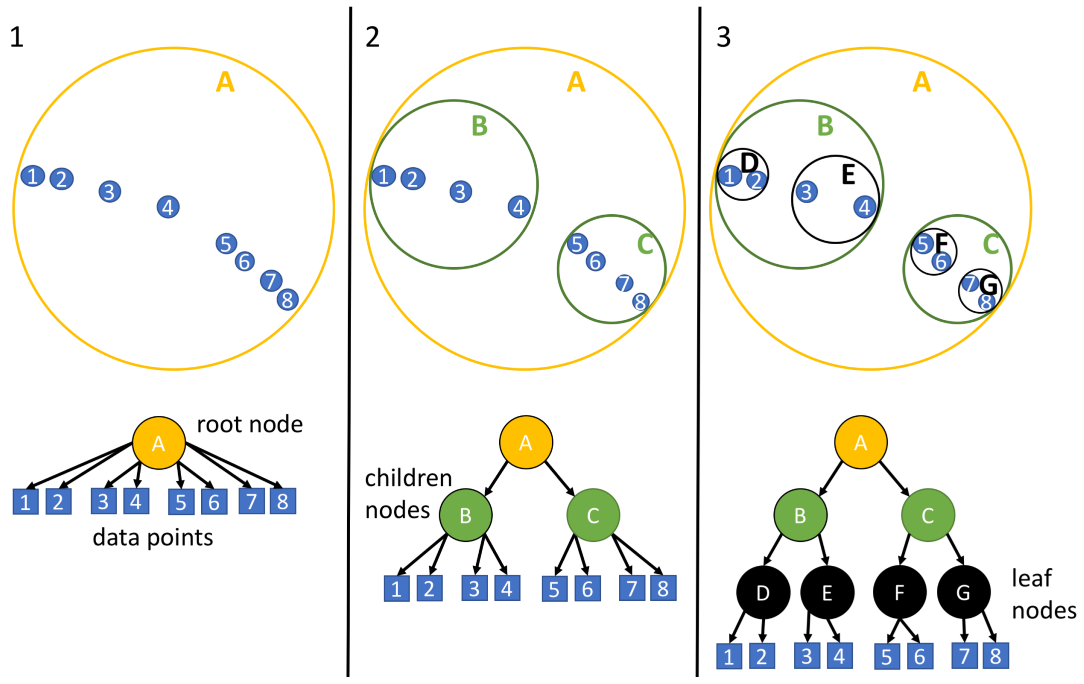


FIGURE 2.5: Construction of a ball tree.

After creating the tree, it allows very fast range queries. For example, we can use it to find collocations between A and B. One starts with checking whether the query area, a circle with the radius $\Delta S_{\max}$ around a point in B, overlaps with the circle of the root node. If it does not, one can be sure that this point from B has no collocations with A and can move to the next point in B. If it overlaps, one goes one layer deeper in the tree and makes the same checks with the circles of the children nodes. If both do not overlap, the query stops. But if one or both do, their children nodes will be also checked and so on. When reaching a leaf node, all points in there will be checked using the point-to-point search. See Figure 2.6 and algorithm 1 for an example of a query within the ball tree.

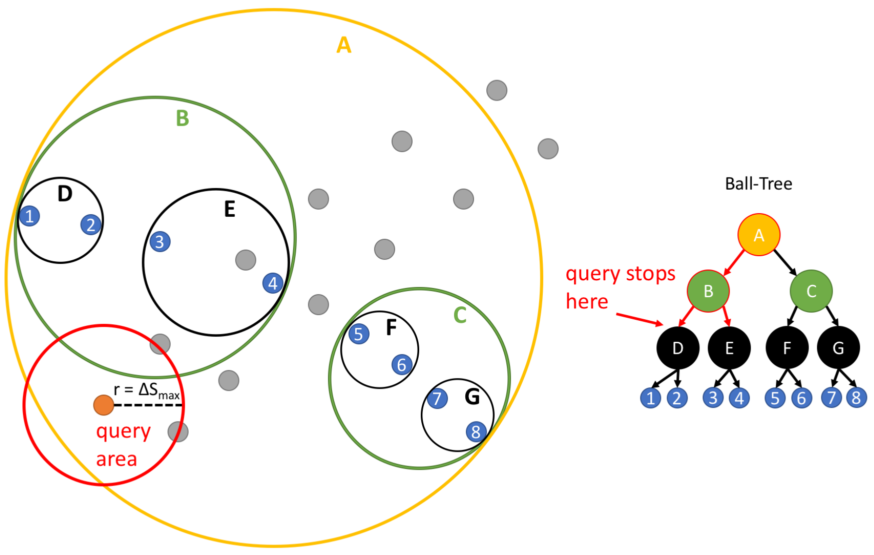


FIGURE 2.6: **Range-query with a ball tree**: One starts with checking whether the query area (red circle) overlaps with the circle of the root node (A). It does, so the circles of its children (B and C) will also be checked. Only B does overlap therefore only its children are checked in the next step. However, neither D or E overlap with the query area and the query stops. No collocations were found for the given query point.

```
// Define a set of query points $B$ and a ball tree $T$ holding the points
   in $A$.
for $b_i \in B$ do
    // Query circle $q$ defined by pivot $b_i$ and radius $\Delta S_{\max}$:
    $q = \text{Circle}(b_i, \Delta S_{\max})$;
    // Start with the root node:
    $N = \{\text{Root(T)}\}$;
    while $N$ is not empty do
        Remove the front node of $N$ and assign it to $n$;
        if $q$ overlaps Circle($n_p$, $n_r$) then
            if $n$ is leaf node then
                Check for each $p \in \text{Points}(n)$ whether it is in $q$;
            else
                Add Children($n$) to $N$;
            end if
        end if
    end while
end for
```

Algorithm 1: Pseudo-code of a range-query with a ball tree.

The query algorithm is so fast because each time when moving one layer deeper one might discard the half of all remaining points if the radius of the query area is small enough. This makes the ball tree one of the most efficient methods for range queries in a low-dimensional space (Liu et al., 2006; Omohundro, 1989) and reduces its query time complexity to $\mathcal{O}(N_B \log N_A)$. However, the creation of a ball tree takes also some time and must be considered too. The most efficient creation algorithms reach time complexities of $\mathcal{O}(N_A \log N_A)$ (Dolatshah et al., 2015).

2.2 Neural Networks

An artificial neural network (ANN) is a machine learning technique which can learn to regress or to classify target values from input data by selecting important features. For example, it can be used to find a function which calculates the IWP from observed brightness temperatures as in SPARE-ICE. It is built upon virtual neurons in multiple layers. If it uses non-linear activation functions within its neurons, it is also called multilayer perceptron (MLP) and can learn to approximate any smooth, measurable function (Hornik et al., 1989). In this section, I shortly explain how a MLP is designed and how it is trained. I also address the practical issues which arise during the training of MLPs and show how they can be solved.

2.2.1 Concept

A MLP consists of neurons, which are arranged in stacked layers. There is one input, at least one hidden and one output layer. The neurons of each layer are linked to all neurons from the next layer via weighted connections. Connections within one layer are not allowed (see Figure 2.7).

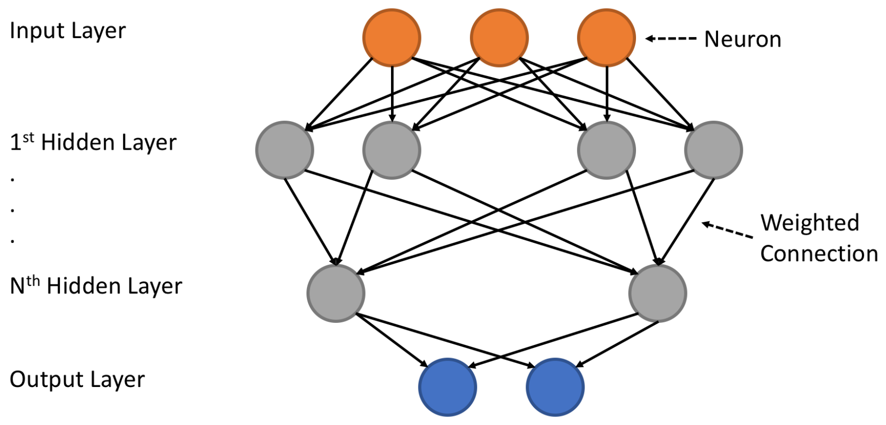


FIGURE 2.7: An example for a multilayer perceptron (MLP). Each layer of neurons is connected with its next layer except for the output layer. The connections are weighted with values that can be trained.

During the *forward pass*, the numbers coming from the input layer will be transferred via the connections through every neuron in the hidden layers ending in the output layer. Each neuron sums up the information from its input weighted connections, apply an activation function to it and pass the result via it output connections to the next neuron layer. Expressed in mathematical terms, the neuron j does this:

$$y_j = f(\sum_i w_{ij} \cdot x_i) \tag{2.1}$$

where x_i are the outputs from the previous layer and w_{ij} are the weights of the connections. f is called activation function and usually a non-linear and differentiable function such as the sigmoid or the tangens hyperbolicus. When training a MLP, one needs a set of input and desired output (target) vectors. The input vectors are then passed forward through the MLP to produce output vectors. The difference between the output and target vectors is the error of the network. To minimize the error, each weight w_{ij} is adjusted according to its error gradient, i.e. its influence on the total network error. The error gradient is calculated using a back-propagation algorithm (LeCun et al., 1998; Rumelhart et al., 1986). The name comes from the working direction of the algorithm: it starts to derive the error gradients between the last hidden layer and the output neurons and passes them *backward* through the neuron layers until the first hidden layer is reached. Different hyperparameters such as learning rate and momentum help the error correction to converge more quickly to a global minimum. After successful training, the MLP stored the features and statistics of the training data in its weights.

2.2.2 Training Issues

There are several issues which might hinder a successful training of an ANN.

Hyperparameters

The back-propagation algorithms, which are used to minimize the error of an ANN, can be stuck in a local minimum or can converge only very slowly due to ill-chosen hyperparameters. Hyperparameters are values like the number of hidden neurons, type of activation function, etc. Unfortunately, they are dataset-dependent and cannot be optimized before the training. To avoid these issues, one usually trains many ANNs with different randomized connection weights and combinations of hyperparameters. In the end of the training, one only chooses the best-performing network. The scikit-learn package provides the GridSearch tool, which tries different hyperparameters step-by-step until it finds the best ones (Pedregosa et al., 2011).

Overfitting

Due to the high number of training iterations and parameters which can be adjusted, an ANN tends to *overfit* its weights to the training data. The resulting approximated function then performs very well on the training data but fails on data that was not seen during the training. Instead of generalizing knowledge, it simply memorizes the specific features of its training data. A first solution is to split the data into three parts: training, validation and testing. The training data is literally used to train the weights of the network; the error between the output and the training target will be used to adapt the weights. After each training iteration, the error for the validation data is calculated. If it gets bigger with the training time, the training is stopped since the ANN starts to forget general knowledge about the data and overfits to the training data. After hyperparameter tuning, the ANN is chosen which performs the best on training and validation data. However, to prevent an overfitting of the hyperparameters to the training and validation data, one needs the testing data. The error of the testing data does not influence the weights training or hyperparameter selection but is simply a estimation of the real error for the ANN on unseen data.

Another way to prevent overfitting is regularization. When using regularization, a penalty term is added to the total error of the ANN, which gets bigger the higher the weight values are. This leads to smaller weights in general and concentrating on the most important features of a dataset.

Holl et al. (2014) simply split their data in 40% training, 20% validation and 40% testing parts. However, there are more efficient ways to use the data; one of them is cross-validation (Kohavi, 1995; Olden and Jackson, 2000). Using cross-validation, one starts by splitting the data in training and testing sections (75% to 25%). The training data is further split in k parts called *folds* (k is usually between 3 and 10). k models are trained each using $k-1$ folds for training and the remaining fold for validation. Each model uses a different combination of folds than the others; so all folds were used once as validation and $k-1$ times as training dataset. The model which performs the best on both datasets wins and is returned. Cross-validation allows to use more data during training than the simple splitting approach. Moreover, it also assures that all data has been used once as training data to avoid a bad selection owing to the splitting. However, cross-validation increases the run-time since k times more models must be trained.

Feature Scaling

ANNs are sensitive to the scaling of its input values. For example, SPARE-ICE has i.a. brightness temperatures and the latitude as input values. While brightness temperatures have a numerical range from 150 to 300, latitudes are within -90 and 90. The ANN might overestimate the importance of the brightness temperatures on the desired output, just because they have a higher numerical value than the latitudes. To counter this, all input values should be rescaled such as their numerical values are in the same range. One approach is to divide all values by their average.

2.2.3 Feature Importance

ANNs or many other machine learning models act like a black-box, i.e. one collects sample data from a phenomenon, trains the model with it and uses the output values. One does not need much prior knowledge about the relations and connections between the data in the forehand since the model learns it from the data. However, the model offers results but it is difficult to interpret them for human scientists. For example, an ANN can consist of several hundred up to thousand adjustable weights. Analysing them individually to get the influence of features in the data on the output is simply exhausting. There are algorithms to measure the feature importance by using the product of weights or the output gradients to the inputs (Gardner and Dorling, 1998; Olden and Jackson, 2002). However, the results are sometimes misleading. A simpler way is to repeat the trainings of the ANNs with a different combination of inputs fields. However, this increases the development time a lot.

2.3 Decision Tree Classifier

A decision tree classifier (DTC) is a hierarchical structure, which learns to classify data by going through a chain of yes/no questions in its nodes (Safavian and Landgrebe, 1991). Each answer to these questions helps to assign the data to its corresponding class. The questions are very simple: *is input X greater or lower than T*? The values for X and T are learnt from a sample of training data, mostly by performing principal component analysis.

DTCs have many advantages over ANNs. Owing to their simplicity, DTCs are much faster to train and easy to visualize for humans. Therefore, they are often referred to white box models in contrast to ANNs. They are not sensitive to feature scaling and do not need an exhaustive hyperparameter tuning. DTCs allow to calculate the feature importance with the Gini index, i.e. how important a feature is for the resulting classification of the data (Louppe et al., 2013).

