universität wien

MASTERARBEIT / MASTER'S THESIS

Titel der Masterarbeit / Title of the Master's Thesis

The determination of cloud cover and cloud types from ground based images with pixel recognition and machine/deep learning methods

verfasst von / submitted by

Thomas Aistleitner, BSc

angestrebter akademischer Grad / in partial fulfilment of the requirements for the degree of

Master of Science (MSc)

Wien, 2021 / Vienna, 2021

Studienkennzahl lt. Studienblatt / degree programme code as it appears on the student record sheet:	UA 066 614
Studienrichtung lt. Studienblatt / degree programme as it appears on the student record sheet:	Masterstudium Meteorologie
Betreut von / Supervisor:	Ass.-Prof. Mag. Dr. Manfred Dorninger

Abstract

The influence of clouds on the atmosphere is indisputable, and therefore an accurate determination of cloud cover and cloud types is inevitable. Upcoming gaps of manual weather observations are partly replaced by remote-sensing and ground based instruments. An alternative to these instruments is the use of weather images, taken by people with their smartphones. In this thesis, several approaches from pixel recognition as well as machine and deep learning methods are used to analyse ground-based images. For the determination of the cloud cover and cloud types, different threshold methods (normalised red/blue ratio and euclidean geometric distance) and machine learning models (k-nearest neighbour, k-means clustering, support vector machine and convolutional neural networks) are applied on different datasets. The verification states that the determination of the cloud cover depends heavily on the contrast of colours in the cloud image, but the machine learning methods achieve rather accurate results. The classification of the clouds can be solved by machine learning models on trivial datasets (over 90% accuracy), but the results for complex cloud images are insufficient.

Zusammenfassung

Der Einfluss der Wolken auf die Atmosphäre und auf den Treibhauseffekt sind unumstritten. Eine exakte Bestimmung der Wolkenbedeckungen sowie eine genaue Kategorisierung der Wolkenarten somit von großer Bedeutung. Aufgrund finanzieller sowie personeller Aspekte, werden manuelle Wolkenbeobachter in den nächsten Jahren zunehmend durch Satelliten sowie bodengestütze Instrumente ersetzt. Eine Alternative zu den manuellen Beobachtungen könnten Wetterbilder darstellen, welche von Personen mit ihren Smartphones aufgenommen werden. In dieser Arbeit wird nun untersucht, wie diese Bilder mit Hilfe von verschiedene Methoden analysiert werden können. Es wurde versucht mit Methoden aus dem Bereich der Pixelerkennung (Threshold Methoden: Normalized Red/Rlue Ratio und Euclidean Geometric Distance) sowie mit Modellen aus dem Bereich des Machine Learnings (K-nearest neighbour, K-means clustering, Support Vector Machine und Convolutional Neural Networks) den Bedeckungsgrad der Wolken in den Bildern sowie auch deren Wolkengattungen zu bestimmen. Als Datengrundlage werden verschiedene Datensätze aus der Literatur verwendet sowie auch Bilder, welche während der Laufzeit des Projektes H-View an der Universität Wien gesammelt wurden. Die Ergebnisse zeigen, dass die Bestimmung des Bedeckungsgrad stark vom Kontrast der Farben in den Bildern abhängig ist. Besonders die Modelle aus dem Bereich des Machine Learnings erzielten gute Ergebnisse. Die Klassifizierung der Wolkengattungen ist ebenso mit Machine Learning Modellen mit einfachen Datensätzen (fünf Wolkengattungen) erfolgreich. Die Anwendung auf komplexere Datensätze (über zehn Wolkengattungen) zeigte jedoch, dass eine ausreichende Bestimmung aller Wolkengattungen komplexe Machine Learning Modelle sowie auch einen exakt klassifizierten Trainingsdatensatz benötigen.

Contents

1. Introduction

1.1 Motivation

Clouds have an enormous influence on the energy balance of the atmosphere. Especially in terms of climate change and greenhouse gases, water vapour is one of the biggest factors [1]. The weather forecaster also gains important information for the analysis by looking at the clouds, but the information comes mostly from images taken by satellites.

When people first started recording weather conditions, parameters such as temperature, moisture and precipitation were measured with technical equipment. Accurate information of the clouds could not be measured automatically, therefore people were trained to observe the weather at certain stations and times. This was mostly the case at airports. Due to the technological change, more equipment was explored to measure the parameters for the atmosphere. Today, a weather station measures temperature, wind, radiation, precipitation and other parameters automatically in short time intervals. Since the number of weather observers is decreasing dramatically because of personnel and financial difficulties, they are partly replaced by remote-sensing (satellite) and ground based instruments. Another reason for them to be replaced by automatic observations is the subjectivity and inconsistency of observations by humans [2]. Clouds appear in a great variety and they are divided in ten different types with subcategories depending on their height and form. It is a great challenge to capture the cloud cover and in particular the cloud types by automated methods.

One ground based instrument to measure the cloud cover is the so called ceilometer. A laser source is used to determine the height of clouds or simply the cloud base. Algorithms then calculate the cloud cover, but verification has shown that the ceilometer has a clear tendency to underestimate the total cloud cover and the low cloud cover of the sky [3].

To compensate for these upcoming gaps in weather observations, we can fall back on people interested in taking photos of the weather. Since nowadays almost everyone owns a smartphone with a camera, which can take high-resolution images and therefore photos of clouds are taken within a few seconds. The idea is to motivate people - in our context students - to take images of the all-day weather and not only of extreme appearances (e.g thunderstorms, tornadoes, large hail). These photos will then be collected and stored in a database, which will be used for the analysis of cloud cover and cloud types. This was the scientific idea behind the H-View Citizen Science project of the Faculty of Earth Sciences, Geography and Astronomy of the University of Vienna.

The aim of this master's thesis is to develop and test several methods to analyse ground based cloud images. The analysis covers the determination of the cloud cover as well as the classification of the clouds in the images. The two following approaches will be applied:

The use of threshold methods is the first approach. These methods transform the pixels of an RGB image into certain values. The transformed values are then compared to a threshold and classified as clouds or sky if the value is below or above this threshold. The normalised red/blue ratio (NRBR) and the euclidean geometric distance (EGD) are the two methods used here. [4]

The second approach includes various methods from artificial intelligence (AI). For image and classification problems, methods from the machine and deep learning field are frequently used. This thesis employs k-means clustering, k-nearest neighbour, support vector machine and convolutional neural networks.

The methods are verified with three different datasets. The SWIMSEG [5] dataset is used for the verification of the methods which determine the cloud cover. The SWIMCAT [6] and CCSN [7] serve as a basis for the classification problem, as they contain pre-labelled data. Additionally, some images, which were collected by students during the H-View project, are used to verify the methods.

1.2 H-View

The project Hydrometeors-View (H-View) was a pilot study part of the citizen science campaign from the Faculty of Earth Sciences, Geography and Astronomy. The following chapter will briefly describe the project's steps and how the idea of using machine and deep learning methods evolved.

Citizen science is a term from the field of sociology. It means the involvement of amateur scientists in scientific projects. Since the term is also included in the Oxford Dictionary, the exact definition should not be withheld:

- **citizen science**: scientific work undertaken by members of the general public, often in collaboration with or under the direction of professional scientists and scientific institutions.
- **citizen scientist**:
  * a scientist whose work is characterised by a sense of responsibility to serve the best interests of the wider community (now rare)
  * a member of the general public who engages in scientific work, often in collaboration with or under the direction of professional scientists and scientific institutions; an amateur scientist. [1]

[1] 'Citizen science' added to Oxford English Dictionary. The Daily Zooniverse. 16 September 2014

The idea for the citizen scientists in the project H-View was to use them for the collection of the cloud images. Keeping citizens motivated is the most difficult task for scientists working in the field of citizen science. As a result, if someone sends data, they should always receive feedback or a reward.

Therefore, we needed methods for the feedback. The plan was to use threshold methods to analyse the cloud cover in the photos. After that, we created an email response that was sent automatically. The users received a response minutes after sending the taken image to an email address (hview.univie@gmail.com). The images that were analysed along with the cloud cover were included in the response.

In the next step, we tried to get students interested in the project. On the Kahlenberg in Vienna, ten students agreed to do a training on cloud observation. All necessary steps for a good image were explained during the observation, and the cloud images were then sent to the email address. The students received the response minutes later and were able to provide feedback on the algorithms. Based on their feedback, the methods were then improved.

At some point during the project, it was discovered that threshold methods have their shortcoming to determine the type of the clouds. As a result, AI models have been applied.

2. Fundamentals of RGB, threshold methods and artificial intelligence

2.1 The RGB colour model

It is necessary to understand the RGB model and how an RGB image is structured in order to understand how the applied methods work.

The acronym RGB stands for red, green and blue. These are the additive primary colours, which ca be combined to create any other colour. The RGB cube is a three-dimensional cube created by combining the colours.

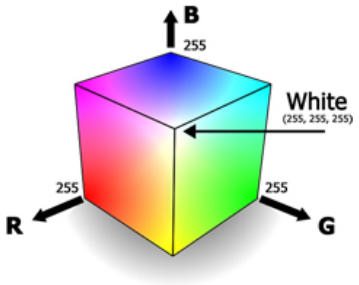


Figure 1: Illustration of RGB cube with red, blue and green in the corners[1]

This cube is illustrated in figure 1. The colours red, green and blue are located in each corner and range from 0 to 255. The higher the value, the more intense the colour in the combination will be. The RGB pixel is a three-dimensional point in a cube with a specific position that is determined by its values. Pixel values range from (0,0,0) to (255,255,255). The colour red has the value (255,0,0), green has (0,255,0) and blue (0,0,255). In the context of this thesis, other important colours are black(0,0,0), white(255,255,255), and grey (255,255,255). (125,125,125). As a result, sky pixels should be close to (0,0,255) and cloud pixels should be close to (0,0,255). (255,255,255).

An image is basically made up of columns and lines of pixels. These pixels define a specific point in the image that contains colour information. The image's resolution increases as the number of columns and lines in the image increases.

The images are implemented in Python in this thesis, and each RGB value of a pixel is transferred to an array. For example, a cloud image with a resolution of 800x600 pixels will be converted to an array with (800,600,3). The image's columns and rows are defined by the values 800 and 600. The depth of the array made up of the colours red, green, and blue is three.

[1] Source image: http://matlab.izmiran.ru/help/toolbox/images/color4.html

2.2 Threshold methods

The threshold methods are the first algorithm discussed. These are simple and effective methods, but they are limited to complex images with a wide range of colours.

The distinction between clear sky and clouds is obvious to human observers. The colour is the most important factor. Cloud particles and air molecules scatter sunlight in different ways, so they appear in different colours. The sky is mostly blue, and the clouds are mostly white or grey, depending on how much moisture is present in the clouds. The sky and clouds can appear in different shades of red at sunrise and sunset. Blue, red, and green make up a certain percentage of all colours. The intensity of the colours in a single pixel is used in threshold methods, and these values are compared to a fixed or variable threshold. A pixel is classified as cloud or sky when its value is above or below the threshold. The image channels (RGB values) in a image have a lot of mathematical relationships. Some of these relations can be found in the work of Chauvin et al. (2015) [4]. The red-blue ratio, as well as the euclidean geometric distance, are used in this thesis. Despite the fact that these methods are simple to programme and the algorithms are not computationally intensive, they produce high-quality results on images that only contain clouds.

2.3 Artificial Intelligence

Artificial Intelligence (AI) is a branch of computer sciences that focuses on the development of intelligent machines capable of solving complex problems in the same way that humans do. In Chollet, 2018 [8] the author describes it as "the effort to automate intellectual tasks normally performed by humans". Since human intelligence is complex and difficult to understand, the tasks that computers perform are usually simpler algorithms. In the 1950s, the field AI was used to run early chess programmes. Speech recognition, classification algorithms, learning and planning tasks are some of the most well-known AI applications today.

Machine and deep learning are key words in the field of AI learning algorithms. Especially in the last few years, AI has made great progress. AI models have become more popular as computing power and the availability of data has increased. Machine and deep learning programme codes are simple to implement into programmes, due to online platforms like Github (https://github.com/) and Towardsdatascience (https://towardsdatascience.com/).

In addition, AI is also getting popular in meteorology. Young et al. (2005) [9] claims, that AI forecast systems have outperformed traditional regression models in a number of studies. Increasing the use of AI in the meteorology community, particularly among students, can be beneficial. The Zentralanstalt für Meteorologie und Geodynamik (ZAMG) in Austria has its own working group, which is dedicated to implement and test machine learning models to improve weather forecasts [10].

Another example: Model output statistics (MOS), which uses linear regression to downscale numerical weather prediction (NWP) forecasts results, is partly improved with machine learning methods [11].

Figure 2 shows the connection between AI, machine (ML) and deep learning (DL). ML and DL are subcategories of AI and DL is part of ML. The next few chapters will go deeper into the field of ML and DL.

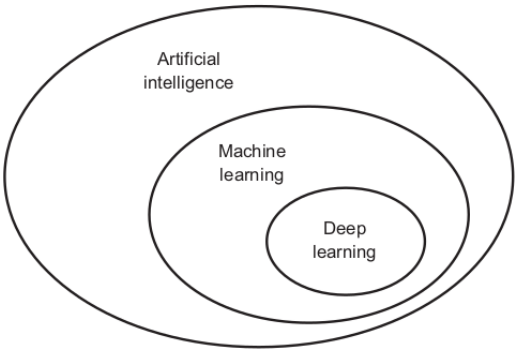


Figure 2: Connection of AI, ML and DL [8]

2.3.1 Machine Learning

There are two definitions in Géron, A.'s book [12] about machine learning that describe ML in a few sentences:

- Machine Learning is the field of study that gives computers the ability to learn without being explicitly programmed. (Arthur Samuel, 1959)
- A computer program is said to learn from experience E with respect to some task T and some performance measure P; if its performance on T, as measured by P, improves with experience E. (Tom Mitchell, 1997)

The definition in 1959 shows that ML already existed before the revolution of computing power and the availability of data in the last 20 years. Arthur Samuel coined the term "machine learning" for the first time. Most of the statistical methods used in machine learning were developed and refined at that time. It was not until the 1990s that computers had enough processing power and data to solve certain problems using machine learning techniques.

In classical programming, the programmer enters rules and data into a program. The data will be processed according to these rules and the outcome is the result. In ML, however, the data and answers are the input, while the rules are the output. These rules will then be applied to new data. [8]

Figure 3 demonstrates the procedure.

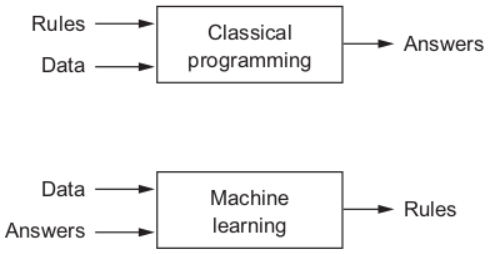


Figure 3: Difference between classical programming and machine learning models [8].