However, they strongly overfit the training data if one allows too complex trees. To avoid overfitting, one usually creates a forest of random initialized trees with limited depths, also called random forest classifiers (RFC, Breiman (1999)). Another technique is pruning where nodes with too specific questions are removed from the tree. Trees have another disadvantage, they cannot extrapolate on data which are outside of the range of its training data.

2.4 SPARE-ICE

In this section, I explain the motivation behind SPARE-ICE. Furthermore, I shortly present the its used datasets and its major components.

2.4.1 Motivation

There are two types of remote sensing instruments used for IWP retrievals: *active* and *passive* ones. Active instruments (such as a RADAR and LIDAR) send out electromagnetic waves and measure the backscattered signal from those waves; they have one sender and one receiver unit. Passive instruments only measure incoming radiation without actively sending electromagnetic waves by themselves, so they only consist of one receiver unit. On the one hand, the active instruments have a strategic advantage for the retrieval because one knows the amplitude of the emitted signal in the first place. On the other hand, this makes active instruments more expensive and limits their life-time in space due to higher energy costs and faster abrasion. Furthermore, satellite-based active instruments are nadir-viewing, i.e. they measure only a narrow area vertically under the satellite. This constrains their spatial coverage drastically in comparison with passive instruments which normally scan a larger area. The SPARE-ICE product introduced by Holl et al. (2014) uses collocations amongst active and passive instruments on different satellites to provide IWP measurements with a larger spatial coverage. The IWP coming from active instruments is used as target for a neural network while the inputs are the collocated measurements from passive instruments. After training the neural network, it can retrieve the IWP from passive instruments only while having a similar accuracy as using active instruments hopefully. Since the passive radiometers cover a very long time period (almost 20 years) and scanning with a large swath width, the resulting IWP product provides much more spatial and temporal coverage than the active instruments.

2.4.2 Used Datasets

Four instruments are mainly used for SPARE-ICE: two active and two passive instruments; combined in three datasets. Additionally, there are also two auxiliary datasets used. The dataset with active instruments is only needed as a target for the training. The operating wavelengths of the instruments can be split into three ranges: visible light (VIS), infra-red (IR) and microwave. Owing to the change of scattering regimes of ice particles with the observing wavelength, the instruments have different sensitivities. Holl et al. (2014) and Li (2015) already described the used instruments very extensively including the systematic errors of the target dataset. Hence, I pass to explicate them here in detail and I focus on summarizing their most important features for SPARE-ICE instead.

2C-ICE Dataset (Target)

2C-ICE (Deng et al., 2010) is an IWP dataset retrieved from two active instruments: the Cloud Profiling RADAR (**CPR**) carried on CloudSat and the Cloud-Aerosol LIDAR with Orthogonal Polarization (**CALIOP**) on the Cloud-Aerosol LIDAR and Infrared Pathfinder Satellite Observation (CALIPSO). CloudSat and CALIPSO are both in the A-Train[2], so their measurements can continuously be collocated and combined. The CPR operates at 94 GHz (microwave) and is sensitive to large ice particles since the backscattered signal is $\propto D^6$ (rayleigh scattering regime)[3]. It is able to penetrate thick clouds. Complementing these measurements, CALIOP works at two visible channels: 532 and 1064 nm. Although the backscattered signal is still stronger for larger particles ($\propto D^2$ due to geometrical optics), it is dominated by small particles at the top of the cloud owing to their much higher occurrence. Therefore the laser signal attenuates quickly and CALIOP cannot look through deeper clouds.

Both instruments are nadir-viewing and have roughly a footprint size of 1 km. Therefore, they have only a sparsely global spatial coverage. 2C-ICE provides IWP values and is therefore the training target of SPARE-ICE.

MHS Dataset (Input)

The Microwave Humidity Sounder (**MHS**) or its predecessor, the Advanced Microwave Sounding Unit (**AMSU**), are passive instruments on the satellites NOAA-15 to 19 and Metop-A and B (Kleespies and Watts, 2006). Holl et al. (2014) trained SPARE-ICE only with data from NOAA18 as it has the most collocations with 2C-ICE. Both instrument versions operate at five channels with microwave frequencies. They are listed in Table 2.1. Channel 1 & 2 are surface channels, while the other channels never see the surface except for very dry atmospheres. Like the CPR, the microwave channels are only sensitive to large ice particles and are able to penetrate thick ice clouds.

TABLE 2.1: Channels and frequencies of MHS and AMSU. Note AMSU has originally more channels but they are not listed here.

Channel	Frequency [GHz]
1	89
2	150 (AMSU), 157 (MHS)
3	183 ± 1
4	183 ± 3
5	183 ± 7

[2]The A-Train is a constellation of five satellites which fly in the same polar orbit shortly after each other.

[3]D denotes the volume-equivalent radius.

MHS and AMSU are scan-viewing, i.e. they rotate to cover a larger area. For each rotation (also called scan line), they provide 90 measurements with a diameter of ca. 16 km at nadir. Off-nadir pixels are larger. The observations of the off-nadir pixels also suffer from the limb cooling effect due to the longer path through the atmosphere, i.e. brightness temperatures from off-nadir pixels are systematically lower than from nadir pixels. Holl et al. (2014) passed the viewing angle of MHS as input field to SPARE-ICE to trigger an automatic limb correction by the neural network.

AVHRR Dataset (Input)

The Advanced Very High Resolution Radiometer (**AVHRR**) is also a passive instrument on the same satellites as MHS and AMSU but it measures with a higher spatial resolution and at other frequencies (Cracknell, 1997). The newest version of AVHRR, AVHRR/3, uses five channels. Two channels operate at visible wavelengths, one switches between visible and near-infrared wavelengths and the remaining two channels are only infrared. Table 2.2 shows the AVHRR channels and their operating wavelengths. The switch between 3A and 3B happens on Metop satellites only. The NOAA satellites continuously use 3B.

TABLE 2.2: Channels and wavelengths of AVHRR/3.

Channel	Wavelength [μm]
1	0.630
2	0.862
3A	1.61
3B	3.74
5	10.80
6	12.00

As Holl et al. (2014) already pointed out, the visible channels can be used for detection of larger clouds while the infra-red channels are helpful to retrieve the IWP from even thin cirrus. The used AVHRR dataset (namely AVHRR GAC) has a higher spatial resolution than MHS; its nadir pixels have a diameter of around 4 km. Since it is also a scan-viewing instrument, a limb correction must be applied to its off-nadir pixels as well.

CFSR Dataset (Auxiliary Input)

The Community Forecast System Reanalysis (CFSR) dataset contains hourly surface temperatures for the whole Earth (Saha et al., 2010). It is used as an auxiliary input for SPARE-ICE (atmlab) to improve the performance of the infra-red channels.

Elevation Dataset (Auxiliary Input)

To enhance SPARE-ICE on regions with a higher terrain, an elevation dataset was added as auxiliary input (Amante and Eakins, 2009). It contains the surface elevation in meters on a 1 arc-minute grid.

2.4.3 Workflow

Figure 2.8 shows a simplified overview about the workflow. SPARE-ICE mainly consists of three major components: a collocator and two ANNs. One regresses the raw IWP retrieval, the other predicts the ice cloud probability. The ice cloud probability is required since the IWP regressor was trained in log space to learn the large value range of IWP better (10^{-2} up to 10^4 g/m^2). Otherwise, the ANN would reduce the errors of the large IWP values preferentially while neglecting smaller values. Because operating in log space, it cannot predict a IWP value of zero. Hence, one needs a cut-off value for small IWPs. There are two working modes for SPARE-ICE: *training* and *retrieval*.

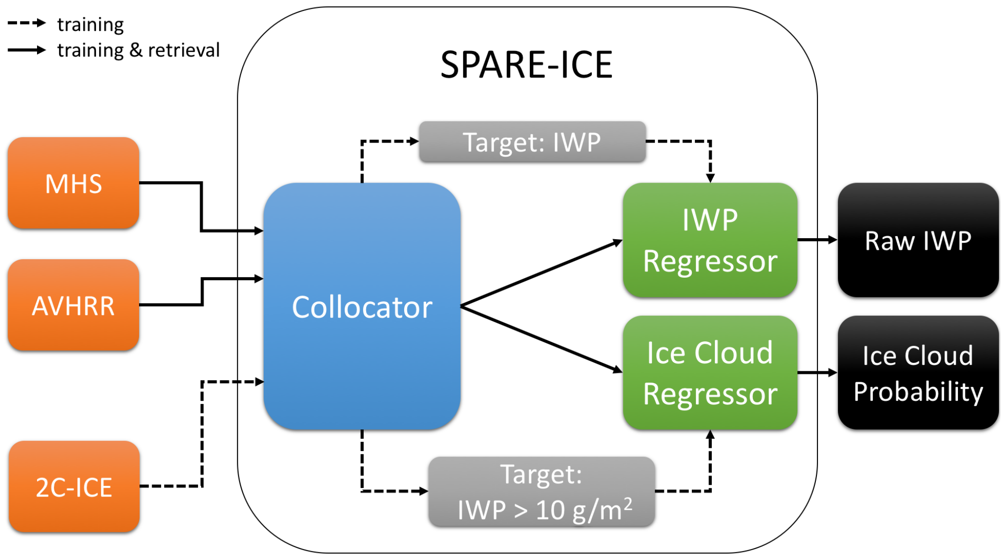


FIGURE 2.8: A simplified structure of SPARE-ICE. The source data (orange) is collocated by a collocator (blue). In the training mode, the collocations will be used to train both neural networks (green), the IWP and the ice cloud regressor. The target of used during the training is the IWP (grey) coming from the 2C-ICE product. In the retrieval mode, the neural networks only need the collocations coming from MHS and AVHRR and returns the raw IWP and ice cloud probability (black) directly to the user.

In the *training* mode, the collocator searches for collocations amongst 2C-ICE, MHS and AVHRR to gather input and target fields for both neural networks. See Table 2.3 for a list of the fields. A temporal filter of ΔT_{max} = 10 min and a spatial filter of ΔS_{max} = 7.5 km are applied. These filters are a compromise between getting too much noise through a change of scene and obtaining too few collocations for the limb angles. Because the footprint size of the 2C-ICE and AVHRR instruments are smaller than of MHS, the collocator finds multiple pixels of them for one MHS pixel. The multiple pixels are averaged (*collapsed*) to obtain one value for each collocation. The number and standard deviation of the averaged pixels are stored. The maximum number of 2C-ICE pixels collocating with one MHS pixel is 15. Even then, the 2C-ICE pixels only cover 6.5% of the area of the MHS pixel. This can lead to a systematic error due to misrepresentation. To reduce this effect, an inhomogeneity filter is applied which let only collocations pass where enough 2C-ICE measurements have seen the same scene, i.e. at least 10 or more 2C-ICE pixels with a maximum variation coefficient of 1 are used for training.

TABLE 2.3: Input and target fields for SPARE-ICE (atmlab) according to Holl et al. (2014). Note to s*surface temperature*: it is not entirely clear whether and how it was used (compare Table 2 and description of Figure 2 in Holl et al. (2014)). Deducing from the SPARE-ICE (atmlab) performance and code, I assume it is used for both retrievals.

Input Fields	Used in IWP Retrieval	Used in Ice Cloud Classifier
MHS Channel 3, 4, 5	X	
AVHRR Channel 3B, 4, 5	X	X
Local Zenith Angle	X	
Local Azimuth Angle	X	
Surface Temperature		X
Surface Temperature	X	X
Surface Elevation	X	X
Ice Water Path	**As Target**	
Ice Water Path > 10 g/m^2		**As Target**

After training and in the *retrieval* mode, SPARE-ICE can retrieve the IWP from collocations between MHS and AVHRR without needing the target IWP from 2C-ICE any longer. The resolution is limited to the size of one MHS pixel, i.e. roughly an area with a diameter of 16 km at nadir.

2.5 Programming Terms & Concepts

2.5.1 Parallel Programming

Owing to the large amount of data which have to be processed, modern software solutions often use parallel workers to reduce their runtime. There are two different types of workers which can run in parallel.

- A **process** is an executed application, i.e. a running program. It is usually independent from other processes and does not share data in the working memory with them. Communicating between processes is possible via pipes or queues but might be computationally expensive.
- A **thread** runs asynchronously within a process and shares its data with other threads in the same process.

Depending on the task and programming language, processes and threads are used for different purposes. Processes work autarkically and are well-suited for *embarrassingly parallel* problems where the workload can be distributed on multiple parallel workers without much extra effort. Embarrassingly parallel problems does not require the parallel workers to communicate with each other or to synchronize. For example, processes are ideal when many files must be processed and the working routine can be applied on each file alone. Threads are more efficient when the working routine needs to share data with other workers or when its return value is large. For example, pre-loading files in a parallelized queue to use them later in the main program, should be done rather by threads than processes.

2.5.2 Object Oriented Programming

Object-oriented programming (OOP) is a programming paradigm where one bundles functions with code on objects that contain data in form of attributes (Kindler and Krivy, 2011). It shall facilitate the development and reuse of larger software solutions and "has come to be considered a panacea to all computing aches" (Mili et al., 1995). In the following, I explain the most important terms and concepts of OOP.

Objects & Classes

Objects are the core elements of an OOP language. They bundle functional code in form of *methods* with data which is stored in their *attributes*, together they are called members. Using objects in the program code allows to model ideas, concepts or items of a real or imaginary world and make them more concrete for the programmer. Classes denote the blueprints for objects. They tell the compiler and human what methods and attributes an object possesses.

Encapsulation

The most important concept of OOP is encapsulation of classes, i.e. the class hides functionality and more complex things from the user. The user does not have to know how a class solves a problem but rather that it does solve it. In order to doing so, a class can define methods or attributes as *private*. Private parts of a class are neither visible or accessible outside of the scope of a class. The opposite of private members are *public* members. They provide the user API of a class and and are normally well-documented and tested.

Inheritance

Classes can inherit their methods and attributes from other classes. For example, if there is a class A which inherits from a parent class B, it can use all its methods and attributes including its private members. Inheritance from multiple classes and over generations is also possible, i.e. A can inherit from B and C. It would also get access to members of potential parent classes of B and C.

Composition

Instead of using inheritance, an object can also use other object compositionally. This means, it stores the object as one of its attributes. In this case, it has only access to its public members.

Chapter 3

Reimplementation of Collocator and SPARE-ICE

This chapter deals with the technical reimplementation of SPARE-ICE (atmlab). I work out improvements for its technical issues as the complex structure and slow performance on large datasets. Afterwards, I present the new classes on that SPARE-ICE (typhon) is built and explain their roles in the implementation. Since the collocator is a powerful stand-alone toolkit which might have other purposes than just SPARE-ICE, I evaluate it at the end of this chapter and show further applications.

3.1 Technical Issues of SPARE-ICE (atmlab)

Fixed-Size Grids

Although the collocation routine in SPARE-ICE (atmlab) is fast, it can still be optimized since it uses fixed-size grids. As described in subsection 2.1.3, this collocation method suffers from unbalanced cells and the grid cell size problem. Both affect the performance of the collocation algorithm. I want the collocator generally applicable to any kind of geo-referenced data so a prediction of path crossing will not work (as in Cao et al. (2004) or Nagle and Holz (2009)). As mentioned earlier, ball trees are better suited for range-queries and work more efficiently than fixed-size grids. Hence, I decided to switch to ball trees in the reimplementation.

Serial Processing

It takes around one week to create the SPARE-ICE product for one whole year on a single-core machine. The most of the runtime is consumed by the collocator, specifically its reading and collocation routines. Table 3.1 shows the benchmarks of the collocator for two scenarios needed for SPARE-ICE. Collocating MHS with AVHRR than with 2C-ICE takes longer because one AVHRR orbit contains much more data points than one from 2C-ICE.

TABLE 3.1: Runtime of the collocator in SPARE-ICE (atmlab) on different datasets for a data period of 30 days.

Datasets	**Runtime [min]**
MHS & 2C-ICE	75
MHS & AVHRR	806

However, using parallel processing could speed-up this process a lot. For example, unzipping and reading of one orbit file of 2C-ICE and its overlapping MHS files can take up to 7 seconds on a regular machine. The collocation search using fixed-size grids takes barely 2 seconds. This means, the collocator mostly waits until the reading of the files is done before it can start with collocating. If the collocator searches in one day of data (15 orbits), reading and collocating would take circa $(7\text{s} + 2\text{s}) \cdot 15 = 135\text{s}$. An improvement could be a queue which reads the file pairs in background threads, so the main workflow does not have to wait for the reading routine. Assuming four background reading threads and one thread to collocate the loaded data, the collocation thread is not held back by the reading routine any longer except for the very first file pair. This would reduce the runtime to $7\text{s} + 15 \cdot 2\text{s} = 37\text{s}$. Additionally, all file pairs could be distributed on multiple processes which work through their stack of file pairs independently.

Performance of Haversine Metric

To calculate the distance between two points on the Earth's surface, Holl et al. (2010) uses the great circle distance formula, also known as the haversine metric:

$$d_{\text{haversine}} = 2 \arcsin\left(\sqrt{\sin\left(\frac{\varphi_2 - \varphi_1}{2}\right)^2 + \cos(\varphi_1) \cdot \cos(\varphi_1) \cdot \sin\left(\frac{\lambda_2 - \lambda_1}{2}\right)^2}\right) \tag{3.1}$$

The haversine metric consists of many trigonometric functions, which make it computationally expensive. The euclidean (or tunnel) metric is much faster to compute but works in a Cartesian coordinate system only:

$$d_{\text{euclidean}} = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2} \tag{3.2}$$

However, geocentric coordinates can be converted into Cartesian ones. Although this requires trigonometric functions as well, it is cheaper because it only has to be done once for each point. Instead of the standard euclidean metric one can also use the reduced version: $d^2_{\text{euclidean}}$. This avoids the expensive calculation of the square root but still allows to use it as a criterion to order a list of points and their distances. A ball tree can use the reduced version to speed up its creation time.

The euclidean metric denotes a straight line between two points which goes through the Earth's surface while the haversine metric is the distance *as the crow flies*. Hence, one introduces an error by using the euclidean instead of the haversine metric. The error gets larger the larger the distance between the two points is. Figure 3.1 shows the error, which depends on the distance between the points and the position on the Earth's surface. Since the error is the largest close to the poles, the figure is limited to the pole region. For a distance of 7.5 km as it is used in SPARE-ICE, the error is always smaller than 10 m. Therefore, I decided to use the euclidean metric for distance calculation between collocations. However, the collocator can be used with the haversine metric as well.

Expanded Output

The output of found collocations must contain the time and geolocation of both collocating points and potential additional data fields (such as brightness temperatures, etc.). When searching for collocations, it might be possible that several points from one dataset collocate with the same point of another dataset. This is often the case

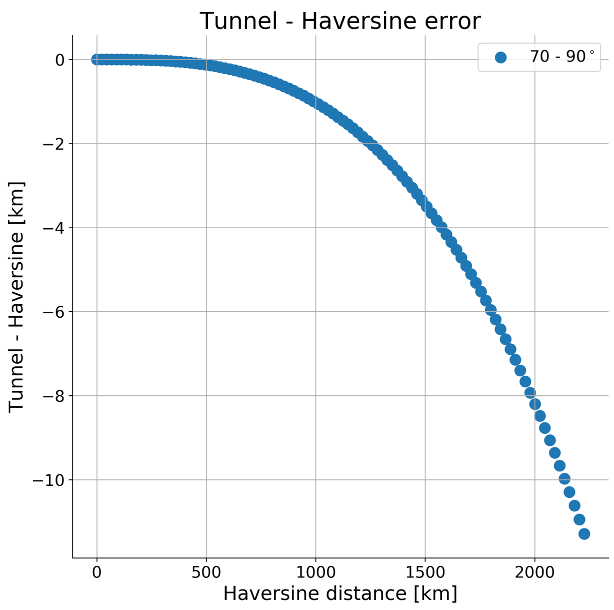


FIGURE 3.1: The error between the euclidean (tunnel) and haversine metric for the north pole region. On the x-axis is the real distance computed with the haversine metric. The y-axis shows the difference between the tunnel and haversine metric.

when the search criteria are large or the resolution of one dataset is higher than from the other dataset, e.g. when collocating 2C-ICE with MHS. There are three ways to store the collocations:

- **Collapsed:** Multiple collocations of one pixel are averaged and *collapsed* to one value. This helps to reduce the amount of data drastically and facilitates further post-processing. The averaging could also be weighted respecting the distances or uncertainties of the single values. However, the collapsing is irreversible.
- **Expanded:** Each pixel with multiple collocations will be duplicated to conserve the aligned data structure. This needs more memory space because all information for a point must be copied for each of its collocations. However, this is the conceptually easiest form.
- **Compact:** Instead of aligning the data directly with each other, only the indices of the data are aligned. This allows to avoid copying the data fields and saves a lot of memory space in comparison to the expanded mode.

Since collapsing of data points is irreversible and expanding increases the memory space a lot, I decided to store the collocation output in the compact mode per default. SPARE-ICE (atmlab) stores collocations in the expanded mode, which may cost a lot of memory space for collocations with high-resolution datasets.

Excessive Use of Multiple Inheritance

SPARE-ICE (atmlab) was implemented object-oriented. The intention of object-oriented programming is to make the code reusable and less complex as explained in subsection 2.5.2. However, the excessive use of multiple inheritance makes the overall code very complex. According to Snyder (1986), multiple inheritance should be used sparely and with caution otherwise it violates the important principle of encapsulation in OOP. This makes further development very laborious since the *related* classes[1] do not only communicate via their public interface but they can also access their private members. This means, if one wants to change the behaviour of one child class, one must be aware of the complete implementation of all ancestor classes since they might interfere with each other. To avoid multiple inheritance, many OOP architectures are built on composition. Instead of inheriting from all classes to bundle their functionality in a child class, the child class uses them through their public interface and build wrappers to provide their functionality to the users. This allows to separate the functionality coming from different sources and facilitates the further development. During designing the classes in SPARE-ICE (typhon), I follow the principle of composition over inheritance.

3.2 Implementation of SPARE-ICE (typhon)

The overall workflow stays the same as shown in Figure 2.8. However, the individual components change. In this section, I explain how I implemented them in classes and what their role in SPARE-ICE (typhon) is.

FileSet

All datasets which are used in SPARE-ICE (2C-ICE, MHS, AVHRR) are stored in one file per orbit. The path and name of each file contain information such as the satellite id and the time coverage of the orbit. The files are bundled in daily directories and, in case of 2C-ICE, are compressed. All datasets have their own file format which build on common formats as HDF4 and HDF5. To facilitate the collocation workflow, I wrote the generic class *FileSet*. It parses the file paths with regular expressions and can find all files of a dataset matching a given time period or other criteria. It allows to align the temporally-overlapping files of multiple datasets and open them in a queue. The queue is powered by parallel threads and can load files to the memory in the background while the main thread does other things such as collocating. To read the different file formats, it uses a *FileHandler* passed by the user.

FileHandler

There are many different file formats and each one of them needs its special handling. This is where the family of *FileHandler* classes are good for. They provide reading and writing routines for different file formats and are passed to a *FileSet* object to allow it to handle the files. This keeps the *FileSet* class flexible and generic and makes it possible that the users build their own *FileHandler* classes when they need to read very specific files.

Collocator

The *Collocator* class allows to collocate datasets in form of *FileSet* objects by using spatial and temporal criteria. Since collocating is an embarrassingly parallel problem, it is possible to speed-up the processing by using parallel workers. I use the efficient *BallTree* class from Pedregosa et al. (2011) since it is tested and very fast due to C-code implementation. Figure 3.2 shows the workflow of the collocator.

[1]Related classes are classes which are connected via inheritance.

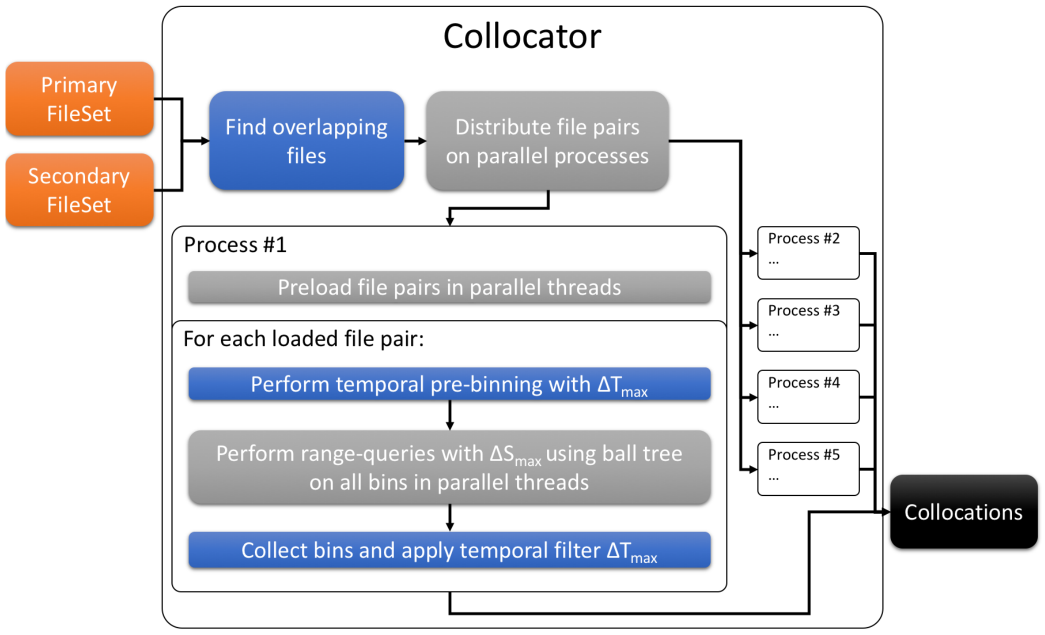

Collocator
Primary FileSet
Secondary FileSet
Find overlapping files
Distribute file pairs on parallel processes
Process #1
Preload file pairs in parallel threads
For each loaded file pair:
Perform temporal pre-binning with ΔT$_{max}$
Perform range-queries with ΔS$_{max}$ using ball tree on all bins in parallel threads
Collect bins and apply temporal filter ΔT$_{max}$
Process #2 ...
Process #3 ...
Process #4 ...
Process #5 ...
Collocations

FIGURE 3.2: The workflow of the *Collocator* class. Workflow steps are gray where parallelization is applied.

At first, it searches for all files in the input datasets that may overlap each other according to the information in their file names. This overlap search also considers the temporal criterion ΔT_{max}. The file pairs are distributed on multiple processes but grouped according to the primary file of the pair. This avoids to read the same primary file multiple times. Each process starts to read and load the passed file pairs to memory in the background while it starts with the actual collocation. If a temporal criterion is given, the data is binned with ΔT_{max} as described in subsection 2.1.1. For each bin, a ball tree with the largest dataset is created and a range query on it with the other dataset is performed. The results are collected and the final temporal filter is applied. The collocations are then returned or stored to disk using the *compact* mode.

The Collocator class is applicable to any kind of geo-referenced data and can also be used without SPARE-ICE.

Collocations

Collocations is a specialized *FileSet* class. It is designed to help storing and post-processing of collocated data. It allows collapsing and expanding of collocated data.

RetrievalProduct

The python package scikit-learn (Pedregosa et al., 2011) already offers very powerful machine learning classes such as *MLPRegressor* and *DecisionTreeClassifier*, also called estimators. However, it does not allow to store estimators directly to disk and to reuse them on other systems. Therefore, this class is a wrapper around a scikit-learn estimator class and makes it possible to store their coefficients to JSON files.

SPAREICE

SPAREICE is the class that the users should use if they want train or generate the SPARE-ICE product. It combines the *Collocator* and *RetrievalProduct* classes.

3.3 Collocator

Since the collocator (typhon) works as a stand-alone toolkit, I evaluate its performance with the collocator (atmlab) in this section. I also show further applications which are outside of the scope of SPARE-ICE.

3.3.1 Speed Comparison with Holl et al. (2010)

I compare the speed of collocator (typhon) with collocator (atmlab) implemented by Holl et al. (2010) and upgraded by John et al. (2012). SPARE-ICE needs two collocated datasets. For the training mode, MHS is collocated with 2C-ICE and AVHRR in two stages, while in the retrieval mode only collocations between MHS and AVHRR are required. For this evaluation, I only collocate MHS& 2C-ICE ($\Delta T_{\max}$ = 10 min, $\Delta S_{\max}$ = 7.5 km) and MHS & AVHRR ($\Delta T_{\max}$ = 30 sec, $\Delta S_{\max}$ = 7.5 km). Table 3.2 shows the benchmarks for both collocators when collocating these datasets from October to November 2008.