Therefore, machine learning programs are not explicitly programmed, but they are trained from data and answers. The model generates rules based on statistical structure found in the data. The well-known linear regression algorithm is an example of a machine learning algorithm. The system can predict a target numeric value simply by inputting data, such as stock price. [8]

A large amount of data must be incorporated into the model in order to develop these rules. This is the major disadvantage of these models: insufficient amount of data does not result in accurate models.

2.3.2 Deep learning

Deep learning (DL) is a sub-genre of machine learning. The deep in deep learning is derived from the word depth and does not refer to a deeper understanding of machine learning algorithms. In DL the models built up with several or hundreds or even thousands of layers. The depth of the DL model increases as the number of layers in the model increases, hence the adjective deep. The data is repeatedly sent through the layers and the model learns from the training set correcting itself through every iteration.

Deep learning is now used in a variety of applications, but it is mostly used in the background. In social media, for example, DL models are responsible for user-oriented advertising. [8]

Here are a few areas where deep learning has advanced in recent years:

- Image classification
- Speech recognition
- Handwriting transcription
- Text-to-speech conversion
- Digital assistance like Amazon Alexa
- Autonomous driving

These few examples demonstrate how far deep learning has progressed in tasks that were previously performed by humans. DL, or AI in general, will be able to perform even more tasks in the future as computer power increases and more data becomes available.

2.4 Machine and deep learning models

This section gives an overview of the models which are used to determine the cloud cover and the cloud types of the images. First, the machine learning models k-nearest neighbour (KNN), k-means clustering (kmeans) and the support vector machine (SVM) are described. Next, the neural networks (NN), the convolutional neural network (CNN) and its different layers are explained.

2.4.1 Supervised, unsupervised and semi-supervised models

AI offers a variety of different machine and deep learning models. Each model has its own algorithm and learning process. The models are categorised into three different types: supervised, semi-supervised and unsupervised learning models.

A supervised learning model is using labelled data during the learning process. The machine and deep learning models always have prior knowledge of the outcome. Therefore, the goal of supervised learning methods is to approximate the relationship between input and output of the data. Classification problems are typically solved with supervised learning methods. The term 'supervised' is a reference to the supervision of the learning process because it knows the correct answers and optimises itself to make predictions.

Examples for supervised learning models:

- Linear Regression
- k-Nearest Neighbour (KNN)
- Support Vector Machine (SVM)
- Decision Trees and Random Forest
- Neural Networks (NN) and Convolutional Neural Networks (CNN)

However, in unsupervised learning the outcome of the training data is unknown. There is no labelled data or prior knowledge, which is used to correct or verify the results of the algorithm. The model tries to recognise patterns and features out of the data, a common task within unsupervised learning is clustering of data. An example giving in Géron, 2019 [12] is the detection of groups of similar visitors on a website. Data of the visitors is given: their gender, interests or the place of residence. The machine learning algorithms can then sort the data into certain clusters and the programmer can detect different groups.

Examples for unsupervised learning models:

- K-Means Clustering
- Principal Component Analysis
- Hierarchical Cluster Analysis
- Expectation Maximisation

Semi-supervised learning algorithms are between supervised and unsupervised models. During training, these models use a small amount of labelled data and a large unlabelled dataset. In the learning process the model trains itself on the labelled data and then tries to label the samples from the unlabelled dataset, resulting in 'pseudo-labelled' data. The labelled and pseudo-labelled data are then combined and create a unique algorithm which combines the feature of supervised and unsupervised models. These kind of models are often used for large datasets which only have few labels.

Examples for semi-supervised models:

- Generative models
- Low-density separation
- Graph-based methods

2.4.2 K-nearest neighbour

Supervised machine learning methods have shown great results to solve classification problems. The k-nearest neighbour algorithm is part of the these methods. This ML algorithm makes it picks by choosing a k number of data points (neighbours) around the sample. The choice of the neighbours depends on the value of 'k' and on the distance of the data points to the sample.

There are different approaches of how the neighbours are selected. The most common method is to measure the distance from the picked sample to the neighbouring points. If k equals one for example, the nearest neighbour is picked for the classification. The higher k, the more neighbours are considered and the greater the distance of points which are taken into account.

Figure 4 shows examples of a k-nearest neighbour classification process. First, three different classes are created with random numbers (Class A, B and C). Second, a random number defines a data sample (black point). The k-nearest neighbour method is then used to assign this sample to a different number of neighbours (k = [1,2,3,5]). In the upper left image, the sample is assigned to class B, as only one neighbour is considered (black circle). With k=2 the prediction is again class B because the red point is further away from the sample than the green one. When three neighbours are considered (k=3) two red points are inside the black circle and therefore the new prediction is class C, as the number of red points exceeds the number of the other ones. The same applies when k=5: now four red points are inside the black circle and the prediction is again class C.

The example shows that the number of the value k is important for the classification. If the number of neighbours is too low, the model is too specific and generalises the data poorly, it is also sensitive to noise. On the contrary, if k is too large, the model generalises the data too much and cannot make good classifications.

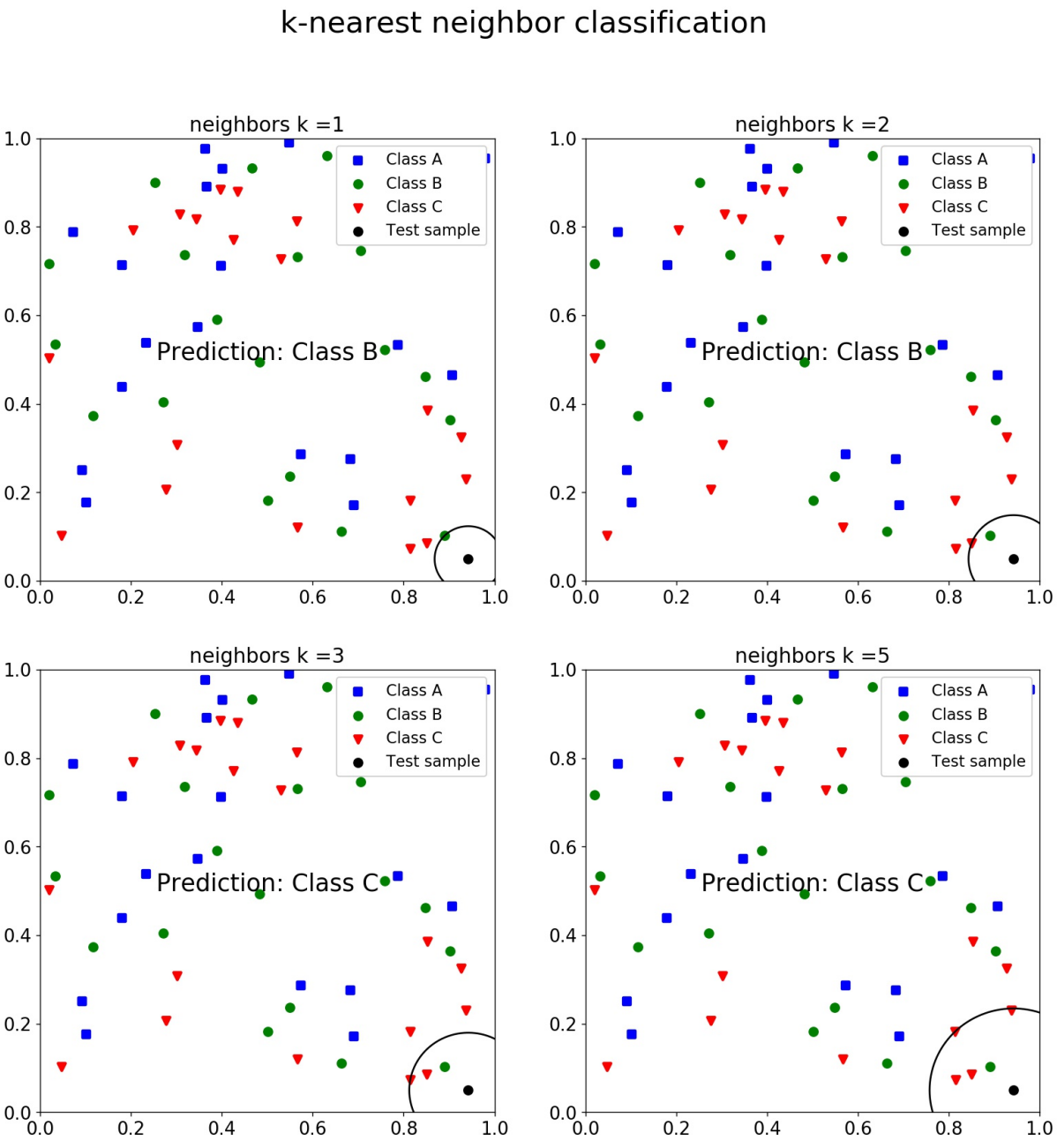


Figure 4: Four examples of k-nearest neighbour classifications. One test sample is part of three different classes and the prediction of the classification method is printed in the centre of the images.

2.4.3 K-means clustering

In contrast to the k-nearest neighbour classification k-means clustering is an unsupervised machine learning algorithm. No prior knowledge of the dataset is required. No training and evaluation of the model needs to be done. The main reason for the use of these methods is to get an intuition about the structure of the data. This method tries to cluster the dataset in 'k' numbers of groups. As in the previous section, the parameter 'k' is also a number.

The algorithm of the k-means clustering method first defines 'k' numbers of centres of the dataset, this is called centre-based clustering. The algorithm begins with a guess about the solution. This step is then repeated with a previously defined number and the result of centres will be the output. To find the nearest points to the k number of centres, the algorithm minimises the mean squared distance to its nearest centre. [13]

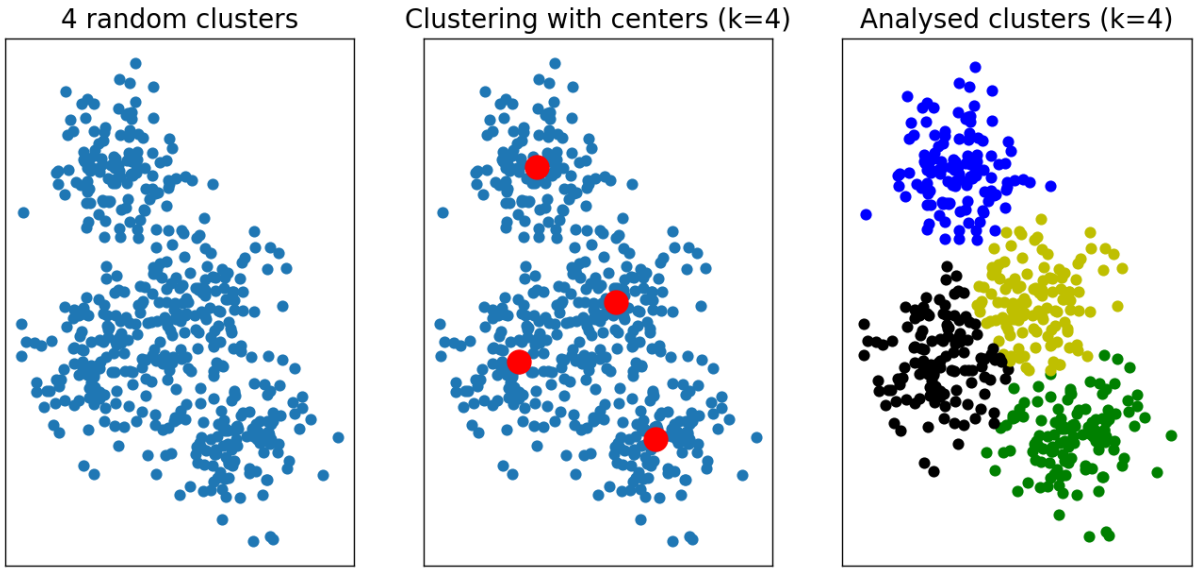


Figure 5: Cluster analysis with k-means clustering. Four random clusters are analysed with k-means clustering method. Centres are marked in red in the second image and each colour of the sample in the last image corresponds to one of the four clusters

In figure 5 one can see an example of a clusters analysis with the k-means algorithm. The left picture shows a randomised but cluster-like distribution of points. The next step is to search for a number of centres with the previously defined number 'k'. In this case, the number of clusters is four. The last step is now, to find all the points which belong to each centre. Usually this step is done by minimising the the mean squared distance.

2.4.4 Support vector machine

The support vector machine (SVM) is part of the supervised learning models. Back in 1995 a work by Cortes et al. (1995) [14] has already shown that the SVM can be applied to solve image recognition problems. In this thesis the SVM is used for the determination of the cloud

cover and for the solution of the classification problem. It thus represents an alternative to the convolutional neural network.

The idea behind the SVM is to separate into different classes as efficiently as possible using a hyperplane. There are several ways to divide those samples into two classes, as shown in the example in figure 6. With the best hyperplane and maximum margin, the support vector machine seeks out the best way to separate those two samples. In this case, dividing the samples with the optimal hyperplane is relatively simple.

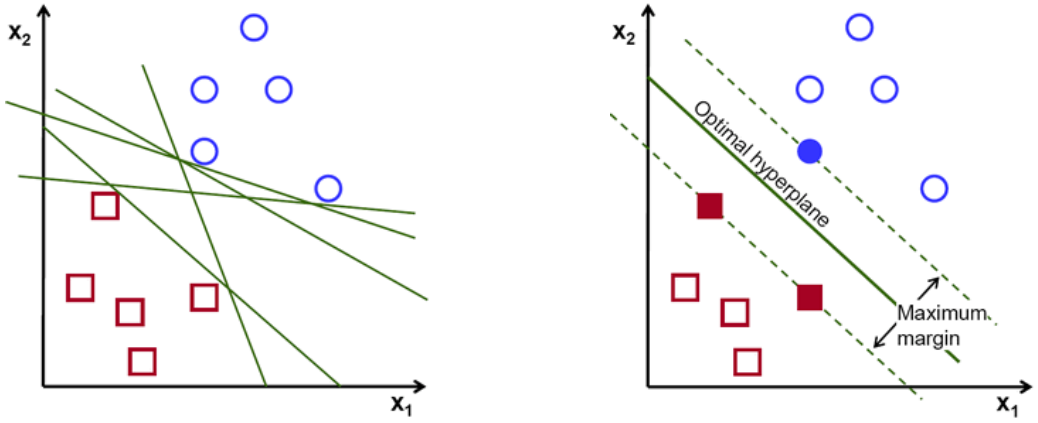


Figure 6: Left image: linear separable data samples with five possible hyperplanes. Right image: Optimal hyperplane with maximum margin, support vectors are filled symbols.[2]

However, the great advantage of the SVM is its ability to locate the hyperplane in an N-dimensional space that separates the data points. In figure 7 the data samples are arranged in a disordered way and therefore cannot be separated in two dimensions. In this case, the SVM is transforming the hyperplane in the R^3 or another N-dimensional space. This transformation is performed by kernel functions, so called non-linear kernels. [15]

Common non-linear kernels for the SVM are:

- Polynomial
- Gaussian radial basis function
- Hyperbolic tangent

Another important feature of the SVM is the margin. It depends on the distance between the support vectors and the hyperplane and can be small or large. The data points closest to the hyperplane are the support vectors. These points are the filled squares and circles on the right side of figure 6. The distance between the support vectors and the hyperplane is the margin. The greater the distance between the points, the better they can be separated from one another, and the classification becomes more precise. To get the best classification, the SVM tries to maximise this distance.