TABLE 3.2: Runtime of the collocator in SPARE-ICE (atmlab) on different datasets for a data period of 31 days.

Toolkit	**MHS & 2C-ICE**	**MHS & AVHRR**
collocator (atmlab)	75 min	806 min
collocator (typhon)	11 min	110 min
collocator (typhon), 5 processes	2 min 35 sec	24 min

As one can see, collocator (typhon) is almost 7 times faster than collocator (atmlab) when collocating 2C-ICE with MHS. If collocator (typhon) uses five processes it is more than 28 times faster. It scales only near-proportionally due to overhead costs, e.g. some secondary files must be read twice.

3.3.2 Further Applications

Warm Collocations Between MHS Instruments[2]

When searching for SNOs between satellites, they normally occur more often in polar regions than at lower latitudes. For example, when collocating the observations of the MHS instruments on the NOAA18 and Metop-A satellites between 10th and 20th December 2013 with a $\Delta T_{\max}$ = 5 min and $\Delta S_{\max}$ = 5 km, one finds only collocations at 70-80° North and South, respectively. This results from the different equator-crossing times of the satellites.

However, this is a problem if one wants to use the found collocations to calibrate the two MHS sensors. For example, the third channel of MHS measures at a frequency of 183 $\pm$1 GHz and is therefore very sensitive to water vapour in the atmosphere.

[2]I collaborated with Marc Prange on the application in this section. You may find further information on this topic in his master's thesis.

The air at the poles is very cold, so the observed brightness temperatures rarely exceed 245 K. This might be a problem because we do not have any data to investigate differences between the instruments at higher brightness temperatures. How could we get this data? How could we find *warm* collocations?

Both MHS instruments have a daily global coverage, i.e. they pass each location at the Earth once per day (exceptions are the so-called pole-gaps). The reason that we do not find any collocations outside the polar region is that the temporal filter is too strict. Metop-A passes the same locations as NOAA18 but several hours later. If we ease the temporal filter (e.g. $\Delta T_{\text{max}} = 8$ hours), we would find much more collocations. But the atmosphere may change in the time between the measurements and they would not be comparable any more. We need an additional filter that guarantees that a scene has not changed significantly between two measurements.

The Spinning Enhanced Visible Infra-Red Imager (SEVIRI) on the geostationary Meteosat Second Generation 10 spacecraft (MSG10) provides two infra-red channels (5 and 6), which are sensitive to similar atmospheric features as the third channel of MHS. Since MSG10 is a geostationary satellite, SEVIRI only observes one side of the Earth's globe but with a high temporal resolution. We can use the SEVIRI measurements as reference to decide whether two collocated scenes have changed. The crucial criterion is then not whether two collocated scenes are close in time but rather whether SEVIRI does not see a strong difference between them. To make this work, we must perform two collocation searches. At first, we collocate each MHS instrument with SEVIRI using strict SNO criteria (ΔT_{max} = 5 min, ΔS_{max} = 5 km). We also apply a cloud filter here, since the sensitivity of the infra-red channels on clouds is different than of MHS channels. Afterwards, we collocate the resulting collocations of both MHS instruments together with a looser temporal criterion of ΔT_{max} = 8 hours but additional criteria for SEVIRI. Channels 5 and 6 are not allowed to change more than 1 K between the collocated measurements (see Figure 3.3 for the workflow).

We limit our collocation search for this case study roughly on the Atlantic (40° N - 40° S, 72° W - 0° E) and on the time period from 10th to 20th December 2013. Figure 3.4 shows an example of some found collocations on a map. They are only found where the difference of the SEVIRI channels is not greater than 1 K. The choice of 1 K is arbitrary but must guarantee a comparability of the scenes and should not be too strict. Otherwise too few collocations are found.

Figure 3.5 shows the collocations found with the traditional SNO search and with our new approach. Since atmospheric regions which are stable for several hours are mostly subsidence regions and therefore dry, high brightness temperatures were measured. Both approaches complement each other and could help to allow a better calibration than one of them alone.

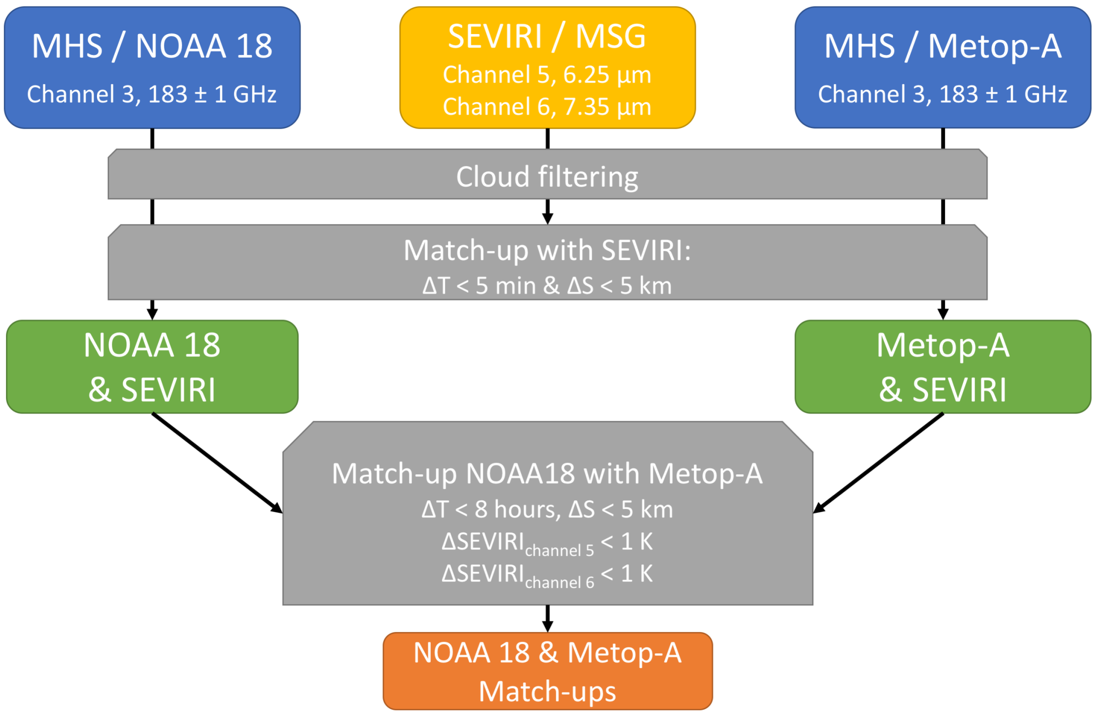


FIGURE 3.3: The workflow to find warm collocations (match-ups) of MHS on NOAA18 and Metop-A with the of SEVIRI.

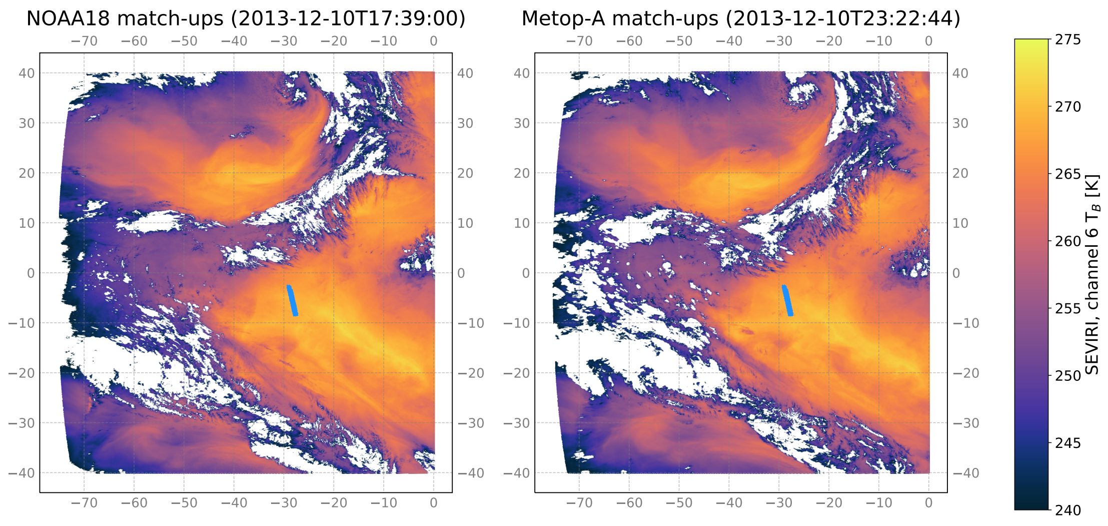

NOAA18 match-ups (2013-12-10T17:39:00)
Metop-A match-ups (2013-12-10T23:22:44)
SEVIRI, channel 6 T_B [K]

FIGURE 3.4: An example for collocations found with the workflow in Figure 3.3. Both images show the brightness temperatures for the SEVIRI channel 6. The blue points indicate the collocations with NOAA18 (left side) and Metop-A (right side). Both images have a time difference of around six hours.

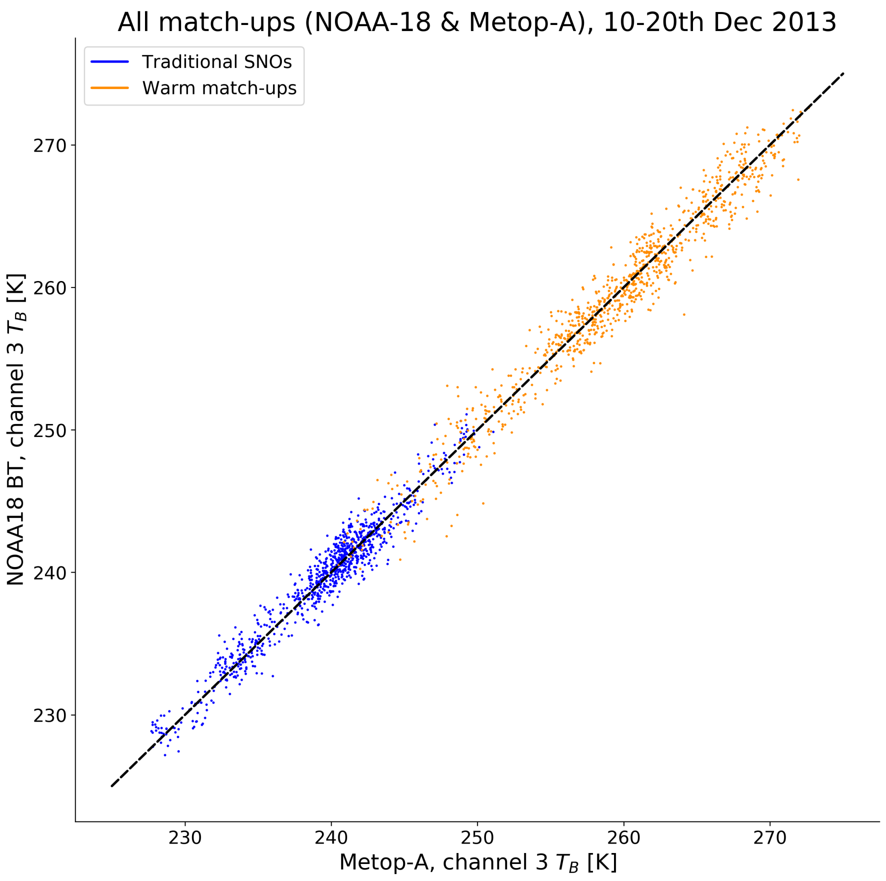


FIGURE 3.5: The brightness temperatures of the third channel of MHS from collocations between Metop-A and NOAA-18 from 10th to 20th December 2013. Only the pixels with a scan position around nadir ± 5 were taken. The blue collocations were found by the traditional method while the orange points are coming from our new approach. The black dashed line is the diagonal.

Comparison of Point and Area Measurements

Collocations are a great opportunity to compare area with point measurements. For example, the Hamburg Ocean Atmosphere Parameters and Fluxes from Satellite Data (HOAPS) dataset provides i.a. the sea surface temperature as 6-hours composites (Andersson et al., 2010). HOAPS measurements are on a grid with a spatial resolution of 0.5 x 0.5 degrees and representing an area up to 50 km^2. One can validate the HOAPS sea surface temperature by collocating them with point measurements from ships. The Ocean Rain And Ice-phase precipitation measurement Network (OceanRAIN) bundles measurements of eight ships of several years and provides variables such as sea surface temperature and precipitation (Klepp, 2015) . Figure 3.6 shows an example for the collocations between HOAPS and OceanRAIN with $\Delta T_{\max} = 5$ min and $\Delta S_{\max} = 25$ km for the time period from October to December 2013. Since one finds multiple OceanRAIN measurements for one HOAPS pixel, they are collapsed to one value.

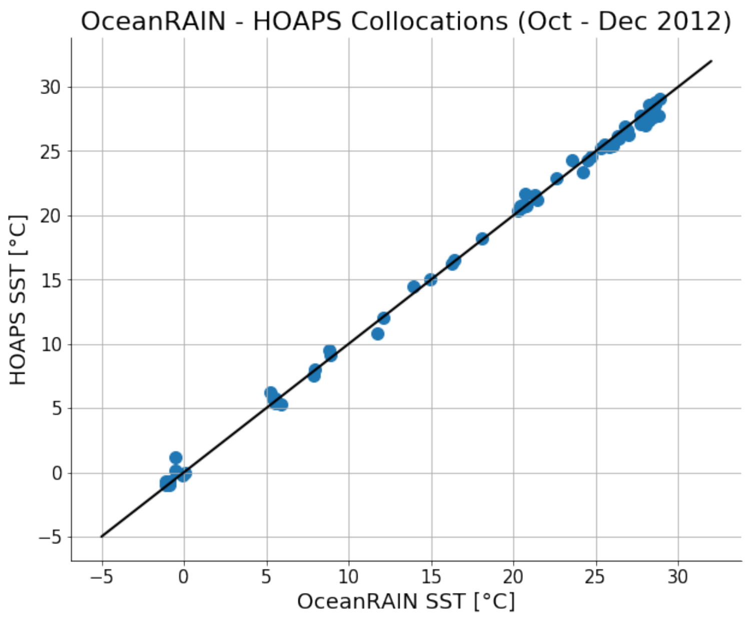


FIGURE 3.6: Found collocations for HOAPS and OceanRain.