[2]Source image: www.towardsdatascience/support-vector-machine-introduction-to-machine-learning-algorithms

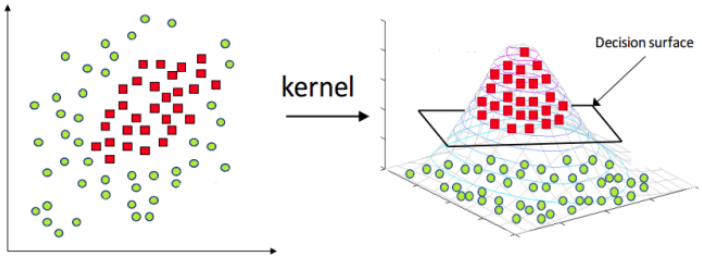


Figure 7: Kernel applied to data samples, which are not separable in a two dimensional way (source image: [15])

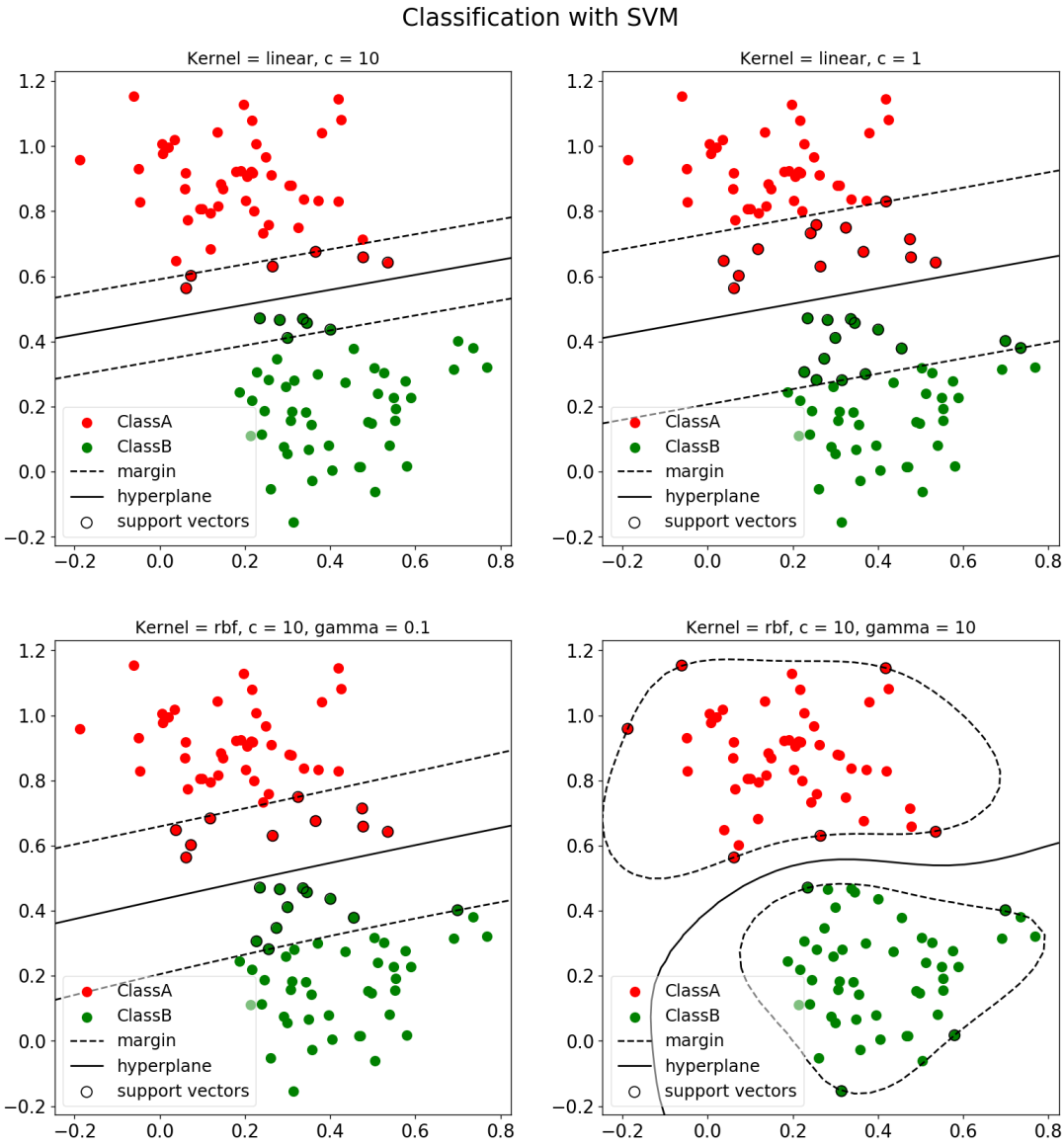


Figure 8: Classification with SVM on two randomised, linear separable clustered samples. Top row: hyperplane chosen by linear kernel with different value for C. Bottom row: samples divided by a hyperplane chosen by RBF kernel with different values for gamma.

When choosing the Gaussian radial basis function (RBF), a non-linear kernel, the parameter gamma needs to be specified before training the model. Gamma controls the influence of

samples on the decision boundary. The higher this value is chosen, the more influence the support vectors have on the hyperplane and therefore the more curved this line is. When gamma is very small, the model is too limited and cannot capture the complexity of the data. [16]

An example for a classification process of a SVM is illustrated in figure 8. Two different classes (red and green) are generated by random samples. These classes are then separated by the SVM with two different kernels and a varying number of c and gamma. The position of the separating hyperplane in the two linear models (top row) is in both examples the same and all points are on the 'correct' side of the hyperplane. But the margin is different and this is due to the value of c. As stated before, this value determines the size of the margin and the distance of the support vectors to the hyperplane. In the first example (c=10), the margin is smaller and therefore fewer points are included in the classification process. In the second picture, c is smaller and consequently the margin is larger and more points are used for the classification. In these two examples, the value c does not change the position of the hyperplane and as a result both models results are therefore accurate classifications.
The models in the bottom row use a non-linear kernel, in this case it is a RBF kernel. Therefore, the samples are not separated by a straight line, but by a curved one. The values gamma are chosen high to demonstrate the influence of this parameter. While in the first example (gamma=1) the hyperplane is almost linear, the line is curved in the second example (gamma =10).

2.4.5 Neural networks

Neural networks are DL models. The term "neural" is a biological term that refers to a nerve or the nervous system. Although the central concept for these models, the architecture, is a reference to the neural network in our brain, the functioning is different. A deep learning model is a mathematical and statistical model.

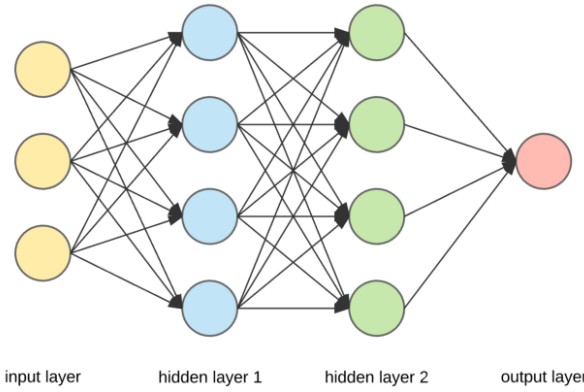


Figure 9: Architecture of a neural network with four layers: input layer, two hidden layers and the output layer.[3]

Figure 9 is an illustration of a neural network. The architecture is the basic structure of the whole model and it consists of different layers. Various mathematical and statistical operations are performed on the data in the layers. The first layer is called the input layer. This is where the initial data is fed into model. The data can be simple like a time series of temperature or the development of a single stock. In addition, three-dimensional vectors or images (four dimensions) or even videos (five dimensions) can be fed into neural networks. After the input layer, the data is then sent through the hidden layers. They are called hidden because the programmer usually cannot look at the data while it is being processed in these layers. Finally, the last layer is the output layer and the results of the neural network come together here.

Figure 10 illustrates the procession of data through a neural network and describes the learning procedure. The specification of the layer that processes the data is stored in the weights and the weights are numbers, usually between zero and one. The weights are applied on the input X through all the hidden layers, X is transformed into the prediction Y'. Y' is then compared to the true values (targets Y), which are present from the initial dataset. The network then measures the error (BIAS, root mean square error, etc.) of the prediction and calculates a loss score. The error is then passed through the network and by minimising the cost function through a gradient descent the weights are optimised. With every iteration the weights are adjusted a little into the correct direction and the loss score decreases. With a high number of iterations (tens, hundreds or even thousands) the loss score is getting

[3]Source image: Everything you need to know about Neural Networks and Backpropagation, towardsdatascience.com

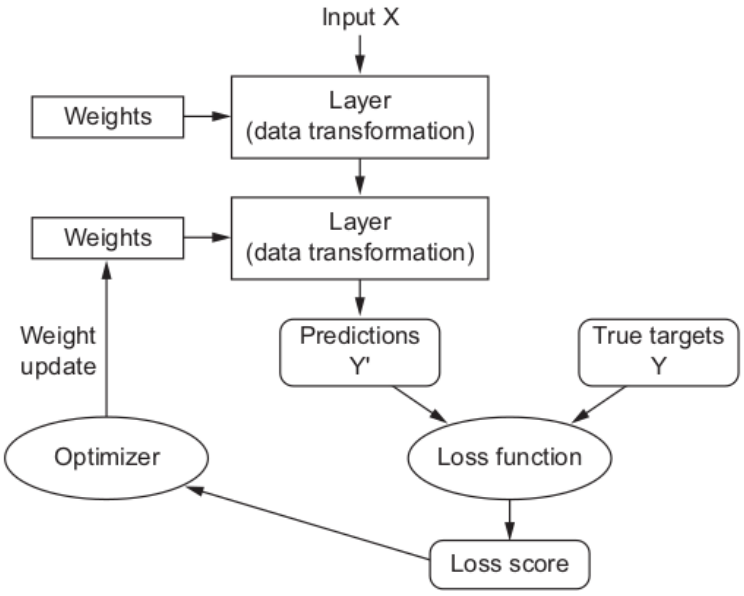


Figure 10: Functions of the different layers in a neural network [8].

smaller and smaller. In the end, the network has learned from the given data and it can be called a trained network.

In the hidden layers, not only are weights applied to the input data, but an activation function must also be passed. An activation function is a non-linear function that gives the neural network access to a much richer hypothesis space. Without it, it could only learn linear transformations. Typical activation functions are, for example, the Rectified Linear Activation (ReLU) or a logistic function (Sigmoid).

2.4.6 Convolutional neural network

Convolutional neural networks (CNN) are also neural networks but they are mostly applied for analysing visual imagery. The term convolutional is a reference to the mathematical operation convolution. Two functions create a third function that expresses how the form of one is modified by the others. Goodfellow et al. (2016) [17] states that: "Convolutional networks are simply neural networks that use convolution in place of general matrix multiplication in at least one of their layers".

The structure of a CNN is quite similar to that of standard neural networks. The first layer is the input layer, which is where the data is fed into the model. In the next layers the image is transformed into numeric values which depend on the structures and features of the image. In the last layer a neural network is processing the transformed image (feature image) and learns it patterns by the usual learning process.

As these this kind of neural networks are often applied to solve visual classification problems, the input is mostly images. When fed into the network, an image is a four dimensional vector. Since the input of a CNN consists not only of one but of several images, the computing power required to process this type of network is quite large. For technical reasons, explained later in this thesis, a CNN is usually computed by a graphic processing unit (GPU) and not by the central processing unit (CPU).

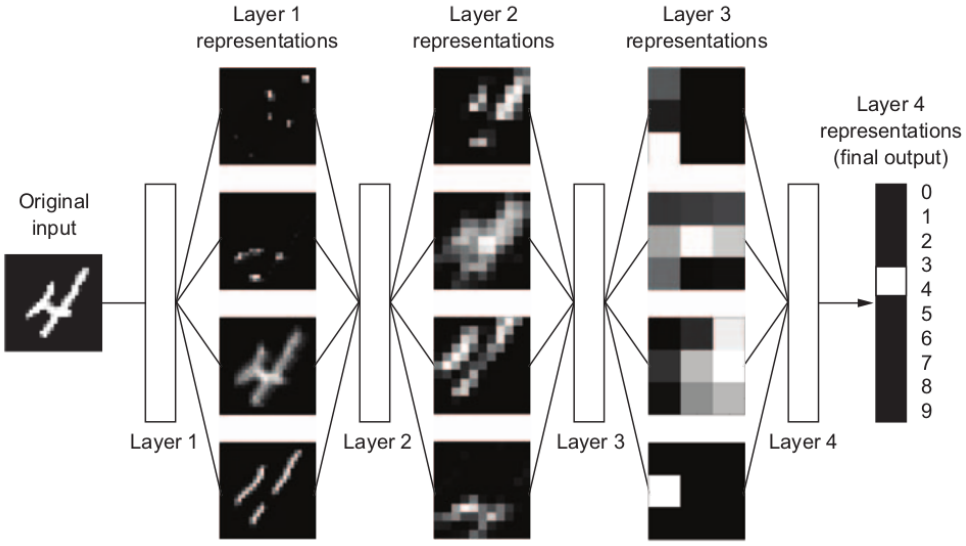


Figure 11: Architecture of a CNN with feature maps in the three layers and the prediction in the output layer (source image: [8]).

The CNN shown in figure 11 is a famous example of image recognition. The network tries to detect the right number from handwritten digits. The handwritten digits are part of the MNIST database. MNIST is short for "Modified National Institute of Standards and Technology database" and this database contains 60.000 training and 10.000 test images. This hand written digits were created by 500 different writers. [18] As shown in the example, the input is a single image with a handwritten number four. This image is then passed through different layers and in each layer local patterns of the images are learned. The deeper a network is, speaking of number of layers, the finer structures are detected. In this example, the CNN makes the correct prediction with number four.

Layers in a convectional neural networks

The architecture of a convolutional neural network looks basically like that of a neural network. But the layers function differently and have different names.

The description of the layers in the coming chapter will be kept simple, since the CNN is one part of this work and it is used with predefined packages from Python. The exact functionality as well as the exact description of the mathematical operations applied in the different layers would exceed the content of this work.