One could use these collocations to determine the bias between HOAPS and OceanRAIN and to evaluate their measurements.

Chapter 4

Retraining of SPARE-ICE

While Holl et al. (2014) explored the synergies of different channel combinations and Li (2015) compared SPARE-ICE with other IWP datasets, I focus on reducing the remaining training error of SPARE-ICE to 2C-ICE including the mid-latitude bias. For this, I analyse the IWP retrieval in more detail and propose solutions. Later, I describe how I created the new training dataset and the set-up for experiments to improve SPARE-ICE. Finally, I present the results and discuss them.

4.1 Analysis of Retrieval Performance

4.1.1 Comparison with 2C-ICE

SPARE-ICE was trained with 2C-ICE as a target, so it cannot perform better than that because it inherits all its errors. Additionally, there is still the remaining error from SPARE-ICE to 2C-ICE after training. According to Holl et al. (2014), SPARE-ICE has a median fractional error (MFE) of 100% to 2C-ICE estimated through testing data. The fractional error for each retrieved IWP is defined as follows:

$$\mathrm{FE} = \exp\left|\ln\frac{\mathrm{IWP}_{\text{SPARE-ICE}}}{\mathrm{IWP}_{\text{2C-ICE}}}\right| - 1 \tag{4.1}$$

Thus, SPARE-ICE differs from 2C-ICE by a factor of 2 in the median. The error is not uniformly constant, the retrieval of higher IWP values between 10^3 and 10^4 g/m^2 works better: the MFE is around 50% here.

However, both training and testing datasets for SPARE-ICE are selected from collocations between 2C-ICE, MHS and AVHRR. Hence, it is difficult to evaluate the performance with the testing dataset alone because it might have sample issues due to collocating. For example, the collocations does not contain all viewing angles of MHS and AVHRR owing to the satellite orbit geometry. It is not possible to test the performance of SPARE-ICE on them directly (Holl et al., 2014). Therefore, instead of limiting the error analysis on the testing dataset, it is also necessary to compare the statistical features between 2C-ICE and SPARE-ICE.

Li (2015) compared the IWP zonal mean of SPARE-ICE (atmlab) and 2C-ICE (see Figure 4.1). SPARE-ICE would be expected to perform similar to 2C-ICE if its training was successful. While it retrieves similar values in the tropics and the South Pole, a strong dry bias can be seen in the mid-latitudes and towards the North Pole.

The bias suggests that SPARE-ICE underestimates the IWP in mid-latitudes regions systematically.

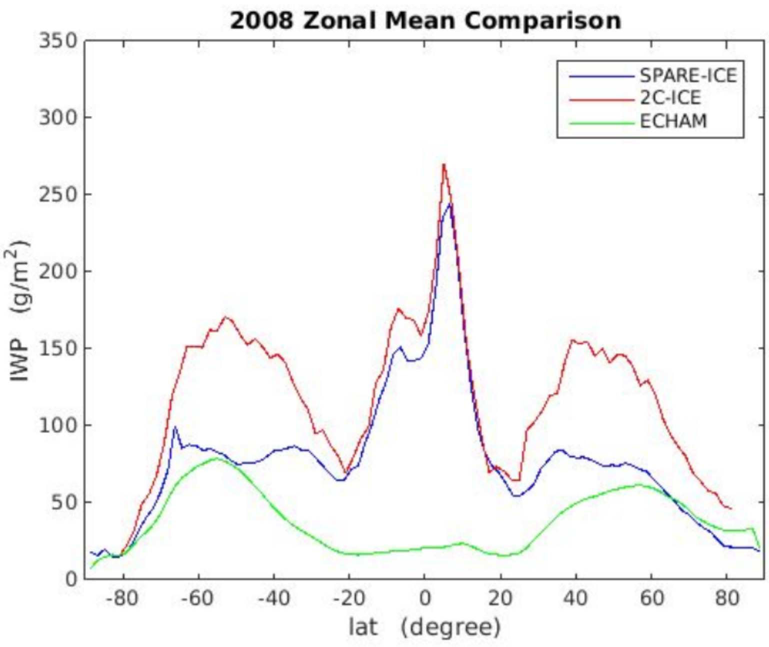


FIGURE 4.1: Zonal mean comparison of SPARE-ICE (atmlab) and 2C-ICE for the year 2008 (Li, 2015, Fig. 14).

4.1.2 Possible Error Sources and Solutions

There are several reasons which might explain the remaining error between SPARE-ICE and 2C-ICE. I address them here and propose solutions.

Underfitting

The performance of an ANN is sensitive to the chosen hyperparameters, e.g. the weights during initialization or the learning rate. It happens often that ANNs get stuck in a local minimum so they do not improve further during training. Holl et al. (2014) stated they trained an ensemble of 5 ANN each with 10 hidden neurons and chose the best performing network. The remaining MFE of 100% could signify that the ANN underfits the data due to ill-chosen hyperparameters or too few hidden neurons. I try to avoid this problem by training hundreds of ANNs with different hidden layers, number of hidden neurons and regularization parameters for each composition of inputs.

Sample Error Due to Collocations

The training and testing datasets of SPARE-ICE (atmlab) are based on collocations amongst 2C-ICE, MHS and AVHRR for the year 2007. After applying an inhomogeneity filter and thinning it to a equally-geographically distribution, $3.4 \cdot 10^5$ points are left for training, validation and testing. Since collocating is already a very strict selection process, errors might arise due to sampling. For example, the training dataset does not contain viewing angles for MHS above 33° because no collocations are found for them. Also the majority of the training data is around noon or midnight localtime due to the equator-crossing times of the satellites. ANNs are able to extrapolate to data points they have not seen during the training but it is difficult to test the performance of the extrapolation without having a complete testing dataset.

It is not possible to avoid this problem completely since we have no training data besides our collocations. However, to reduce the sampling effects of collocating, I use more years of data (January 2007 to August 2010). I also apply the cross-validation during training to make more data available during the training process (see section 2.2.2).

Scene Inhomogeneity

The footprint of 2C-ICE pixels are much smaller than MHS pixels, so one must average multiple IWP values before the training. As mentioned in subsection 2.4.3, a filter is applied to reduce the error of inhomogeneous scenes. I constrain this filter even more by only allowing a standard deviation of half the size of the average of the pixel values. This reduces the number of collocations significantly but provides the ANN with cleaner data.

Variation of AVHRR

Because AVHRR pixels are also smaller than MHS pixels, they must be averaged for one MHS pixel as well. If the the relationship between IWP and the signal observed by the AVHRR channels is not linear, this can lead to a bias due to averaging. Hence, I add the standard deviation of the AVHRR channel 5 to the input data.

Land-Sea Difference

SPARE-ICE might perform differently over land than over sea surfaces. To investigate this, a land-sea mask is applied and used as auxiliary input for the retrievals. The land-sea mask has a 5 arc minute resolution and is available through the typhon package.

Channel Differences

There are other IWP retrievals using the difference of channels from passive radiometers (e.g. Vivekanandan et al. (1991)), i.e. it could be an important feature. Theoretically, an ANN could learn from the training data itself that the difference between channels is important and calculate it by itself. In practice, it is easier to provide the ANN directly with this information. Hence, I use as additional inputs the difference between AVHRR channel 5 and 4 plus the difference between MHS channel 5 and 3.

Microwave Surface Channels

Holl et al. (2014) did not use the 89 and 157 GHz channels of MHS since they are sensitive to surface and would require further emissivity data to be useful. However, the land-sea mask might provide a coarse surface emissivity information and the channels could provide useful auxiliary information.

4.1.3 Ice Cloud Classifier

As mentioned before, the IWP regressor is trained in log space. Thus it cannot retrieve IWP values of 0 g/m^2. This is unrealistic and gives SPARE-ICE a wet bias. To fix this, Holl et al. (2014) added an second ANN which retrieves the probability whether a scene contains an ice cloud or not. If the probability reaches a threshold, the IWP retrieved is set to zero. Since this is a typical classification task, deciding whether there is an ice cloud or not, I replace the second ANN with a random forest of decision tree classifiers (RFC). Instead of a continuous probability for an ice cloud between 0 and 1, there will be only a discrete ice cloud flag: true or false. This facilitates the training since RFC are not sensitive to feature-scaling and do not need hyperparameter tuning (see section 2.3). It also enables an easier estimation of the importance of the input data for the ice cloud classification.

Holl et al. (2014) used as target for the network whether the IWP from 2C-ICE is larger than 10 g/m^2, then a scene was considered to contain an ice cloud. For the reasons of consistency with 2C-ICE, I rather choose 0 g/m^2 as threshold. I also test whether microwave channels help to identify ice clouds, Holl et al. (2014) did not consider microwave channels as input for the ice cloud classifier.

4.2 Experiments

4.2.1 Setup

I retrain the IWP regressor and ice cloud classifier for SPARE-ICE (typhon). To compare my results with SPARE-ICE (atmlab), I also use the MHS and AVHRR instruments on NOAA18. For the IWP regressor, I use the same inputs as Holl et al. (2014) except for the surface temperature since it would add a huge dependency to SPARE-ICE. On the basis of the analysis from the previous sections, I add the following inputs:

- **sea_mask**: The ANN knows whether a pixel was measured above a sea (water) or land surface.
- **avhrr_std**: The ANN gets the standard deviation of channel 5 of the collapsed AVHRR pixels as input.
- **difference**: The ANN gets the difference between AVHRR channel 5 and 4 plus the difference between MHS channel 5 and 3 as input.
- **mhs_surface**: The ANN gets the MHS surface channels 1 and 2 as input.

As described in subsection 2.2.3, it is difficult to determine the importance of each input of ANNs. Hence, I use the best performing ANN with all additional inputs as a *baseline* experiment and drop subsequently one additional input field per training session. This allows a rough estimation of the importance of the dropped input by comparing the resulting model performance with the baseline model. I also train ANNs without any additional input and call the best-performing model *original* since it has almost the same inputs as SPARE-ICE (atmlab).

I train different combinations of hyperparameters for the IWP regressors with a 5-fold cross-validation. I use two measures to identify the best network: the MFE of the testing data and the error of the annual zonal mean to 2C-ICE. Looking only at the MFE might not be sufficient to determine the real error of SPARE-ICE to 2C-ICE since the MFE is calculated with data coming from collocations. However, since the zonal mean error is more expensive to calculate (one has to process one whole year of SPARE-ICE), I do it only for the model configurations with the smallest MFE.

The ice cloud classifier does not need a strong tuning of hyperparameters. The main concern is to prevent overfitting by setting the maximum depth for trees to small values. I train one random forest classifier and use the calculated feature importance to select the most important input data. The performance of classifiers can be evaluated by calculating the rate for true or false positives and negatives and showing them in a confusion matrix.

4.2.2 Training Data

I find the new training data with the collocator. In a first step, 2C-ICE and MHS datasets are collocated with $\Delta T_{\max} = 10$ min and $\Delta S_{\max} = 7.5$ km. The collocations are collapsed to MHS and the standard deviation and number of collapsed 2C-ICE pixels are stored. In a second step, the 2C-ICE&MHS collocations are collocated with AVHRR. Since AVHRR and MHS are on the same satellite and their scans are almost synchronized, a shorter temporal criterion can be applied here to make the search more efficient ($\Delta T_{\max} = 30$ sec). Afterwards, the inhomogeneity filter is applied on the data: at least 10 2C-ICE pixels must have been collapsed to one MHS pixel and their standard deviation must be smaller than the half of their average (see Figure 4.2 for the workflow and input fields).

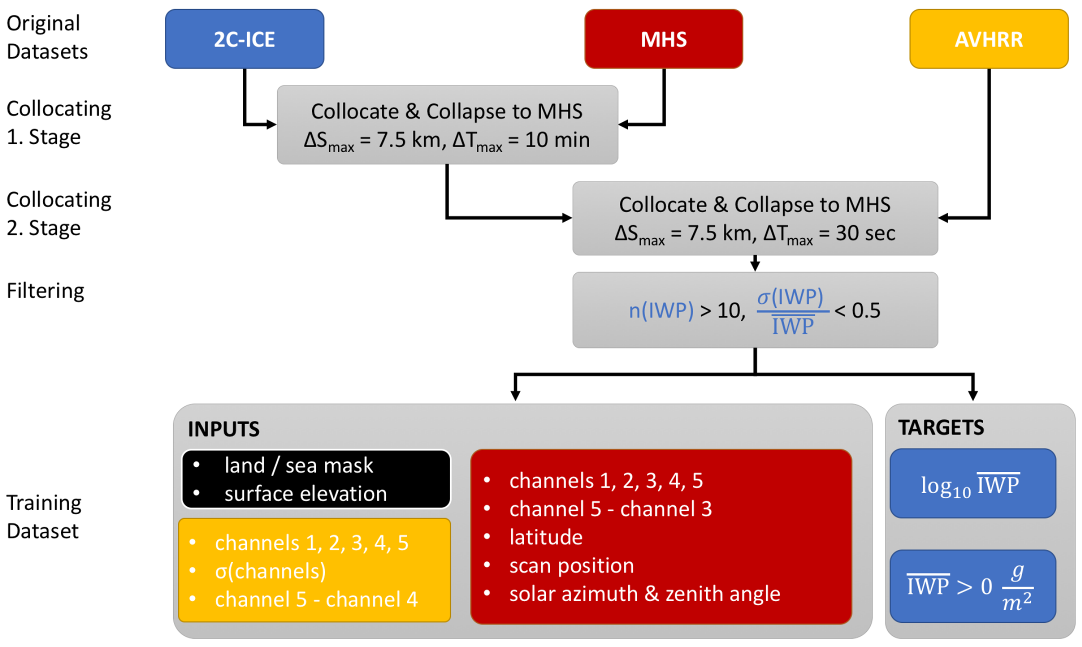

Original Datasets
Collocating 1. Stage
Collocating 2. Stage
Filtering
Training Dataset
2C-ICE
MHS
AVHRR
Collocate & Collapse to MHS ΔS_{max} = 7.5 km, ΔT_{max} = 10 min
Collocate & Collapse to MHS ΔS_{max} = 7.5 km, ΔT_{max} = 30 sec
$n(IWP) > 10, \frac{\sigma(IWP)}{\overline{IWP}} < 0.5$
INPUTS
land / sea mask
surface elevation
channels 1, 2, 3, 4, 5
σ(channels)
channel 5 - channel 4
channels 1, 2, 3, 4, 5
channel 5 - channel 3
latitude
scan position
solar azimuth & zenith angle
TARGETS
$\log_{10} \overline{IWP}$
$\overline{IWP} > 0 \frac{g}{m^2}$

FIGURE 4.2: The collocation workflow for the training dataset of SPARE-ICE. The inputs in the black box come from auxiliary datasets. The targets used for the IWP regressor or ice cloud classifier are in the right corner at the bottom.