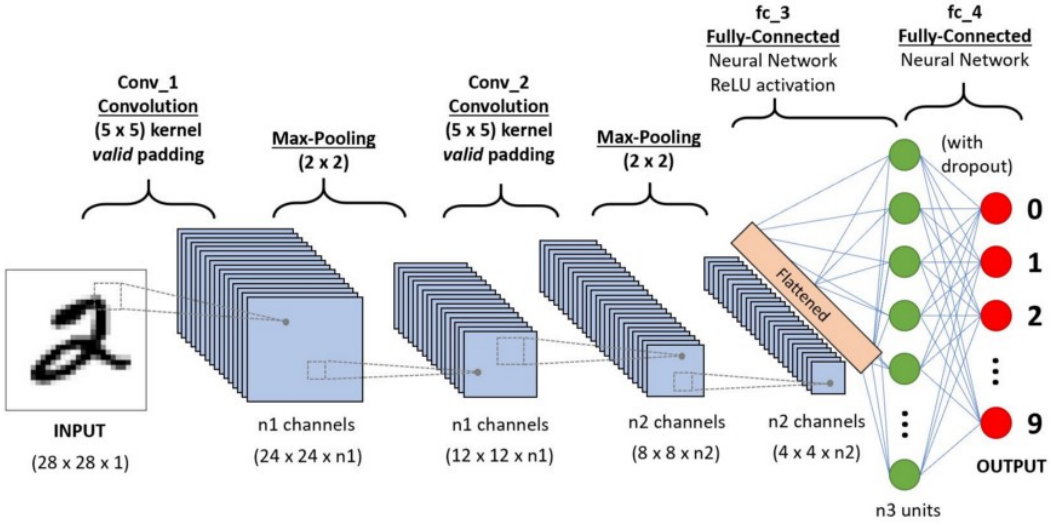


Figure 12: Different layers in a CNN. Two convolutional layers and two corresponding max-pooling layers. One fully connected layer with ReLu activation and one fully connected layer with softmax activation for the output.[4]

The different layers are shown in figure 12. In this example, the input is a single image from the MNIST database. The next layer is called the convolutional layer, this is where the feature from the images are extracted and the spatial size of the image is reduced. In the second layer, called pooling layer, the dominant features of the images are extracted. There are two kinds of pooling: average and max pooling. The first one extracts the average of all values from the portions from the image and the second one returns the maximum value. In figure 12 another convolutional layer as well as a max pooling layer follows. The number of these layers define the depth of a CNN. The layer before the output is the fully connected layer. This is where the matrix is flattened (3x3x3 into 9x1) and fed into a neural network. In this network, weights are applied on the previously analysed features and optimised by backpropagation. The output is the predicted class of the input images.

Convolutional layer

In this layer the features from original image are extracted. As an input, we have an image with 32x32x3 pixel for example. The height and width are 32 pixels and it has a channel for each RGB colour. A filter or also called a kernel is then applied on this image. The kernel is an array of numbers and these are basically weights. We assume the kernel has the size of 5x5x3. For mathematical reasons it is important that the filter has the same depth as the input image. The filter then moves along the image and multiplies each values (weights) with the values of the pixels from the original image. These to matrices are then convoluted into a single number.

[4]Source image: a comprehensive guide to convolutional neural networks, towardsdatasience.com
[5]Source image: a comprehensive guide to convolutional neural networks, towardsdatascience.com

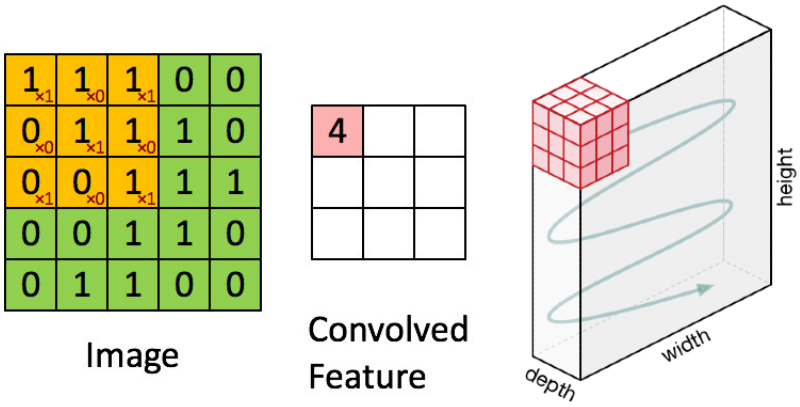


Figure 13: Left: Input image (green) with kernel (orange); Middle: feature map; Right: Movement of kernel along the input image[5]

An example of the kernel applied on an image and the movement of the kernel along the input image is illustrated in figure 13. Starting in the left corner of the given image the first number of the feature map will be constructed by this calculation:

$$\begin{pmatrix} 1 & 1 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 4 \end{pmatrix} \tag{1}$$

The size of the feature map depends on certain options, which are predefined for the CNN. These options determine how the kernel moves along the input image and how it handles the edges of the image. The size of the feature image is usually a bit smaller than the input image. This is due to the fact, that the kernel does not fit perfectly on the input image. If 3x3x3 kernel for example is applied on an input image with the size of 28x28x3, the filter does not fit exactly on the image. If the output size should be same as the input size, the options padding and stride are available.

- **Padding:** This option adds the appropriate number of rows and columns on each side of the input image, so that the kernel would perfectly fit at the image when moving along.
- **Stride:** Also the step size of the kernel can reduce or increase the size of the feature image. The default option is one, so the distance between to windows along the input image is one. The higher the number, the smaller the feature image gets.

Pooling layer

This layer transforms the feature image, which is created by the convolutional layer, into another image with reduced dimensions. In the pooling layer, a kernel is applied on the feature image, like in the layer before. By moving along the feature image and extracting certain

numbers, depending on the type of pooling layer, the dimension of the image is reduced but the important structures are saved.

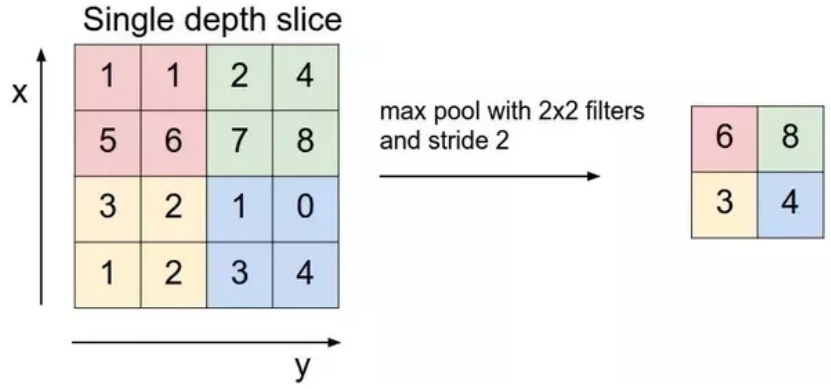


Figure 14: Max pooling applied on feature image [19].

In the illustration above (figure 14) the filter has the dimension 2x2 and stride 2. The filter moves along the feature image and since it is a max pool layer, it selects the maximum value out of the 2x2 window. Stride is again the step size of the kernel.

There are three types of pooling:

- Max Pooling: picks the maximum value of the windows
- Average Pooling: calculates the average of the values in the windows
- Sum Pooling: sums all values in the window up

Fully connected and output layer

The fully connected layer is the last layer before the output. This is where the results of the convolutional and pooling layer are used to make a prediction on the input. This layer is a neural network which applies weights to the input data (feature images) and makes a prediction depending on the values.

In the first step, the feature image, an array with dimensions 25x25x3 for example, is flattened into a single vector. This vector is then used as an input for the fully connected layer (neural network). The network is working like a normal neural network with a backpropagation process and the most accurate weights for the neurons are determined.

A softmax activation function then converts the vector of numbers in the output layer of the neural network into probabilities [values between 0 and 1].

Figure 15 illustrates the working principle of the fully connected layer. First, the feature image array is flattened into a single vector. Then the most accurate weights are applied, and

[6]Source image: fully Connected Layers in Convolutional Neural Networks, missinglink.ai

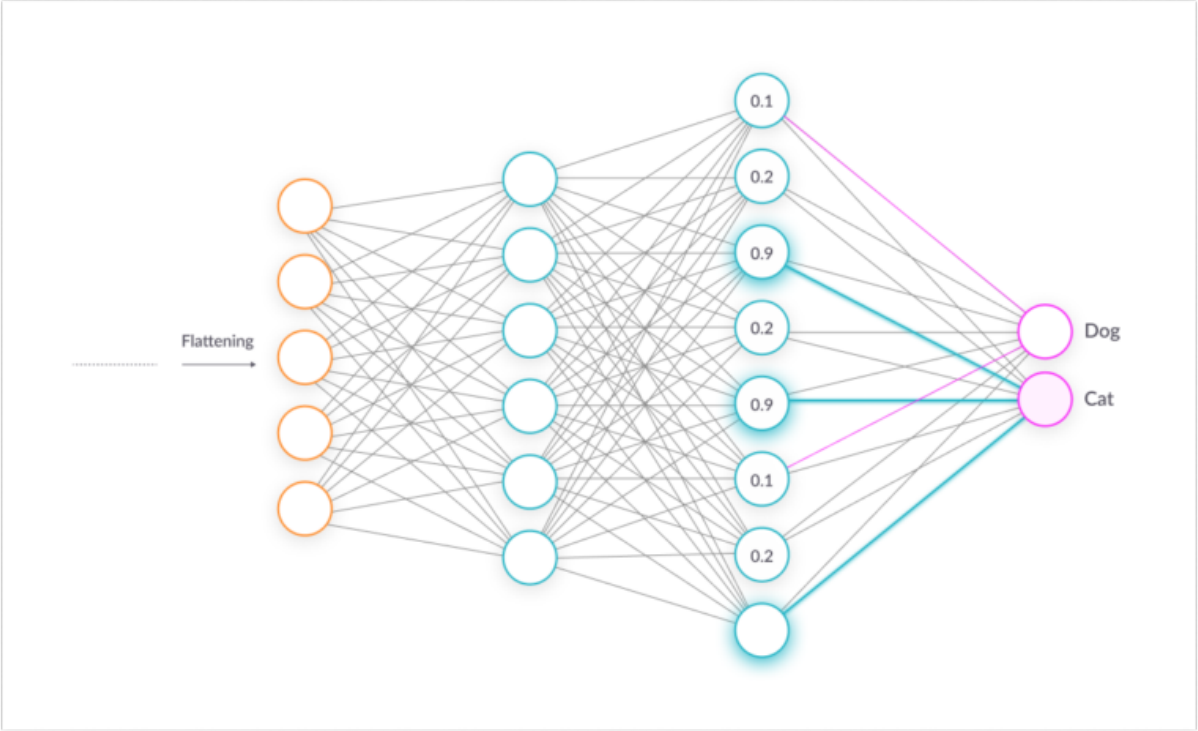


Figure 15: Fully connected layer with dogs and cats classification example[6]

the neurons are activated with an activation function. The result is then the probability for a certain class.

2.5 Basic terms for machine and deep learning models

In this section, several important terms, which are necessary to get a deeper understanding of how ML and DL models are trained and how to optimise the output of these models, will be explained. The models depend on large datasets, which need to be modified before the training. During the training, the programmer monitors the model to see if they are producing the expected outcome. For verification purposes it is also possible to train the model several runs on different splits of the dataset, this process is called k-fold cross validation. Finally, to find the best setting for the model, there are algorithms which automatically choose the best option. These topics are discussed on the next few pages.

2.5.1 Training, validation and test dataset

Supervised learning algorithms are designed to validate and improve themselves during the training process. In the case of a convolutional neural network, the original dataset can be split into three different parts. These parts are called the training, validation and test dataset.

- Training dataset: This dataset is the first input for a ML and DL model. It learns the pattern from the samples of the data and calculates the optimal weights.

- Validation dataset: Unseen data for the model, on which the model validates itself after each iteration.

- Test dataset: This dataset is used to test the completely trained model.

During the first iteration of the model, it only receives input from the data of the training dataset, and it optimises the weights based on this. After each iteration, the model makes new predictions based on the calculated weights from the training dataset on the validation data. With this step, the model is validated with new (unseen) data throughout the training. After the whole training process of the model, it makes predictions on the test dataset. This dataset shows the final result of the model and it is verified on it.

An important point is the size ratio of the three datasets. The ratio of the split depends on two following things: the total number of samples in the data and the model itself. If the model is shallow, few numbers of layers, the number of samples in the validation set can be reduced. If the total number of samples in the dataset is small, the test set can be smaller, so there is enough data for the validation and training dataset. To find a good split for the three dataset, different runs of the entire model are usually necessary.

2.5.2 Overfitting and underfitting of models

Overfitting and underfitting are two key terms to understand the learning process of ML and DL models. As humans are sometimes generalising people, also machine learning models can generalise data.

This of often the case when the model is learning "too much" from a single dataset. In complex models like neural networks, the methods detect pattern in the data. If there is too much noise in the data or the dataset is too small, the model is likely to generalise the data.

An example is shown in figure 16. Linear regression is about minimising the distances between the line and the points. Multiple iterations are applied on the data (points) until the line has an optimal fit to the points. In the picture on the left, the line fits the test data well and captures the trend of the points. A prediction of the trend (left end of the points) is therefore possible. In the image to the middle is a classic example of an overfit of the data. In this case, the overall cost becomes smaller with every iteration the algorithm makes. As a result, the linear regression does not capture the overall trend but it does capture the trend from point to point. In this case, no statement can be made about the future trend in the data. In the last example in figure 16 on the right, the model is underfitting the data. Meaning in this case, the algorithm is stopped early, and the model cannot learn from the data overall. Underfitting results in low generalisation and unreliable predictions. To find the perfect fit and the spot between overfitting and underfitting is the greatest challenge for programmers working with ML and DL models.

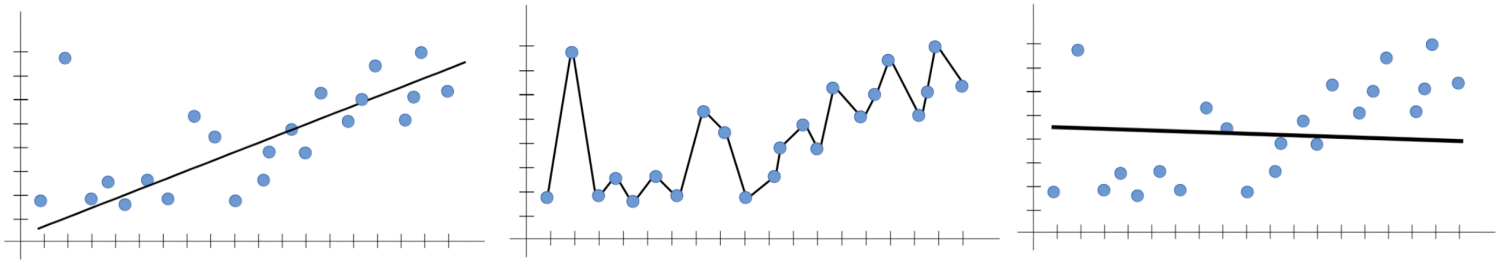


Figure 16: Linear regression fit for random distributed data samples. Left: good fit; middle: overfitting; right: underfitting[7]

When the algorithm learns, the error on the training set is usually decreasing as it learns the patterns of the data and optimises the weights. If the model is running to many iterations, the performance on the training set will still be good, but the model is overfitting the data and will learn irrelevant patterns from the noise. Stopping the model before the error on the test dataset starts to increase is the key for ML models.