Since the IWP regressor is trained in log space, all collocations with an IWP of 0 g/m^2 are filtered out. Table 4.1 shows the amount of the data left for the training and evaluation. Although I use 3.5 years of data, while Holl et al. (2014) used only one, I have not significantly more data due to the stricter inhomogeneity filter.

TABLE 4.1: Number of collocations used for training and evaluation of the IWP regressor. To train the ice cloud classifier, twice as much collocations can be used since they include IWP values of 0 g/m^2. The numbers for SPARE-ICE (atmlab) are taken from Holl et al. (2014).

Datasets	SPARE-ICE (atmlab)	SPARE-ICE (typhon)
Training	134.000	263.000
Validation	66.000	65.000
Testing	140.000	110.000
Total	340.000	483.000

4.3 Results

In this section, I present the results for the best IWP regressor and ice cloud classifier.

4.3.1 IWP Regressor

Figure 4.3 shows the MFE of the best-performing models of all experiments to the training target 2C-ICE as a function of the logarithmic IWP of 2C-ICE. The MFE of SPARE-ICE (atmlab) is also shown as a reference. SPARE-ICE (typhon) has a much lower MFE for IWP values above 6 g/m^2 (up to 30%) than SPARE-ICE (atmlab) while the other is more accurate on lower IWP values. The *baseline* model has a better performance than the *original* model, even if the difference is not large. The other models, which had a single input dropped, perform basically as well as *baseline*.

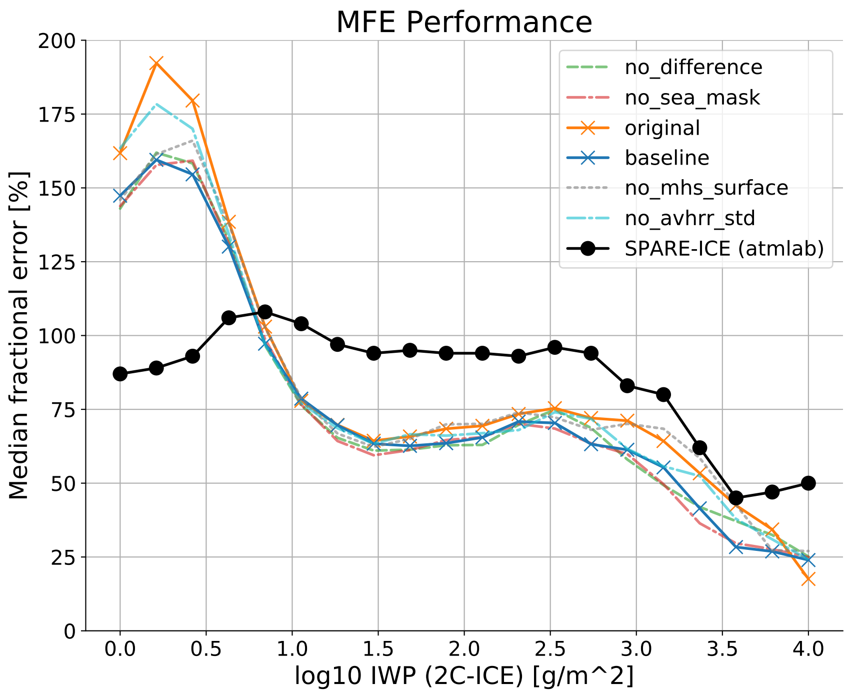


FIGURE 4.3: The median fractional error of both SPARE-ICE products to 2C-ICE for the testing data. Please note: The MFE values of SPARE-ICE (atmlab) were only projected from Holl et al. (2014). They may contain small differences to the original and are based on another selection of testing data.

Figure 4.4 shows the annual zonal mean for 2008 of both SPARE-ICE products and 2C-ICE. The *original* model still underestimates the IWP at the mid-latitudes but not so strong as SPARE-ICE (atmlab). The *baseline* model performs better than all other models and has no large differences to 2C-ICE except for the North Pole where it overestimates the IWP. The *no_sea_mask* model (*baseline* without the sea mask) has the largest difference on the performance in the north hemisphere to *baseline*, followed by *no_avhrr_std*. However, all models do not differ much from 2C-ICE at the South Pole and in the tropics except for *no_difference* which overestimates the tropical IWP. This and the last figure show that the *original* model has a lower MFE and zonal mean error to 2C-ICE than SPARE-ICE (atmlab).

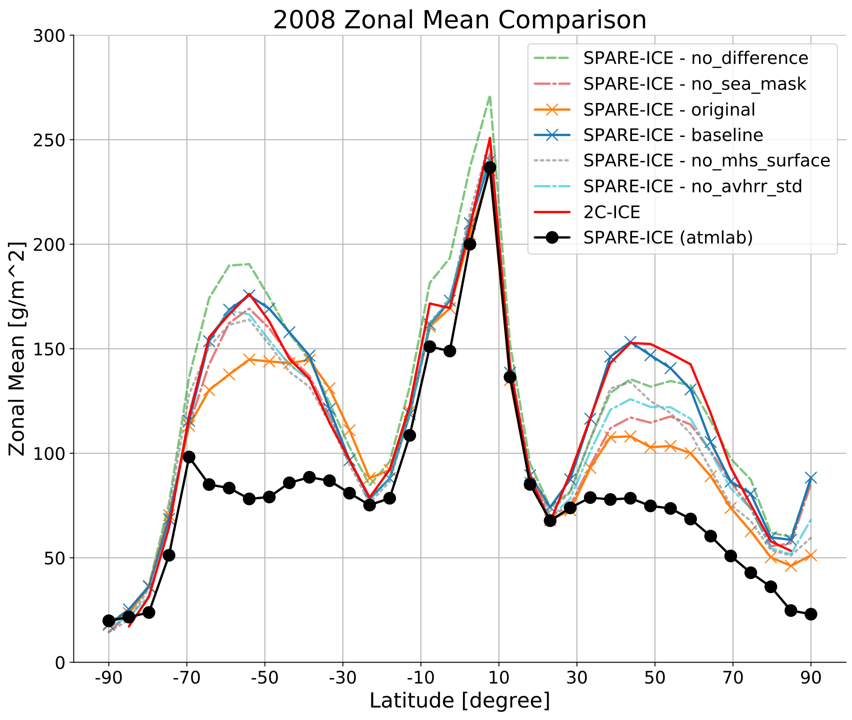
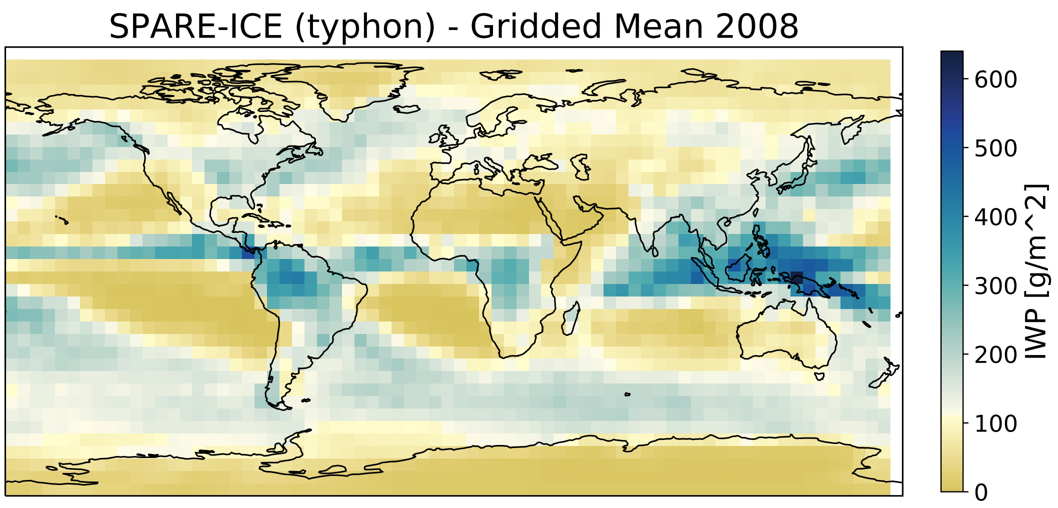


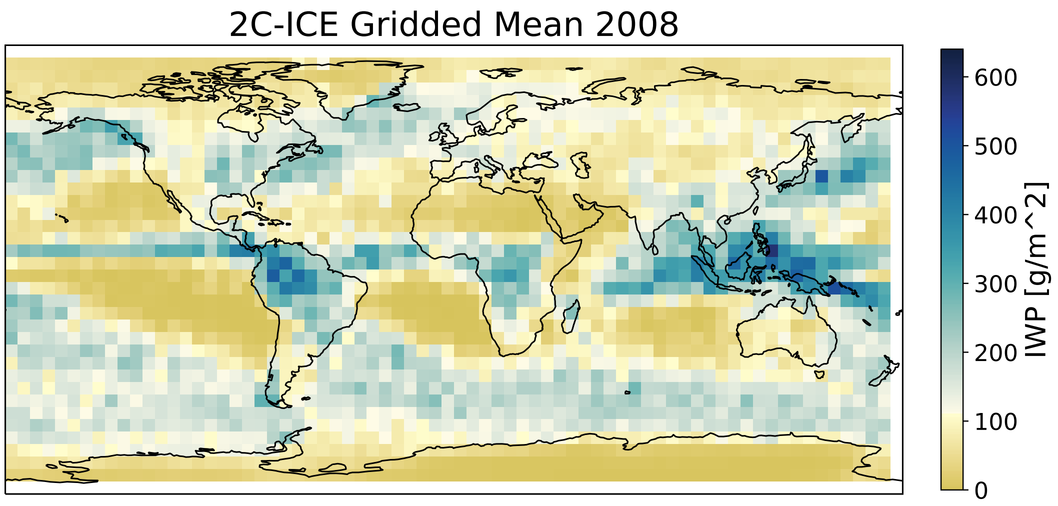
FIGURE 4.4: The zonal mean of both SPARE-ICE products and 2C-ICE for the year 2008. Please note: The MFE of SPARE-ICE (atmlab) were only projected from Li (2015). They may contain small differences to the original.

The *baseline* model is the best-performing model; it has a low MFE and zonal mean error to 2C-ICE. The comparison of the annual gridded mean shows how well it can reproduce the same global distribution of IWP, especially in the tropics (see Figure 4.5). Anyway, it overestimates the IWP in dry zones such as subsidence regions or Antarctica. This can also be seen in Figure 4.6, which shows the comparison between 2C-ICE and the *baseline* model as a heatmap. The majority of the testing points are along the diagonal, i.e. they are retrieved with a low error. The spread of the points gets larger for smaller IWP values.

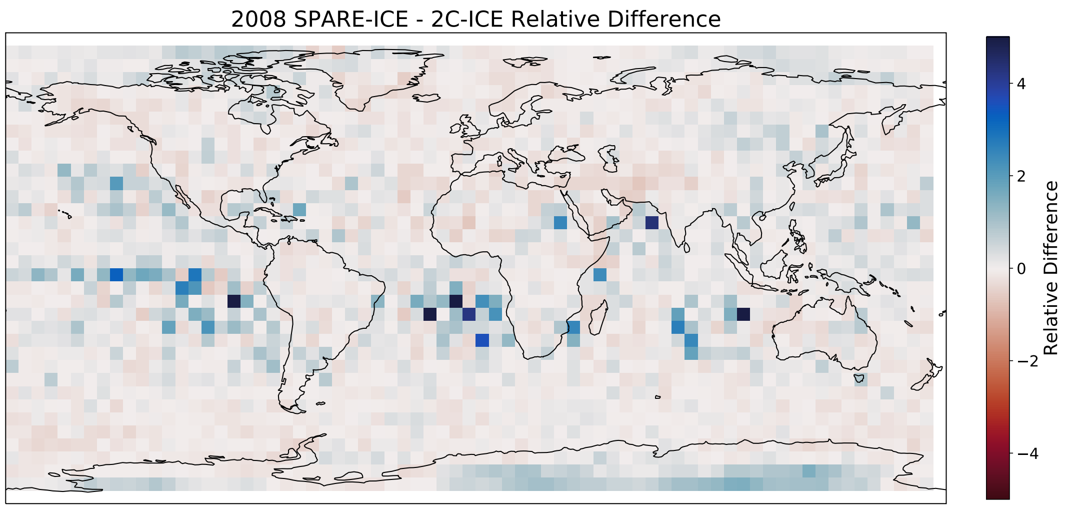
Figure 4.7 shows the bias between the *baseline* model and 2C-ICE changing with the logarithmic IWP. A land-sea mask was applied to see where the model performs better. The model generally overestimates the IWP for values above 10 g/m^2 and performs better over sea than over land.



(A)



(B)



(C)

FIGURE 4.5: (A and B) IWP of SPARE-ICE (typhon) *baseline* and 2C-ICE averaged over 5x5° grid cells. (C) The relative difference between the averaged IWP values. Grid cells are blue where SPARE-ICE is larger than 2C-ICE.