Overfitting is a significant problem particularly with neural networks. Especially when the network is to deep and the amount of data to small, the network has a tendency to overfit the data. A typical sign of overfitting during the training process are improving learning curves

[7] Source image: What Are Overfitting and Underfitting in Machine Learning?, towardsdatascience.com

(accuracy and loss) for the training dataset and worsen learning curves for the validation dataset. In this case the model learns from the training dataset, but it cannot generalise on new images, which were not part of the training set.

There are several solution to this issues:

- Improve and extend dataset: Add more images to the dataset or improve the classification of the dataset by removing images with to much noise.
- Data augmentation: The number of samples in the dataset is increased. This is done by simply shearing, rotating or shifting a single image and add it to the dataset.
- Regularisation: Dropout, batch normalisation and other techniques
- Change architecture of the model

Likewise, underfitting is a limitation for neural networks. But the solution to this problem lies mostly in the deepening of the neural network. This is relatively easy with the help of additional layers.

2.5.3 Hyperparameter tuning

When programming a machine or deep learning model there are many different settings, which can be selected. From the type of model to the architecture of neural networks to the individual settings of the models, there are many different possibilities. Changing just one option, can have an enormous impact on the output of the model.

For the k-nearest neighbour or the k-means clustering the distance measure can be changed or the value for k. For support vector machines, the kernel, the size of the margin and the parameter gamma must be carefully chosen as they influence the performance significantly. For neural networks exist countless methods to modify the models. The number of layers or the number of neurons in the fully connected layer can be chosen individual for every model. Additionally the activation function, the learning rate and many other factors have an enormous impact on the output of the model.

Hyperparameters are the individual settings of the models, and they must be optimised (tuned) to get the best possible results. Finding the best hyperparameters is a challenge for every programmer. The models are trained repeatedly to find the best settings. However, certain methods provide solutions for this problem. One method, which is used in this thesis, is the so called Grid Search. A matrix with different hyperparameters serves as input for the function. The model is then calculated by trying out the settings for each run. It chooses the best run and thus the best hyperparameters automatically after a cross-validation and by calculating the loss and accuracy of the model results.

2.5.4 K-fold Cross Validation

An approach to train and verify the model on different training and test sets is the so called k-fold cross validation. It is often used to estimate the performance of machine learning algorithms on several datasets. [20] The dataset is divided into 'k' number of folds and partitions. The model is then trained on all partitions of the folds, except one. The last part is the training part with unseen data. This procedure is then repeated for each fold, generating five different model results. Because the model is trained and tested on different partitions each fold, the result of a model that uses a k-fold cross validation process is more reliable. In figure 17 is an illustration of how the dataset is split into four training partitions and one test partition. This routine is then repeated for each fold.

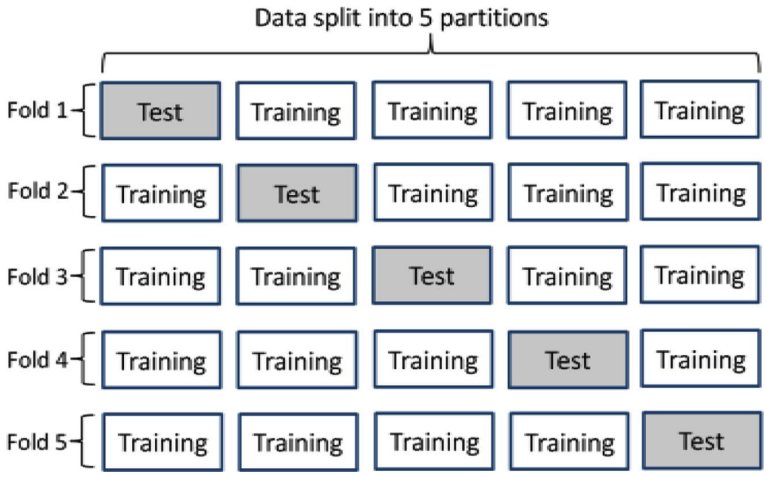


Figure 17: K-Fold Cross Validation with five different splits. Each fold consists of four training datasets and one test dataset[21].

2.5.5 Data vs. algorithms

The great advantage of machine and deep learning models is also a challenge for them. The models are dependent on large amounts of data. If the quantity of provided data is not enough, the algorithms will not provide satisfying results. In the paper [22] the authors state: *"Choose a representation that can use unsupervised learning on unlabelled data, which is so much more plentiful than labelled data."* Especially labelled data, representing the data needed to solve the problem, is hard to get for supervised learning techniques.

In the often cited paper *Scaling to Very Very Large Corpora for Natural Language Disambiguation* [23] the authors built a very large dataset containing of one billion words out of a variety of English texts. The dataset is tested on several machine learning methods and learning rates are documented. The result can be seen in figure 18. In the abscissa the number of words is in logarithmic representation, so one can see that the curves appear in log-linear. The authors suggest, to reconsider the trade-off between spending time and money on the

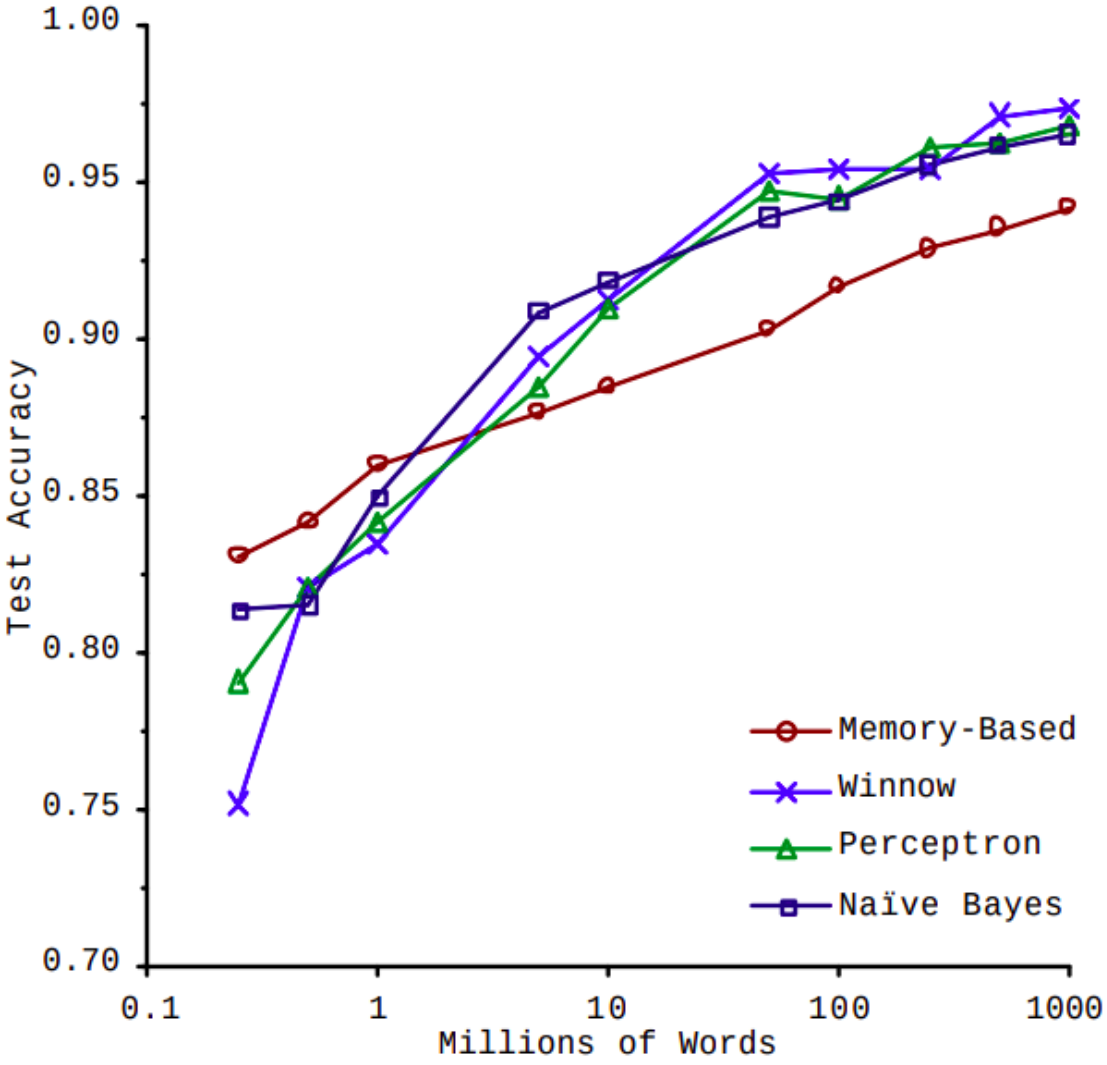


Figure 18: Accuracy of different models depending on the amount of data they were trained on (source image [23]).

development of the algorithms versus spending it on the development of the data. This context again shows, the importance of well-functioning algorithms, but also the importance of large and especially in supervised techniques well labelled data. Today, most data scientists spend a significant time of their job cleaning up the training data or labelling data for the ML and DL models.

3. Methods and algorithms used in this work

In the ongoing section the threshold methods and the settings of the machine and deep learning models are described in more detail. The methods and models are used in certain algorithms, which are adjusted for the usage of the datasets.

3.1 Threshold methods

Threshold methods are used to compare the intensity of the pixels in an image against a fixed or variable threshold. As colour intensities are the primary characteristics to distinguish cloud from sky pixels, it is obvious to use this characteristic to classify the pixel. [24] The first method, which is applied in this thesis, is the so called Normalised Red/Blue Ratio (NRBR). Secondly the Euclidean Geometric Distance (EGD) is used, to determine whether a pixel belongs to the clouds or the sky.

3.1.1 Normalised Red/Blue Ratio (NRBR)

The NRBR is the ratio between red and blue colour channels through their sum (normalised). The predominant factor of scattering light in the atmosphere is due to the Rayleigh Scattering. Therefore visible light is scattered by particles (molecules), which are much smaller than the wavelength of the electromagnetic radiation. The sky appears blue during the day because the strong wavelength dependence ($\sim \lambda^{-4}$) of the scattering causes a stronger scattering of the shorter wavelength. [4]

$$NRBR = \frac{B-R}{B+R} \qquad B\text{...blue pixel} \qquad R\text{...red pixel} \tag{2}$$

Sky pixels with a dominant blue colour have high NRBR values, while cloud pixels, which disperse blue and red light in nearly equal amounts, have low NRBR values, as shown by equation 2. The NRBR ranges in the interval [-1,1]. [2]

The threshold for the NRBR is taken out of the work by Li et al. (2011) [2]. However, the algorithm is simplified somewhat in the context of this thesis and a fixed threshold value is used for this method. The threshold depends on the mean and standard deviation of the image and is selected by $\mu + 3\sigma$. Where μ is the mean of the NRBR values of the image and σ the standard deviation. Pixels below this value are determined as cloud pixels, above as sky pixels.

3.1.2 Euclidean Geometric Distance (EGD)

The Euclidean Geometric Distance compares the RGB value of the pixel with the diagonal in the RGB cube. The diagonal in the RGB cube represents the transition from white to black, with grey tones in the middle. Clouds mostly appear in colours near to the diagonal. Sky pixels appear mostly in blueish colours and therefore they tend to be near the blue corner of the RGB cube.

$$EGD = \sqrt{R^2 + G^2 + B^2 + \frac{(R+G+B)^2}{3}} \qquad G\text{...green pixel} \tag{3}$$

Equation 3 [4] shows how the EGD is calculated. In contrast to the NRBR, the colour green is considered in the EGD method. With some basic knowledge of linear algebra, one can see that the equation includes the norm of the RGB vector and therefore the EGD of a pixel is the distance to the diagonal in the RGB cube. After the distance is calculated, it is again compared to a threshold, which separates between cloud and sky pixels.

Regarding the selected threshold for the EGD method, the definition by Mantelli Neto et al. (2010) [25] is used. Statistical analysis has shown that the cloud and sky pixels can be distinguished by a certain threshold depending on the distribution of the colours in the image. In their work and also in this thesis the threshold is selected by $\mu - 3\sigma$. The mean of the entire image analysed with EGD is μ, and the standard deviation is σ. If the selected pixel is under the threshold it will be defined as cloud pixel else as sky pixel.

3.2 Machine and deep learning algorithms

The methods used for this work and how they work have been explained in previous chapters. This section will now explain how they are applied and which algorithms and functions (in Python) are used. First the k-nearest neighbour method is explained. The next method is an algorithm which combines the k-means clustering and the k-nearest neighbour models. The support vector machine is used for the determination of the cloud cover of the images and for the classification of the cloud images. The last model applied on the datasets is the convolutional neural network. Three different CNN models are tested on the datasets.

3.2.1 K-nearest neighbour

The k-nearest neighbour model is used for the determination of cloud cover. The algorithm implemented in python is part of a Github repository by Ahmet Özlü, 2018 [26]. The steps how the algorithm works are illustrated in figure 19.

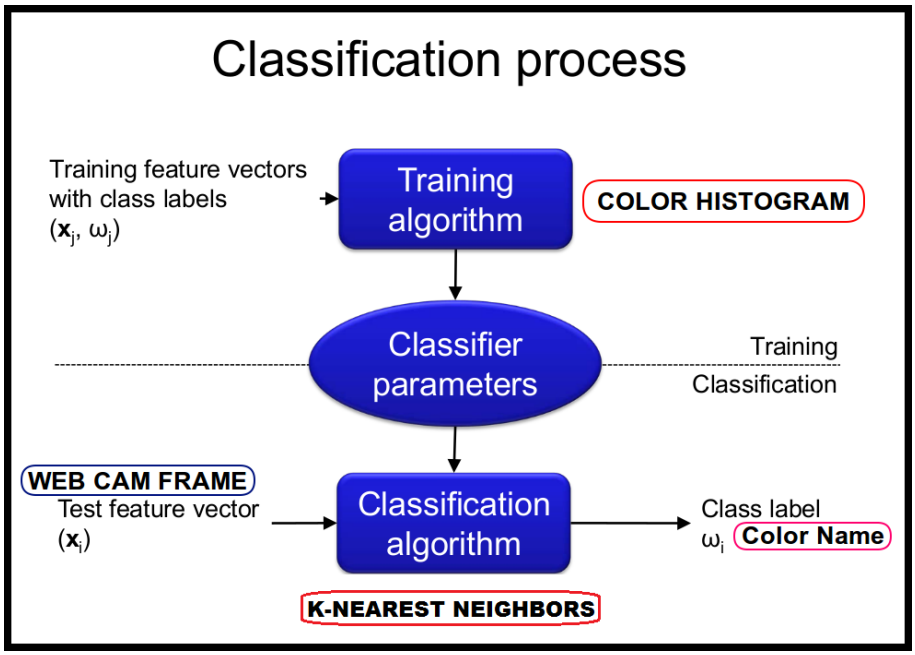


Figure 19: Classification algorithm with k-nearest neighbour. First the colour histogram is created by extracting the colour vectors of training data. Then the pixels of the image is compared to the histogram and picks are made with k-nearest neighbour algorithm.[1]

The first step is to extract the training vectors with the class labels from the training dataset. (Figure 19 "Training algorithm"). The training dataset consists of images with single colours. There are approximately five to ten images per colour and the total number of colours is eight. This results in the so called colours histogram, and a part of the file is shown in figure 20. The list contains the R, G, and B values of each colour as well as the name of the colour. The k-nearest neighbour classification is then trained on this histogram. With the RGB values as the vector and the colour names as the target. (Figure 19 "Classifier parameters")

In the last step, the RGB value from each pixel of the cloud image is compared to the RGB values from the colour histogram (list figure 20). The k-nearest neighbour method compares the values from the image and the list and selects the closest RGB value based on the value for k (number of neighbours) and its euclidean distance. (The algorithm was originally designed to extract colours of a webcam frame. Therefore, the input in figure 19 is not a webcam frame, but a cloud pixel.)