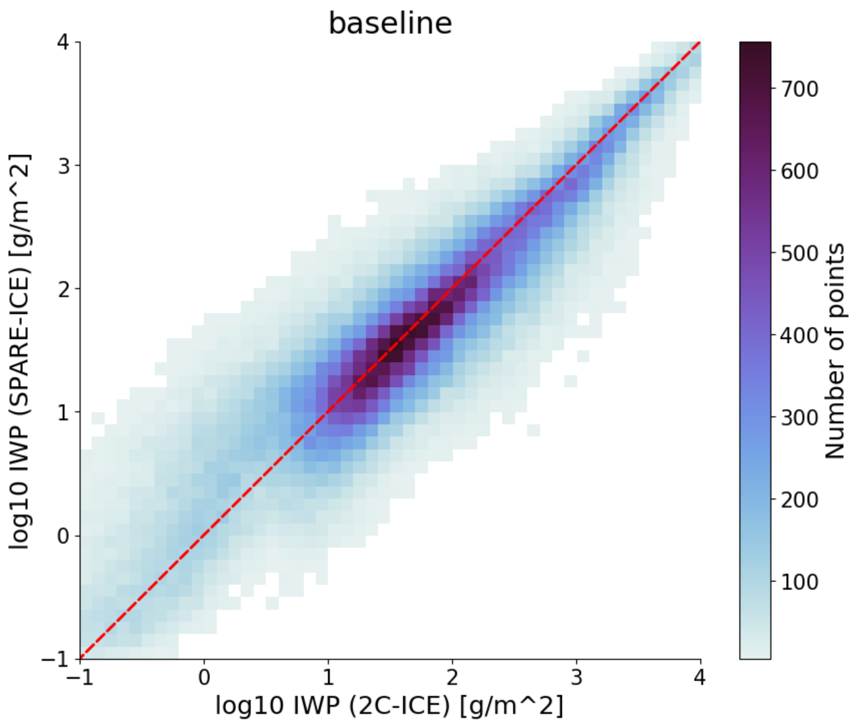


FIGURE 4.6: Comparison between 2C-ICE and *baseline* model using the testing data. The red, dashed line is the diagonal.

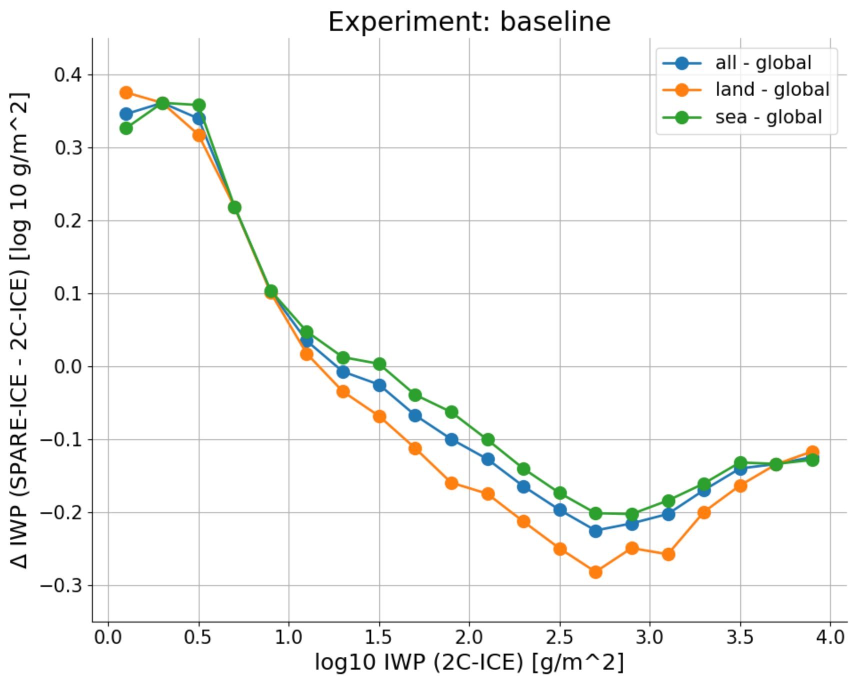


FIGURE 4.7: The bias between the *baseline* model and 2C-ICE changing with the logarithmic IWP of 2C-ICE. The green line shows the performance for pixels over sea, the orange line shows it over land surfaces and the blue line is the averaged performance. The model overestimates the IWP for smaller values and underestimates it for higher ones. The performance for IWP values over land above 10 g/m^2 is worse than over sea surfaces.

4.3.2 Ice Cloud Classifier

Figure 4.8 shows the confusion matrix with the classification errors of the retrained ice cloud classifier on testing data. The false positive rate is 12% while the false negative rate is 6%. This means, the error to predict a non-existing ice cloud is higher than the opposite.

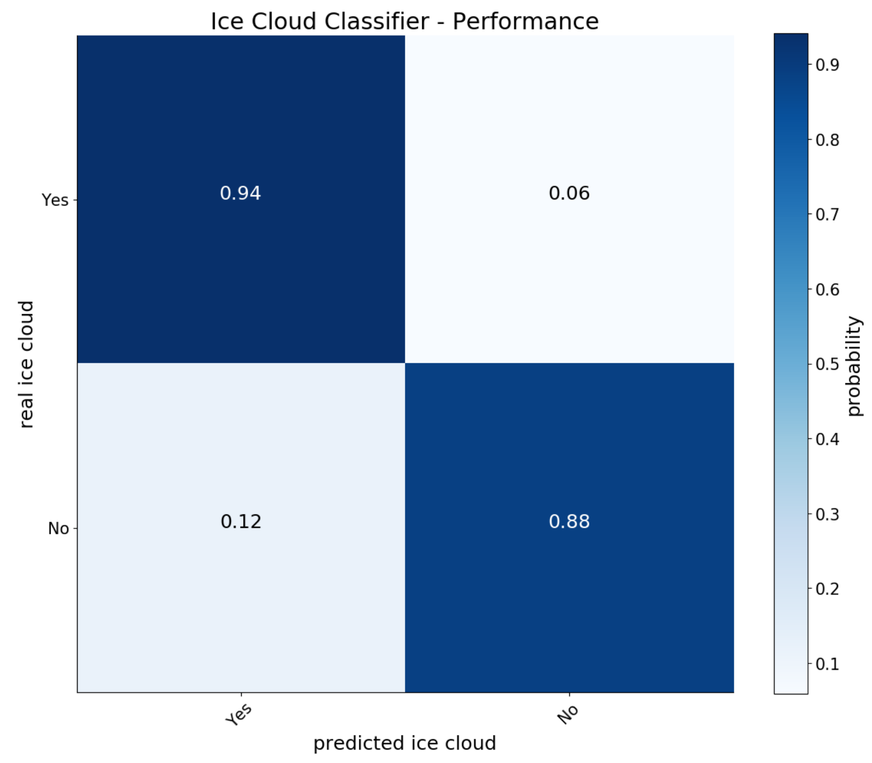


FIGURE 4.8: The confusion matrix of the best performing ice cloud classifier in SPARE-ICE (typhon) created with the testing dataset.

Using the Gini index, the ice cloud classifier identifies the brightness temperatures from AVHRR channel 5 and MHS channel 3 as the most important features to detect an ice cloud in a scene (see Figure 4.9).

4.4 Discussion

In this section, I summarize and interpret the results of the experiments with the IWP regressor and ice cloud classifier. I assess the importance of each additional input and investigate the reasons for the better performance of the *baseline* model over SPARE-ICE (atmlab).

4.4.1 IWP Regressor

The better performance of the *original* model over SPARE-ICE (atmlab) in Figure 4.3 and Figure 4.4 shows that more data and stricter filters lead to an overall improved performance of SPARE-ICE. This is plausible since more precise training data reduce the effect of noise and larger samples help the ANN to generalize better. However, the *original* model still has a strong bias to 2C-ICE.

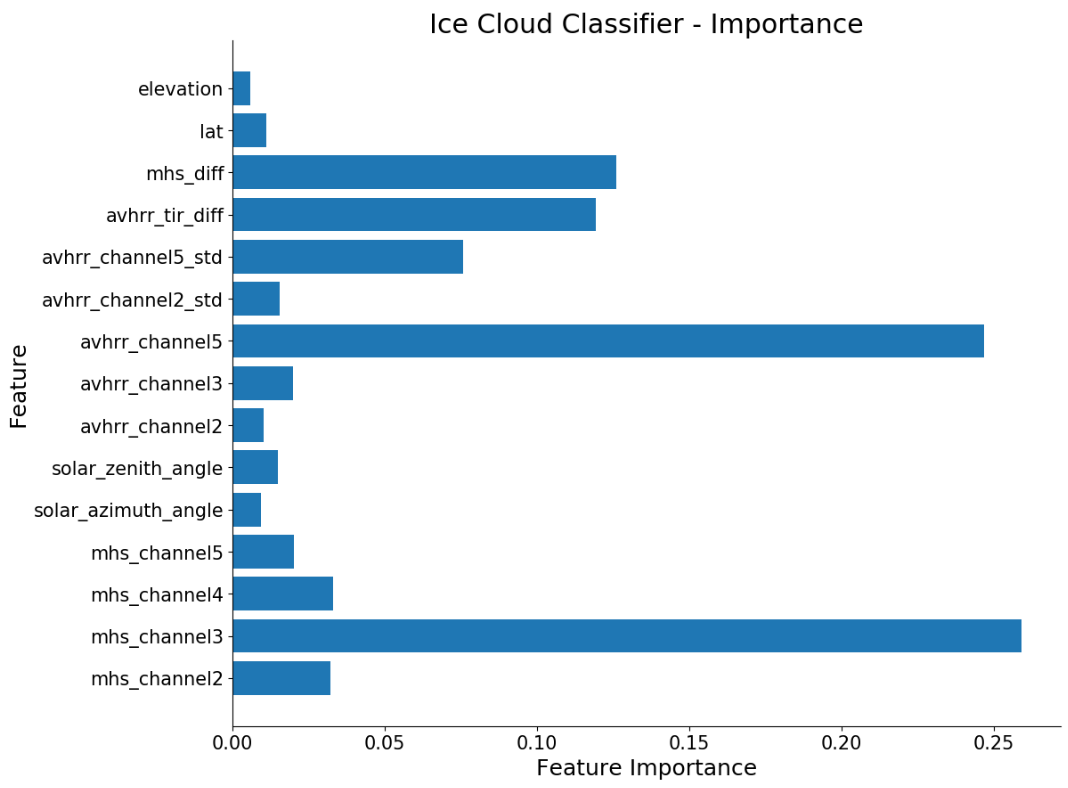


FIGURE 4.9: The feature importances of the ice cloud classifier indicating which feature was used most often for splitting the training dataset at the tree nodes.

The *baseline* model performs far better; the bias is mostly eliminated. All additional inputs seem to be important for this bias reduction even when they reduce it mostly at northern latitudes. The strongest influence has the land sea mask. Since SPARE-ICE underestimates the IWP over land stronger than over sea (see Figure 4.7) and the northern hemisphere contains more land surfaces than the southern hemisphere, the sea mask might be helpful to diminish this bias.

The second strongest effect has the added standard deviation of the collapsed AVHRR measurements. This might be related to the beam filling error known from remote sensing of rainfall (Lim and Han, 1996). The signal of ice clouds might have a non-linear relation to the observed brightness temperature of infra-red channels. Hence, collapsing or averaging would lead to an underestimation of this signal.

The difference between AVHRR and MHS channels help the model to avoid over- or underestimation of the IWP in the southern or northern hemisphere, respectively. It makes sense to consider it as useful information since traditional passive IWP retrievals are often based on the difference between two observing channels (Vivekanandan et al., 1991).

It is not clear why the surface channels of MHS seem to be so important for the IWP retrieval. It might be due to their sensitivity to convective clouds (Sol Galligani et al., 2017).

Looking only at the MFE in Figure 4.3 does not help to identify how good a model performs globally. For example, *no_sea_mask* has almost the same MFE as *baseline* but has a strong mid-latitude bias in the northern hemisphere. The difference of the distributions of the testing data (collocations) and the real data could be a reason for this. They are simply not sufficiently representative for the real data.

In conclusion, I choose the *baseline* model as standard configuration for SPARE-ICE (typhon) since it shows the smallest error to 2C-ICE and show good and reasonable meteorological results.

4.4.2 Ice Cloud Classifier

The error of the ice cloud classifier has the same magnitude as the error of the ice cloud probability network by Holl et al. (2014). However, the error rates are swapped. The network has a higher false negative rate than false positive rate; that is the opposite situation with the retrained ice cloud classifier. The ice cloud probability network has also a slightly better performance. This makes sense since Holl et al. (2014) uses surface temperatures which can help with the detection of thin ice clouds.

According to the calculated feature importance by the classifier, the third MHS channel and the AVHRR channel are the most important features for the ice cloud detection (see Figure 4.9). Since the trained random forest is difficult to visualize (it contains 20 trees), I trained a single decision tree classifier with a similar performance to present the first classifying questions (see Figure 4.10).

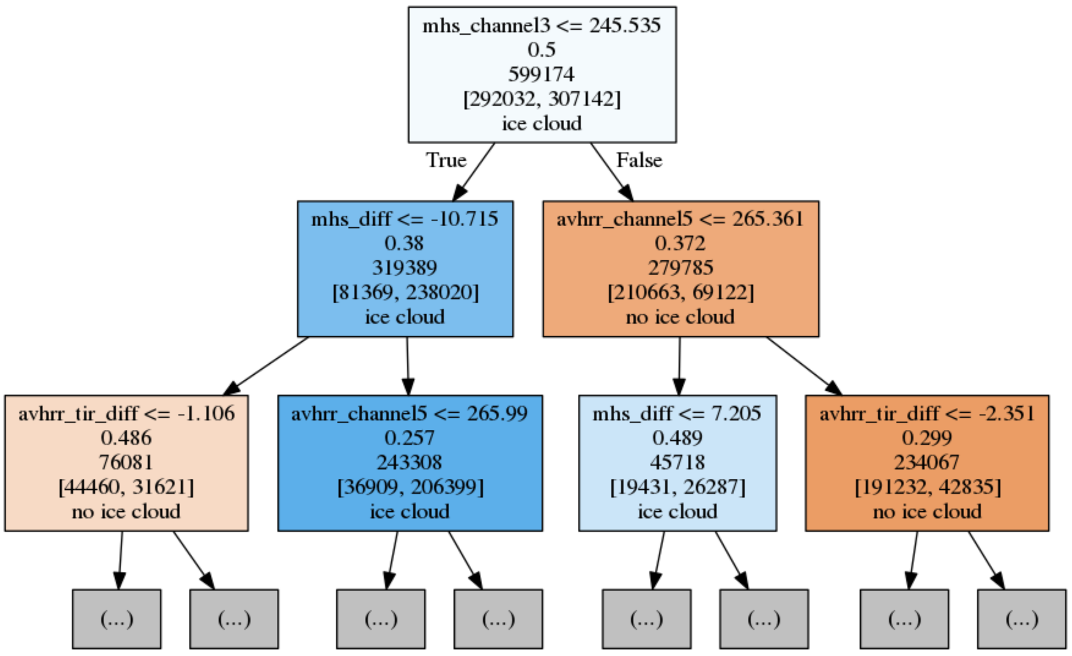


FIGURE 4.10: The decision tree for the ice cloud classifier. If a condition in a node is true, the condition in its left children node is also checked otherwise the right children node is chosen. This continues until one reaches a leaf node and the input is finally classified.