[1] Source image: https://github.com/ahmetozlu

Settings for the k-nearest neighbour model:

- Name of the function: KneighboursClassifier[2]
- Number of neighbours: 3
- Metric for the distance: euclidean
- Weights for picking neighbours: number and distance

```
37,37,37,black
40,39,45,black
36,29,33,black
0,0,254,blue
3,91,188,blue
247,224,23,yellow
255,183,9,yellow
253,251,251,white
248,249,254,white
```

Figure 20: Part of the colour histogram. The columns contain the R,G,B, values and the colour names.

3.2.2 K-means clustering

In this thesis, the k-means clustering method is used to put the cloud image in k number of segments. This process simplifies the image and therefore makes the analysis more efficient. [27] Figure 21 shows the analysis with the k-means clustering algorithm of an image with cumulus clouds. The smaller the number of clusters, the fewer colours are considered in the analysis. A small number of clusters is often enough to fully describe an image with clouds, whereas an analysis with 15 or 20 clusters makes no real difference to the original image.

The centres of the clusters are chosen by the k-means++ algorithm. This algorithm first takes one centre, chosen uniformly at random from the whole dataset. In the next step the distance between the surrounding data points and the centre is calculated. Next, a new centre is chosen by its probability which is directly proportional to its distance from the previous centre. For example the point with maximum distance to the previous centre. This process is then repeated until k centres are chosen. [28]

[2] sklearn.neighbours.KneighboursClassifier

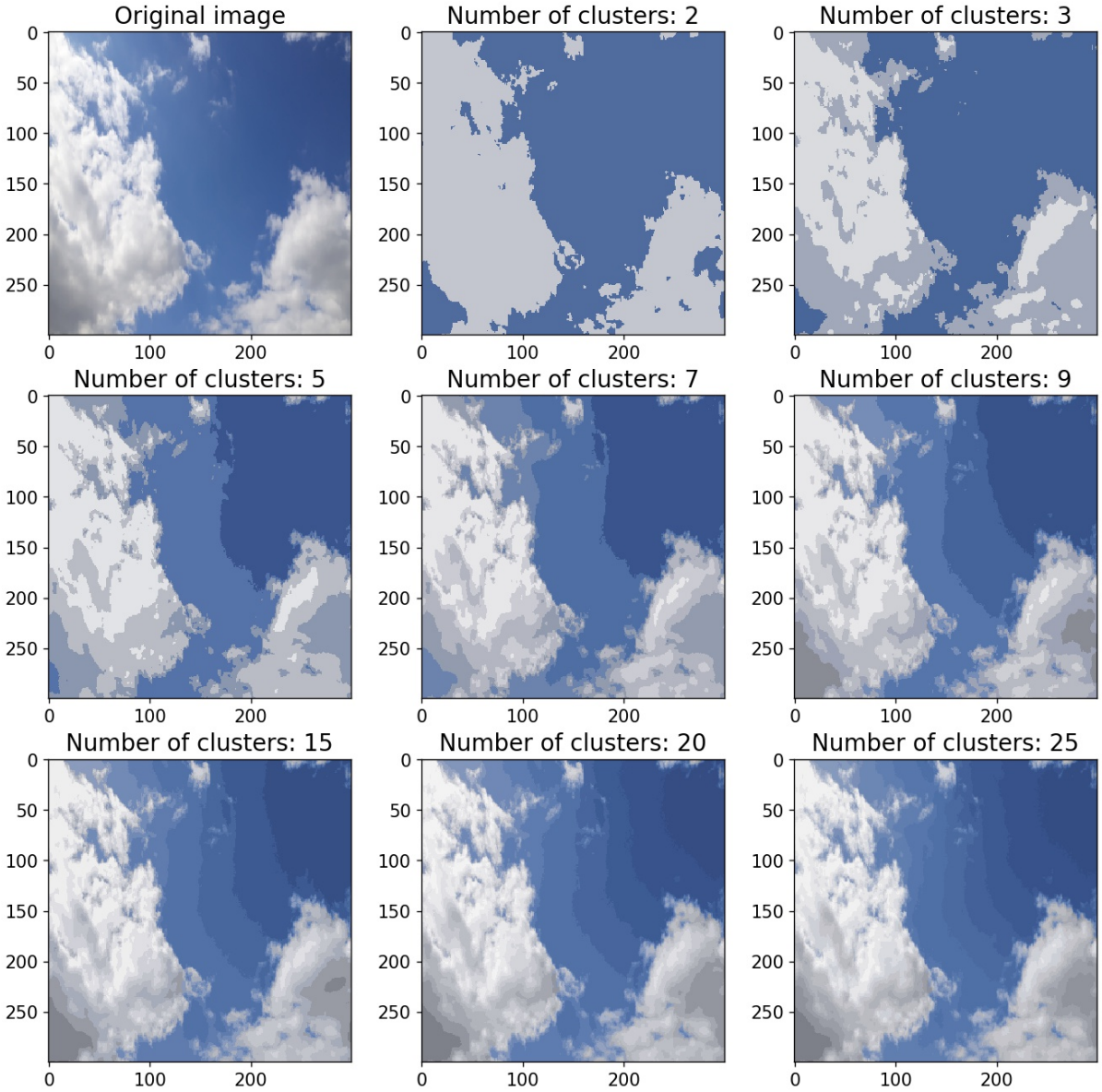


Figure 21: K-means clustering of a cloud image. Starting from two clusters, up to 25.

Settings for the k-means clustering model:

- Name of the function: KMeans[3]
- Method for initialisation: k-means++
- Number of clusters: 3
- Iterations: 300

[3] sklearn.cluster.KMeans

3.2.3 Hybrid

K-means clustering and nearest neighbour are combined to gain the benefits of both machine learning methods. The name of this method is simply the Hybrid method because it is a combination of the two algorithms. In Lothon et al. (2019) [29] the background study for this paper shows several other works, which combine different algorithms for the detection of cloud cover from sky cameras.

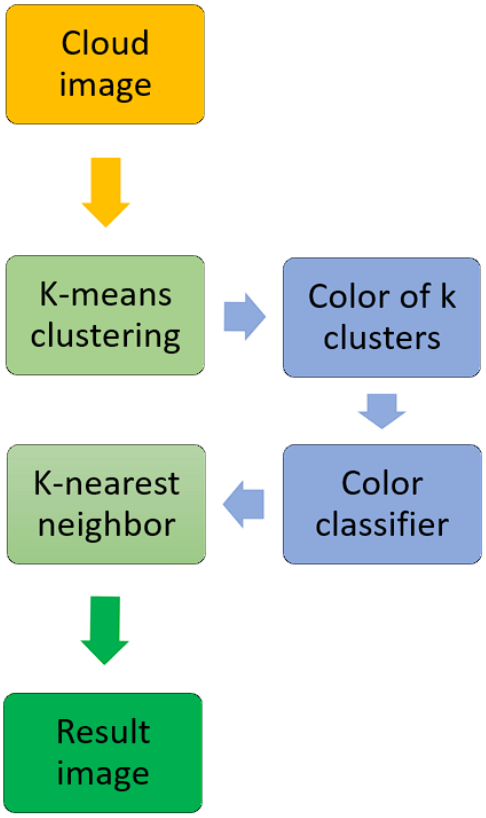


Figure 22: Process of the hybrid method

Figure 22 illustrates the different steps of the hybrid method. The input is a cloud image. In the next step the image is clustered by k-means clustering. This process simplifies the processing of the image and thus accelerates the analysis. The next two steps are marked in blue in figure 22. The "colour of clusters" extracts the RGB values from the analysed clusters. If the number of clusters (k) is four, the picks for the colours could be "white","white","blue and "brown" for example. In this case, the two white clusters represent clouds, while blue and brown represent the sky or any other objects in the image. The RGB vectors from the training set are then loaded into the "colour classifier." (Refer to section 3.2.1 for more information on this process). There are k numbers of RGB vectors from the clusters and a variety of RGB vectors from the training dataset after these two steps. The k-nearest neighbour method compares these two vectors and selects the most common neighbours by distance

and number of neighbours. Cloud or no cloud is determined by the RGB values of the clusters. The result is a cloud and sky clustered image.

3.2.4 Support vector machine

The support vector machine is used for the determination of the cloud cover and cloud types of the images.

The SVM is trained on five images from the SWIMSEG (for an in-depth description of this dataset see chapter 4.1.2) dataset to determine the cloud cover. Images with white clouds and blue clear sky, as well as examples with darker clouds, are included in the training dataset. The size of the images is 100x100x3 pixels. In total the SVM is trained on 120.000 data points and tested an 30.000. The model learns the RGB values of the images. As these images only contain parts of clouds and sky, the RGB values cover blue, white and grey colours. The function 'GridSearchCV'[4]is used to find the best hyperparameters for the model. After the model is trained it is analysing the classes (cloud/no cloud) for each pixel of the image.

For the classification of cloud types, the algorithm is tested with several kernels, different margins and different values for gamma. Linear and non-linear kernels are applied. The margin of the SVM is adjusted with the variable C.

Settings for support vector machine model:

- Name of the function: C-Support Vector Classification[5]
- Kernels: linear and RBF
- C: 1, 10, 100 and 1000
- Gamma: 0.001 and 0.01

3.2.5 Convolutional neural network

Three different model architectures are tested on the data samples. For the first model the architecture by Chollet, 2018 [8] is used. This model is initially used to classify the MNIST dataset and is then further developed to distinguish between dogs and cats. As the classification of dogs and cats is a binary classification problem, the architecture is modified to fit for the SWIMCAT dataset. The other two models are by the work of Rhee et al, 2018 [21] and Zhang et al. (2018) [7]). These two models were originally designed to determine cloud types using the SWIMCAT and CCSN datasets. An in-depth description of these two datasets can be found in chapter 4.1.1 and 4.1.3.

[4]sklearn.modelselection.GridSearchCV
[5]sklearn.svm.SVC

Model by Chollet, 2018 [8]

The first model can be seen in figure 23. There are four convolutional (with ReLU activation function) and four max pooling layers in this model. The model is then flattened, and two fully connected layers are added. The first fully connected layer, in Python also called dense layer, contains 512 units (nodes, neurons) with a ReLU activation function. The second one only contains five units (five classes) with softmax activation function, which transforms the values into probabilities.

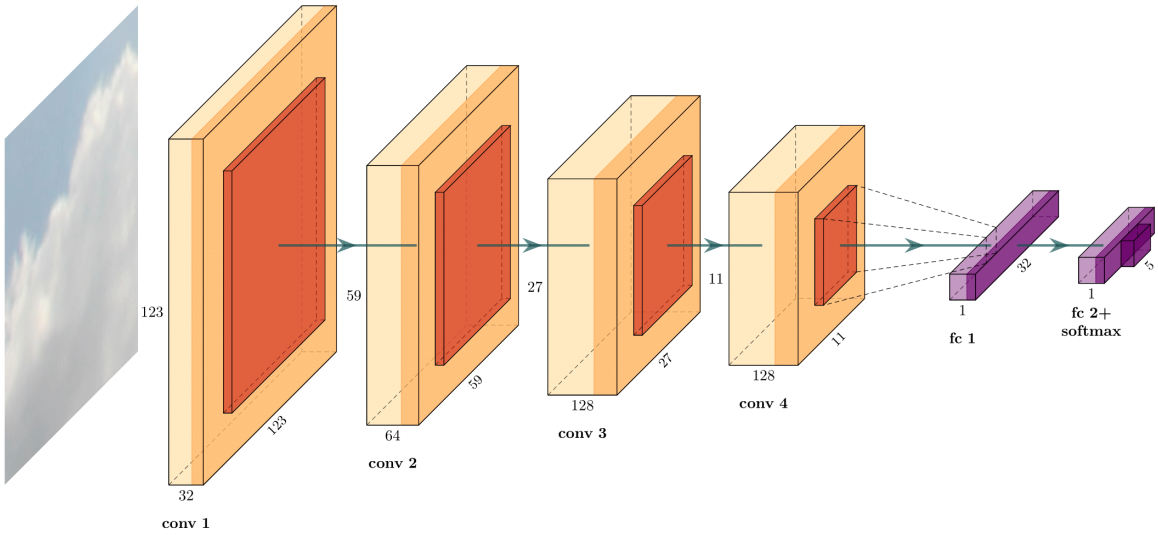


Figure 23: CNN architecture of the model by Chollet, 2018 [8]. Four convolutional layers (conv1 - conv4) in orange, each with a max pooling layer in dark orange. Two fully connected layers (fc1, fc2+softmax) in purple are at the end of the model.

The dimension of the input images is transformed passing each layer. The original shape of the images is 125x125x3 pixels. The depth of the feature maps (number of filters) increases in the network (from 32 to 128), whereas the size of the feature maps decreases (from 123 to 5). This is a pattern for most models, because the feature images should not be too large when reaching the flatten layer. [8]

The summary of the model in table 1 shows how the dimension of the original images changes while passing all the different layers of the convolutional neural network. The name of the layer is in the first column of the table. These layers have special names in the Python. The second column with the name "Output Shape" contains information of the feature image. The last column shows the quantity of the parameters. These are based on the number of weights and biases of the different layers. The total number of parameters is an indicator for the complexity of the model.

```
Model: "Model from Deep Learning with Python"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_4 (Conv2D)            (None, 123, 123, 32)      896
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 61, 61, 32)        0
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 59, 59, 64)        18496
_________________________________________________________________
max_pooling2d_5 (MaxPooling2 (None, 29, 29, 64)        0
_________________________________________________________________
conv2d_6 (Conv2D)            (None, 27, 27, 128)       73856
_________________________________________________________________
max_pooling2d_6 (MaxPooling2 (None, 13, 13, 128)       0
_________________________________________________________________
conv2d_7 (Conv2D)            (None, 11, 11, 128)       147584
_________________________________________________________________
max_pooling2d_7 (MaxPooling2 (None, 5, 5, 128)         0
_________________________________________________________________
flatten_1 (Flatten)          (None, 3200)              0
_________________________________________________________________
dense_2 (Dense)              (None, 512)               1638912
_________________________________________________________________
dense_3 (Dense)              (None, 5)                 2565
=================================================================
Total params: 1,882,309
Trainable params: 1,882,309
Non-trainable params: 0
_________________________________________________________________
```

Table 1: Summary of the model by Chollet, 2018 [8].