The absolute and difference values used for the classification conditions are well known from the literature (e.g. Buehler et al. (2007)) and thus comprehensible.

Chapter 5

Conclusion

In this thesis, I reimplemented and retrained SPARE-ICE to reduce its mid-latitude bias to 2C-ICE and make it more user- and developer-friendly. The toolkit is publicly available via the typhon package.

The reimplemented collocator, which is included in the SPARE-ICE toolkit, can be used as a stand-alone tool to find collocations amongst any type of geo-referenced data. I redesigned its classes using object-oriented programming principles and decreased its run-time by a factor of 7 by using a more efficient ball tree, distance metric and parallel reading of files. The new toolkit allows to use multiple processes natively so the runtime can be simply scaled for larger datasets. I showed how it performs with multiple collocation searches on large data volumes coming from the geostationary satellite SEVIRI and how it can be used to compare point with area measurements.

To reduce the mid-latitude bias of SPARE-ICE to 2C-ICE, I retrained it with more data and added input information. Additional inputs such as a sea mask, the standard deviation of the collapsed AVHRR measurements and surface channels of MHS helped to reduce the the mid-latitude significantly. The model was also provided with the differences between some AVHRR and MHS channels to enhance the training. The resulting model shows an overall improved performance for larger IWP values and is able to reproduce the annual global distribution of IWP very similar to 2C-ICE. However, inaccuracies remain for dry zones such as subsidence regions or Antarctica.

There are further improvements which should be explored in the future. Since some input channels of SPARE-ICE might see the surface especially when observing thin clouds, the error might be reduced by using skin temperatures or emissivities from the surface as auxiliary inputs. Since SPARE-ICE is generally applicable to all MHS/AMSU and AVHRR instruments, it can be trained with other satellite combinations (e.g. MHS and AVHRR from NOAA19). This might expand the SPARE-ICE dataset and allow inter-comparisons amongst them. Simulations with radiative transfer models could allow to fill gaps in the training dataset such as the missing limb viewing angles.

The search for collocations between MHS and AVHRR is the bottleneck of the SPARE-ICE application; it uses 70 % of the running time. This results from the high resolution of the AVHRR instrument. For each orbit, almost 5’500’000 AVHRR pixels must be collocated with 300’000 MHS pixels. This is very time-consuming even for optimized collocation methods. However, MHS and AVHRR are on the same satellite and their scan lines might have an offset which could be predicted. This can be used as a heuristic to build a simple function which maps AVHRR pixels onto MHS pixels and accelerate the collocating significantly.

Bibliography

Amante, C. and Eakins, B. (2009). ETOPO1 1 Arc-Minute Global Relief Model: Procedures, Data Sources and Analysis. *NOAA Technical Memorandum NESDIS NGDC-24*, (March):19.

Andersson, A., Fennig, K., Klepp, C., Bakan, S., Graßl, H., and Schulz, J. (2010). The Hamburg Ocean Atmosphere Parameters and Fluxes from Satellite Data – HOAPS-3. *Earth System Science Data Discussions*, 3(1):143–194.

Bentley, J. L. and Maurer, H. A. (1980). Efficient worst-case data structures for range searching. *Acta Informatica*, 13(2):155–168.

Bentley, J. L., Stanat, D. F., and Williams, E. H. (1977). The complexity of finding fixed-radius near neighbors. *Information Processing Letters*, 6(6):209–212.

Breiman, L. (1999). Random Forests. *Machine Learning*, 45(5):1–35.

Buehler, S. A., Kuvatov, M., Sreerekha, T. R., John, V. O., Rydberg, B., Eriksson, P., and Notholt, J. (2007). A cloud filtering method for microwave upper tropospheric humidity measurements. *Atmospheric Chemistry and Physics*, 7(21):5531–5542.

Cao, C., Weinreb, M., and Xu, H. (2004). Predicting simultaneous nadir overpasses among polar-orbiting meteorological satellites for the intersatellite calibration of radiometers. *Journal of Atmospheric and Oceanic Technology*, 21(4):537–542.

Cracknell, A. P. (1997). *Advanced very high resolution radiometer AVHRR*. CRC Press.

Deng, M., MacE, G. G., Wang, Z., and Okamoto, H. (2010). Tropical composition, cloud and climate coupling experiment validation for cirrus cloud profiling retrieval using cloudsat radar and CALIPSO lidar. *Journal of Geophysical Research Atmospheres*, 115(17):1–18.

Dolatshah, M., Hadian, A., and Minaei-Bidgoli, B. (2015). Ball*-tree: Efficient spatial indexing for constrained nearest-neighbor search in metric spaces.

Eliasson, S., Holl, G., Buehler, S. A., Kuhn, T., Stengel, M., Iturbide-Sanchez, F., and Johnston, M. (2013). Systematic and random errors between collocated satellite ice water path observations. *Journal of Geophysical Research: Atmospheres*, 118(6):2629–2642.

Gardner, M. W. and Dorling, S. R. (1998). Artificial neural networks (the multilayer perceptron) - a review of applications in the atmospheric sciences. *Atmos. Environ*, 32(14-15):2627–2636.

Holl, G. (2013). *Remote Sensing of Ice Clouds: Synergistic Measurements and Radiative Transfer Simulations*. Number November.

Holl, G., Buehler, S. A., Rydberg, B., and Jiménez, C. (2010). Collocating satellite-based radar and radiometer measurements - Methodology and usage examples. *Atmospheric Measurement Techniques*, 3:693–708.

Holl, G., Eliasson, S., Mendrok, J., and Buehler, S. A. (2014). SPARE-ICE: Synergistic ice water path from passive operational sensors. *Journal of Geophysical Research*, 119(3):1504–1523.

Hornik, K., Stinchcombe, M., and White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366.

John, V. O., Holl, G., Buehler, S. A., Candy, B., Saunders, R. W., and Parker, D. E. (2012). Understanding intersatellite biases of microwave humidity sounders using global simultaneous nadir overpasses. *Journal of Geophysical Research Atmospheres*, 117(2):1–13.

Kindler, E. and Krivy, I. (2011). Object-oriented simulation of systems with sophisticated control. *International Journal of General Systems*, 40(3):313–343.

Kleespies, T. J. and Watts, P. (2006). Comparison of simulated radiances, Jacobians and linear error analysis for the Microwave Humidity Sounder and the Advanced Microwave Sounding Unit-B. *Quarterly Journal of the Royal Meteorological Society*, 132(621 C):3001–3010.

Klepp, C. (2015). The oceanic shipboard precipitation measurement network for surface validation — OceanRAIN. *Atmospheric Research*, 163:74–90.

Kohavi, R. (1995). A Study of Cross-Validation and Bootstrap for Accuracy Estimation and Model Selection A Study of Cross-Validation and Bootstrap for Accuracy Estimation and Model Selection. *International Joint Conference on Artificial Intelligence*.

Kottayil, A., John, V. O., Buehler, S. A., and Mohanakumar, K. (2016). Evaluating the diurnal cycle of upper tropospheric humidity in two different climate models using satellite observations. 8(4).

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2323.

Li, S. (2015). *SPARE-ICE Satellite Cloud Ice Retrievals*. Master thesis, University Hamburg.

Lim, H.-S. and Han, D. (1996). An Approach to Remedy Beam Filling Error of TMI. *Journal of the Meteorological Society of Japan*, 74(4):415–423.

Liu, T., Moore, A. W., and Gray, A. (2006). New Algorithms for Efficient High Dimensional Non-parametric Classification. *Journal for Machine Learning Research*, 7(c):1135–1158.

Louppe, G., Wehenkel, L., Sutera, A., and Geurts, P. (2013). Understanding variable importances in forests of randomized trees. *Advances in Neural Information Processing Systems 26*, pages 431–439.

Mili, H., Mili, F., and Mili, A. (1995). Reusing Software: Issues and Research Directions. *IEEE Transactions on Software Engineering*, 21(6):528–562.

Nagle, F. W. and Holz, R. E. (2009). Computationally efficient methods of collocating satellite, aircraft, and ground observations. *Journal of Atmospheric and Oceanic Technology*, 26(8):1585–1595.

Olden, J. D. and Jackson, D. A. (2000). Torturing data for the sake of generality: How valid are our regression models? *Écoscience*, 7(4):501–510.

Olden, J. D. and Jackson, D. A. (2002). Illuminating the "black box": A randomization approach for understanding variable contributions in artificial neural networks. *Ecological Modelling*, 154(1-2):135–150.

Omohundro, S. M. (1989). Algorithms. pages 1–22.

Pedregosa, F., Weiss, R., Brucher, M., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, É. (2011). Scikit-learn : Machine Learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature*, 323(6088):533–536.

Safavian, S. R. and Landgrebe, D. (1991). A Survey of Decision Tree Classifier Methodology. *IEEE Transactions on Systems, Man and Cybernetics*, 21(3):660–674.

Saha, S., Moorthi, S., Pan, H.-L. L., Wu, X., Wang, J. J. J., Nadiga, S., Tripp, P., Kistler, R., Woollen, J., Behringer, D., Liu, H., Stokes, D., Grumbine, R., Gayno, G., Wang, J. J. J., Hou, Y.-T. T., Chuang, H.-y. Y., Juang, H.-M. M. H., Sela, J., Iredell, M., Treadon, R., Kleist, D., Van Delst, P., Keyser, D., Derber, J., Ek, M., Meng, J., Wei, H., Yang, R., Lord, S., Van Den Dool, H., Kumar, A., Wang, W., Long, C., Chelliah, M., Xue, Y., Huang, B., Schemm, J.-K. K., Ebisuzaki, W., Lin, R., Xie, P., Chen, M., Zhou, S., Higgins, W., Zou, C.-Z. Z., Liu, Q., Chen, Y., Han, Y., Cucurull, L., Reynolds, R. W., Rutledge, G., Goldberg, M., Saha, S., Moorthi, S., Pan, H.-L. L., Wu, X., Wang, J. J. J., Nadiga, S., Tripp, P., Kistler, R., Woollen, J., Behringer, D., Liu, H., Stokes, D., Grumbine, R., Gayno, G., Wang, J. J. J., Hou, Y.-T. T., Chuang, H.-y. Y., Juang, H.-M. M. H., Sela, J., Iredell, M., Treadon, R., Kleist, D., Delst, P. V., Keyser, D., Derber, J., Ek, M., Meng, J., Wei, H., Yang, R., Lord, S., van den Dool, H., Kumar, A., Wang, W., Long, C., Chelliah, M., Xue, Y., Huang, B., Schemm, J.-K. K., Ebisuzaki, W., Lin, R., Xie, P., Chen, M., Zhou, S., Higgins, W., Zou, C.-Z. Z., Liu, Q., Chen, Y., Han, Y., Cucurull, L., Reynolds, R. W., Rutledge, G., and Goldberg, M. (2010). The NCEP Climate Forecast System Reanalysis. *Bulletin of the American Meteorological Society*, 91(8):1015–1058.

Snyder, A. (1986). Encapsulation and inheritance in object-oriented programming languages. *ACM SIGPLAN Notices*, 21(11):38–45.

Sol Galligani, V., Wang, D., Alvarez Imaz, M., Salio, P., and Prigent, C. (2017). Analysis and evaluation of WRF microphysical schemes for deep moist convection over south-eastern South America (SESA) using microwave satellite observations and radiative transfer simulations. *Atmospheric Measurement Techniques*, 10(10):3627–3649.

Stocker, T. F., Qin, D., Plattner, G.-K., Tignor, M., Allen, S. K., Boschung, J., Nauels, A., Xia, Y., Bex, V., and Midgley, P. M. (2013). Climate change 2013: The physical science basis. Technical report. 578 pp.

Vivekanandan, J., Turk, J., and Bringi, V. N. (1991). Ice Water Path Estimation and Characterization Using Passive Microwave Radiometry.

Waliser, D. E., Li, J.-L. L. F., Woods, C. P., Austin, R. T., Bacmeister, J., Chern, J., Del Genio, A., Jiang, J. H., Kuang, Z., Meng, H., Minnis, P., Platnick, S., Rossow, W. B., Stephens, G. L., Sun-Mack, S., Tao, W.-K. K., Tompkins, A. M., Vane, D. G., Walker, C., and Wu, D. (2009). Cloud ice: A climate model challenge with signs and expectations of progress. *Journal of Geophysical Research*, 114(8):1–27.

Yan, B. and Weng, F. (2008). Intercalibration between special sensor microwave imager/sounder and special sensor microwave imager. *IEEE Transactions on Geoscience and Remote Sensing*, 46:984–995.

Acknowledgements

I would like to thank my supervisors Stefan Buehler and Ákos Horváth for the assistance during my work on this Master's thesis. I really enjoyed being part of the *rare* research group and had a wonderful time here. I also thank the other python experts and main developers of typhon, Olli and Lukas, who helped me out to write more pythonic code and taught me to limit my line length to 79 characters. Thanks to my dearest editor Lina whose English is simply natively perfect. I also would like to thank my family in Darmstadt, Duisburg, Berlin and Mainz for the great support over all the years, I could not feel happier and more privileged.

Versicherung an Eides statt

Hiermit versichere ich an Eides statt, dass ich die vorliegende Arbeit im Studiengang M. Sc. Meteorologie selbstständig verfasst und keine anderen als die angegebenen Hilfsmittel – insbesondere keine im Quellenverzeichnis nicht benannten Internet-Quellen – benutzt habe. Alle Stellen, die wörtlich oder sinngemäß aus Veröffentlichungen entnommen wurden, sind als solche kenntlich gemacht. Ich versichere weiterhin, dass ich die Arbeit vorher nicht in einem anderen Prüfungsverfahren eingereicht habe und die eingereichte schriftliche Fassung der auf dem elektronischen Speichermedium entspricht.

Unterschrift: ______________________________

Datum: ______________________________