Explanation of the output shape from table 1: (None,123,123,32)

- None: Defines the number of images determined at model training.
- (123,123): Size of the image (x,y) within a two dimensional colour space.
- 62: Number of filters or number of feature images within a layer.

Model by Rhee et al. (2018) [21]

The model, seen in figure 24, is taken out from the work by Rhee et al. (2018) [21]. The model is built up on three convolutional layers with ReLU activation function and three corresponding max pooling layers. In between, there are two dropout layers. These layers remove 25% and 50% of the neurons from the model, respectively, and reduce overfitting. The fully connected layer is a neural network with 32 neurons. The last layer contains five neurons and a softmax activation function. The image is transformed from 123x123x32 pixels down to 13x13x16. The total number of parameters in table 2 shows, that the model is shallower than the one described in the previous subparagraph.

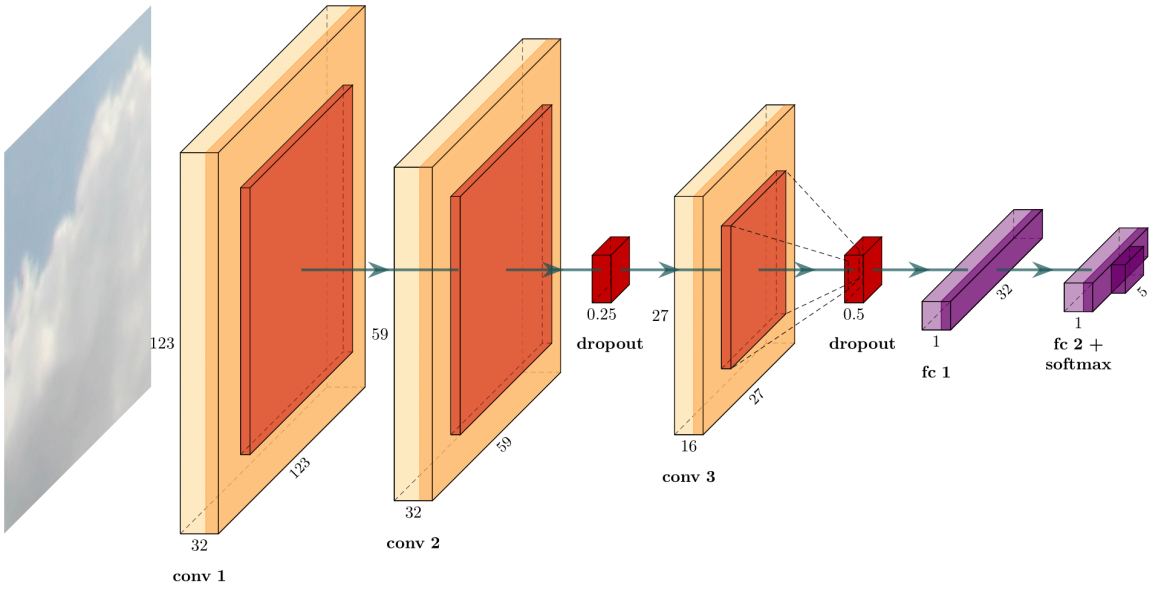


Figure 24: CNN architecture of the model by Rhee et al. (2018) [21]; Two convolutional layers with max pooling layers are followed by a dropout layer (red). Next comes another convolutional layer and another dropout layer with 0.5% dropout. At the end two fully connected layers follow.

```
Model: "Model from Rhee2018"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_8 (Conv2D)            (None, 123, 123, 32)      896
_________________________________________________________________
max_pooling2d_7 (MaxPooling2 (None, 61, 61, 32)        0
_________________________________________________________________
conv2d_9 (Conv2D)            (None, 59, 59, 32)        9248
_________________________________________________________________
max_pooling2d_8 (MaxPooling2 (None, 29, 29, 32)        0
_________________________________________________________________
dropout_1 (Dropout)          (None, 29, 29, 32)        0
_________________________________________________________________
conv2d_10 (Conv2D)           (None, 27, 27, 16)        4624
_________________________________________________________________
max_pooling2d_9 (MaxPooling2 (None, 13, 13, 16)        0
_________________________________________________________________
flatten_2 (Flatten)          (None, 2704)              0
_________________________________________________________________
dropout_2 (Dropout)          (None, 2704)              0
_________________________________________________________________
dense_4 (Dense)              (None, 32)                86560
_________________________________________________________________
dense_5 (Dense)              (None, 5)                 165
=================================================================
Total params: 101,493
Trainable params: 101,493
Non-trainable params: 0
_________________________________________________________________
```

Table 2: Summary of the model by Rhee et al. (2018) [21].

Model by Zhang et al. (2018) [7]

Finally, a much more complex model is used for comparison. This model is the so called CloudNet by Zhang et al, 2018 [7]. The architecture and summary of the model is illustrated in figure 25 and table 3. The size of the images (input shape) is the most significant difference between the two previous models. Previously, the input shape was (125,125,3), but now it is (250,250,3). This is due to the fact that the model was created for images containing more pixels, but the input shape is adapted to the images for the used datasets. The number of units in the fully connected layer in this model is greater compared to the models before. As a result, the total number of parameters is significantly greater, making this model more complex than the previous ones.

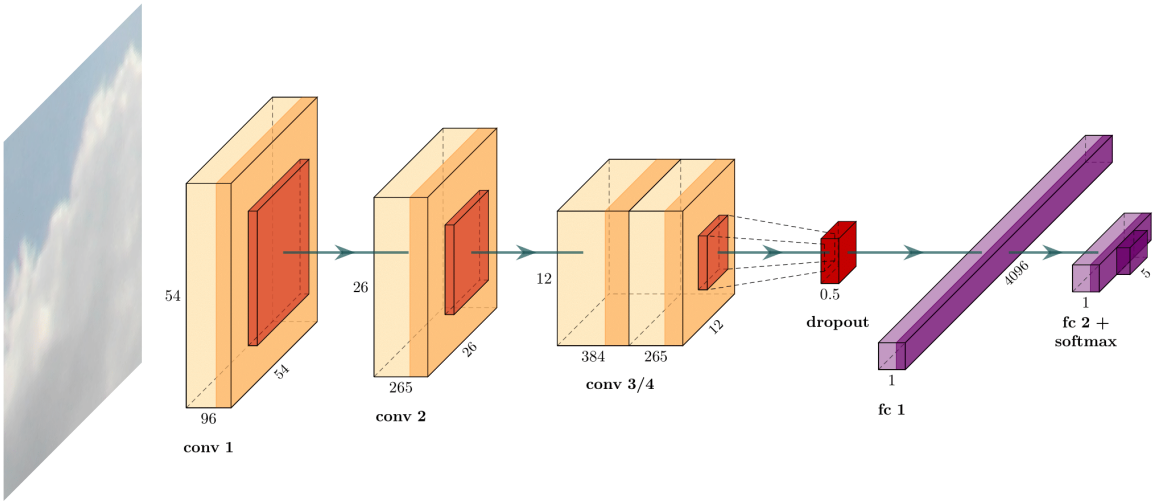


Figure 25: CNN architecture of the model by Zhang et al. (2018) [7]; The model contains of two convolutional layers (ReLU activation) with max pooling layers and two consecutive convolutional layers with one max pooling layer. These are followed by a dropout layer where 0.5% of the neurons are dropped out. Two fully connected layers are at the end, the first with ReLU activation and the last with softmax activation.

```
Model: "CloudNet"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_12 (Conv2D)           (None, 54, 54, 96)        34944
_________________________________________________________________
max_pooling2d_12 (MaxPooling (None, 26, 26, 96)        0
_________________________________________________________________
conv2d_13 (Conv2D)           (None, 26, 26, 256)       614656
_________________________________________________________________
max_pooling2d_13 (MaxPooling (None, 12, 12, 256)       0
_________________________________________________________________
conv2d_14 (Conv2D)           (None, 12, 12, 384)       885120
_________________________________________________________________
conv2d_15 (Conv2D)           (None, 12, 12, 256)       884992
_________________________________________________________________
max_pooling2d_14 (MaxPooling (None, 5, 5, 256)         0
_________________________________________________________________
flatten_3 (Flatten)          (None, 6400)              0
_________________________________________________________________
dropout (Dropout)            (None, 6400)              0
_________________________________________________________________
dense_6 (Dense)              (None, 4096)              26218496
_________________________________________________________________
dense_7 (Dense)              (None, 5)                 20485
=================================================================
Total params: 28,658,693
Trainable params: 28,658,693
Non-trainable params: 0
_________________________________________________________________
```

Table 3: Summary of the model by Zhang et al. (2018) [7].

K-fold cross validation is already described in chapter 2.5.4 and it is used to compare the results of each model on different test and validation sets. The dataset is split into five parts and these parts are then used to train and validate the model. In the end, the algorithm produces five different results per model. For the number of epochs 1000 is chosen. An epoch is complete if an entire dataset is passed forward and backward through the neural network once. Due to limited computing resources, running the entire dataset through the model is often not possible. As a result, the dataset is divided into batches of images. These batches are then sent through the network until all of them have passed the model, at which point one epoch is complete. The number of images in one batch depends on the size of the images as well as on the processing power of the computer.

During the calculation process the CNN already generates an output, this can be seen in table 4. The first few lines describe the training set. The training set contains 627 images with 125x125x3 pixels. The next line shows the batch size and the last one the validation split. As the validation split is 0.2, the model is trained on 501 images and verifies itself on 126 (= 627*0.2) images. During training, the model sends batches of 60 images through all layers nine times (60*8+21*1=501). One epoch is completed after the model has been verified on 126 images from the validation dataset.

```
------------------ Starting model ------------------
Shape training set: (627, 125, 125, 3)
Batch size: 60
Validation split: 0.2

----------------------------------------------------------------------
Training for fold 1 ...
Train on 501 samples, validate on 126 samples
Epoch 1/100
501/501 [==============================] - 10s 19ms/sample - loss: 1.5497 - acc: 0.4212 - val_loss: 1.4332 - val_ac
c: 0.4444
Epoch 2/100
501/501 [==============================] - 9s 18ms/sample - loss: 1.3005 - acc: 0.5289 - val_loss: 1.1039 - val_ac
c: 0.4444
Epoch 3/100
501/501 [==============================] - 9s 18ms/sample - loss: 1.0224 - acc: 0.5130 - val_loss: 0.9343 - val_ac
c: 0.7302
Epoch 4/100
501/501 [==============================] - 9s 18ms/sample - loss: 0.9465 - acc: 0.6188 - val_loss: 0.8689 - val_ac
c: 0.7143
Epoch 5/100
224/501 [============>.................] - ETA: 4s - loss: 0.9246 - acc: 0.6473
```

Table 4: Output of the CNN during the calculation process.

The cost and loss function is an important consideration. These two functions indicate how well the model learns from the data. Because the values of the two functions are always displayed at the same time for each epoch, the programmer can determine whether it is more efficient to run the entire model to completion or to stop it early. By comparing the "loss" (loss function training set) and "val loss" (loss validation set) or the accuracy of the two sets, one can also tell if the model is over- or underfitting the data.

4. Data and programming

4.1 Data

The best foundation for any neural network is the data it is trained on. Today, there are thousand of labelled or classified datasets available all over the internet. To find an accurate dataset to address the problem of classifying cloud images, expertly labelled data must be available. As Zhang et al. (2018) [7] quotes: "the availability of sufficient training samples is the fundamental issue that should be addressed." Having good labelled data is one of the major problems for neural networks.

For this work, three different types of datasets are used. The samples from the SWIMSEG dataset are used to train the SVM and to verify the methods for determining cloud cover. The other two datasets contain labelled data and are used for the SVM and CNN for the classification of the clouds. The first and simpler one is the so called SWIMCAT dataset and the second one is called CCSN.

4.1.1 SWIMCAT

SWIMCAT is an acronym for Singapore Whole sky IMaging CATegories Database. The database was originally designed for the work by Dev et al. (2015) [6]. The images for this dataset come from a calibrated ground-based whole sky imager, the so called WAHRSIS (Wide Angle High Resolution Sky Imaging System). They were captured over the period January 2013 to May 2014. Based on the visual features of sky and cloud conditions 784 patches of cloud images were selected and categorised in cloud categories with experts from the Singapore Meteorological Services. The dataset contains five categories, which are labelled as follows: A-sky, B-pattern, C-thick-dark, D-thick-white and E-veil. An excerpt from each category of the dataset can be seen in figure 26. All images are in the dimension of 125 × 125 pixels.

Figure 26: Categories: 1. A-sky, 2. B-pattern, 3. C-thick-dark, 4. D-thick-white, 5. E-veil [6].

4.1.2 SWIMSEG

The Singapore Whole sky IMaging SEGmentation Database (SWIMWEG) was originally created by Dev et al. (2016) for their work *Colour-based Segmentation of Sky/Cloud Images from Ground-based Cameras* [5]. The images come from WAHRSIS and the dataset contains of 1013 images. The images were handpicked by the authors 2016 in the period from October 2013 until July 2015 in the city of Singapore. The image size is 600x600 pixels. The main advantage of this dataset is the corresponding ground truth masks to each image. These masks were created in consultation with meteorologist from the Singapore Meteorological Services. An example of the images from the dataset can be seen in figure 27. This dataset is used to verify the threshold and machine learning methods. The methods try to analyse the cloud cover of the images of the dataset. The ground truth mask represents the true cloud cover of the images and therefore is used for the verification of the methods.

Figure 27: Images from SWIMSEG dataset with corresponding ground truth [5].

4.1.3 CCSN

CCSN is short for Cirrus Stratus Nimbus dataset. This database was constructed by Zhang et al. (2018) [7] and it contains 2543 images. The size of all images is 256x256 pixels. The images were divided by meteorological experts based on the visual characteristics of the clouds. In total there are eleven categories of different cloud appearances, also contrails were added as a category. A subset of all the categories can be seen in figure 28 and a detailed description of each category is given in table 5.

Figure 28: Categories: left to right, top to bottom: Ac, As, Cb, Cc, Ci, Cs, Ct, Cu, Ns, Sc, St [7].

Category	Number of images	Type	Description
Ci	139	Cirrus	Fibrous, white feathery clouds of ice crystals
Cs	287	Cirrostratus	Milky, translucent cloud veil of ice crystals
Cc	268	Cirrocumulus	Fleecy cloud, cloud banks of small, white flakes
Ac	221	Altocumulus	Grey cloud bundles, compound like rough fleecy cloud
As	188	Altostratus	Dense, gray layer cloud, often even and opaque
Cu	182	Cumulus	Heap clouds with flat bases in the middle or lower level
Cb	242	Cumulonimbus	Middle or lower cloud level thundercloud
Ns	274	Nimbostratus	Rain cloud; grey, dark layer cloud, indistinct outlines
Sc	340	Stratocumulus	Rollers or banks of compound dark gray layer cloud
St	202	Stratus	Low layer cloud, causes fog or fine precipitation
Ct	200	Contrails	Line-shaped clouds produced by aircraft engine exhausts
Total	2,543		

Table 5: Description of the categories [7].

4.2 Programming

The Python programming language is used to programme the machine and deep learning algorithms. This programming language is open source and it comes along with a variety of modules, especially for machine and deep learning. In Python, there are three major frameworks for machine and deep learning. TensorFlow (tensorflow.org) is a software library for machine learning and is required for the deep learning library called Keras (keras.io). Scikit-learn (scikit-learn.org) is another library for machine learning methods. In this work these three frameworks are used to implement the machine and deep learning models. The reasons for using Keras and TensorFlow to programme the models are explained in figure 26. Because the libraries are more popular than comparable ones, support as well as the availability of additional modules and pre-programmed models is much greater.

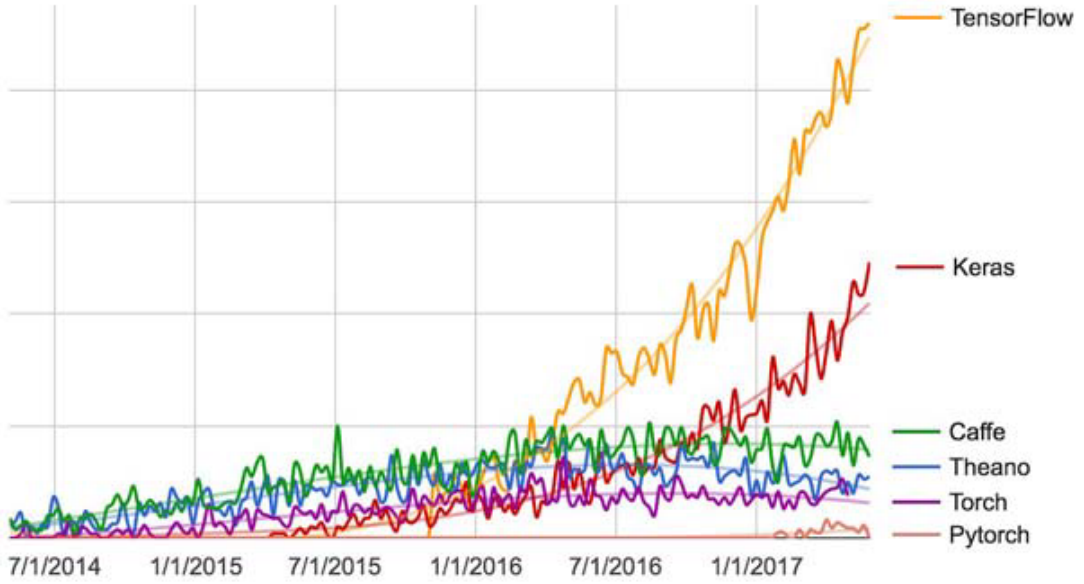


Figure 29: Google web search interest for different deep-learning frameworks over time [8].

As previously stated, the computationally intensive effort is the most significant challenge in running machine and deep learning models. In the past, the available computing power was significantly lower than today. Deep learning has advanced dramatically as a result of the continued development of the graphical processing unit (GPU). The GPU is the graphical unit of a computer and in contrast to the central processing unit (CPU) it has some advantages. The GPU is built for parallelism, whereas the CPU can only handle a few processes at a time. Hundreds of thousands of processes can be handled at once. Many processes must be calculated at the same time because a convolutional network works with images passing through layers with hundreds of neurons. Therefore, and for some other technical reasons which will not be explained here, the graphical processing unit has a clear advantage over a

CPU in the calculation of such networks. The characteristic of the GPU in comparison to the CPU is also shown in figure 30. In terms of bandwidth a GPU is much better than a CPU.

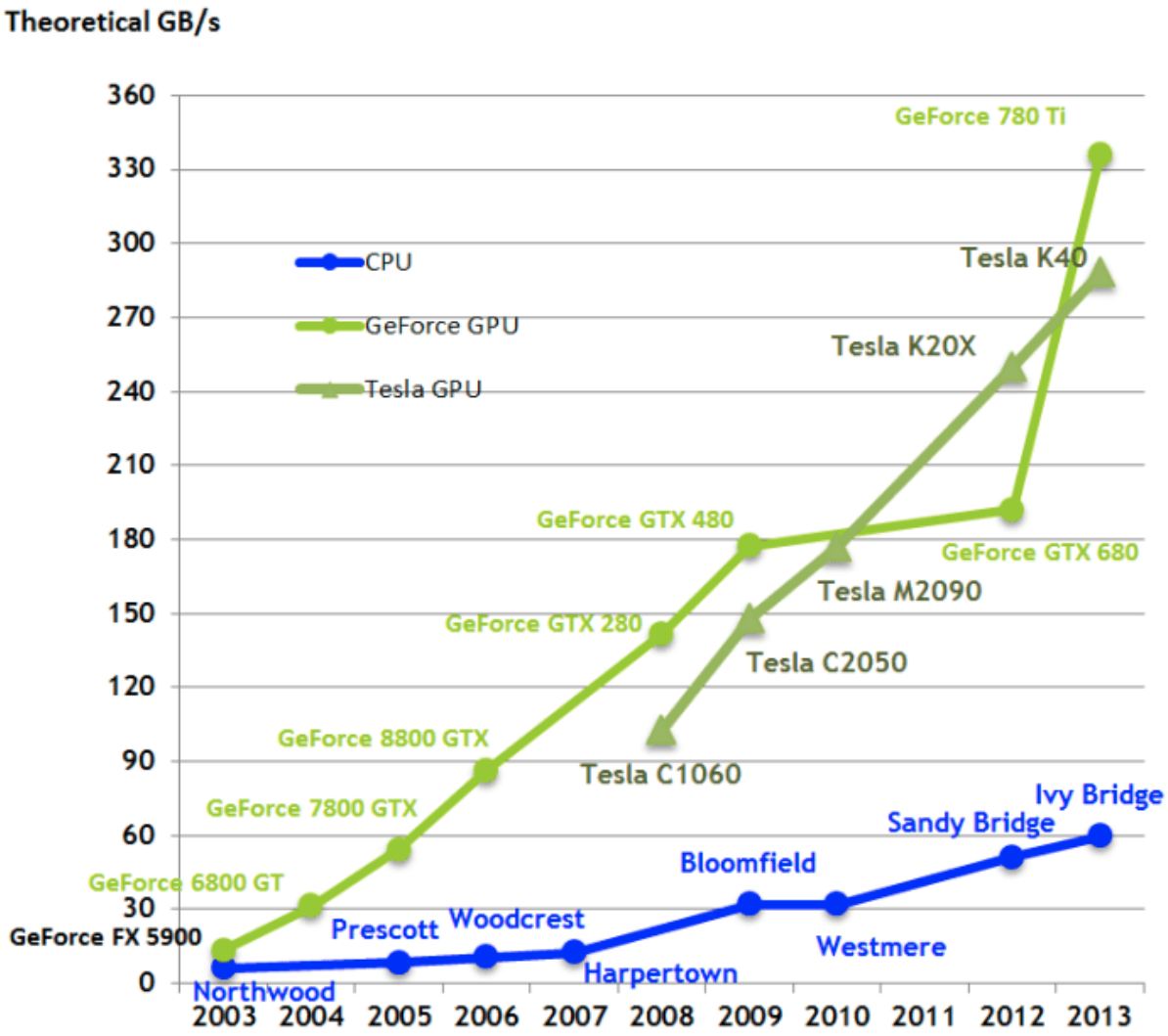


Figure 30: Comparison of GPU and CPU bandwidth[1]

To get the advantage of a GPU working with Python and deep neural networks, an instance from Amazon Web Service (AWS) (aws.amazon.com) is used. AWS offers Ubuntu servers with preinstalled Python software and libraries like Keras, TensorFlow and scikit-learn.

[1] CPU vs GPU, medium.com

5. Verification and scores

This chapter describes the verification of the used methods. Especially the scores for the verification of deep learning methods are the usual ones for these applications. Scatter plots are used for the verification of the machine learning and threshold methods.

5.1 Verification of cloud cover

For the threshold and machine learning methods so-called scatter plots are used. In Python there is a function called heat maps. This function is especially useful for the evaluation of classification problems of this kind. Two examples of different heatmaps are illustrated in figure 31.

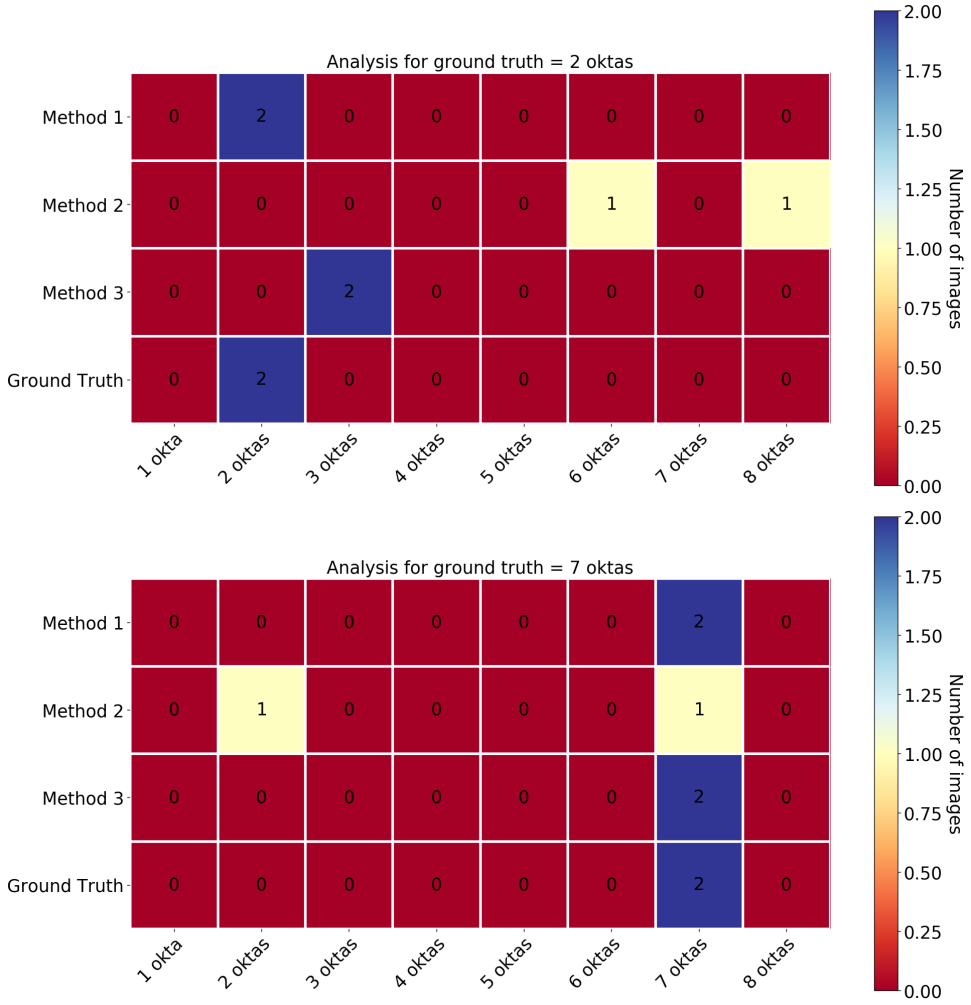


Figure 31: Example of two heatmaps with three different methods and the ground truth. In both examples the total number of analysed images is two.

The abscissa of the plot shows the possible cloud cover. It ranges from one to eight oktas. Where one okta means few clouds and eight oktas completely covered by clouds. As the dataset is only filled with images where clouds can be seen, the cloud cover of clear sky (zero okta) is not included. The vertical axis shows the methods which are used to analyse the cloud images. The headline of the plot specifies the true value of the oktas of the cloud image. This value equals the value of the ground truth.

In this thesis it is assumed that the analysed mask of the ground truth from the SWIMSEG dataset always corresponds to the actual value of the oktas of the cloud images. This assumption is made, to have a valid ground truth value to verify the models.

The colour of the field always represents the number of analysed oktas per image from the method. Therefore, the field from the ground truth is always blue as it contains all the images with the same analysed oktas.

The example in the two heat maps above shows a hypothetical analysis of two cloud categories. In the first example, Method 1 analyses two oktas for two cloud images, just as many as the "Ground Truth" is. This means that the analysis for all images from this method was true. Method 2 however, analyses one image with six oktas and the analysis for the other one is eight oktas. So overall this method performs poorly. The last method, Method 3, has two analyses in the field of three oktas. As the analyses just differs one category and the two images are both in the same category, the method performs quite well.

In summary, the more blueish the colour of the field is, the more images are assigned to this category. And the closer the fields are to the Ground Truth field, the more accurate the analysis was. In the other example, two out of three methods achieve a perfect score with two analyses in the same category as the ground truth. Whereas the other category analyses on image with seven oktas and the other with two oktas.

Furthermore, some standard verification scores like root mean square error (RMSE), mean absolute error (MAE) Bias are used to verify the methods.

$$\text{BIAS:} \quad \frac{1}{N}\sum_{i=1}^{N}(F_i - O_i) \qquad \text{MAE:} \quad \frac{1}{N}\sum_{i=1}^{N}|F_i - O_i| \qquad \text{RMSE:} \quad \sqrt{\frac{1}{N}\sum_{i=1}^{N}|F_i - O_i|} \tag{4}$$

The calculation of the three scores are shown in equation 4 [30]. The Bias is the systematic distortion of a sample. It is a measure of the overall reliability of the forecast. But, it does not provide an information about the magnitude of the error.

- BIAS > 0: Forecast value to high
- BIAS < 0: Forecast value to low

The MAE is used to calculate the accuracy of the forecast and it does give information about the magnitude of the error. Also the RMSE is a score for the magnitude of the error. As seen in equation 4, it is the square root of the MAE. It therefore weights large errors stronger than smaller ones.

$$\text{Ratio pixels:} \quad \frac{p_c}{p_c + p_s} \tag{5}$$

All these scores are calculated from the ratio of sky and cloud pixels. The calculation of the ratio is illustrated in equation 5. Where p_c is the total number of cloud pixels and p_s is the total number of sky pixels. For eight oktas, the cloud cover will be 1 as there are only cloud pixels in an image. For zero oktas the value is 0 if there are no cloud pixels. The verification scores are therefore not applied to the different categories (oktas) but to numbers between 0 and 1. The threshold values for each okta category is shown in table 6.

An example: Assume that the ratio for the ground truth of a cloud image is 0.26, this results in two oktas (2/8) cloud cover. Then two methods analyse the cloud image and method 1 calculates 0.25 (1/8) ratio between sky and cloud pixels and method 2 analyses 0.30 (2/8). Although the second method performs worse than the first in this example, method two analysed the same value for the oktas as the ground truth. In this case, the second method will make a correct analysis for the oktas, even though it is further away from the ground truth observation than the first method. Applying the scores not on the analysed oktas of the cloud images but on the ratio of the pixels solves problems like in this example.

Ratio	**Oktas**	**Ratio**
0 =	0/8	= 0
0 <	1/8	< 0,125
0,125 <	2/8	< 0,25
0,25 <	3/8	< 0,375
0,375 <	4/8	< 0,5
0,5 <	5/8	< 0,625
0,625 <	6/8	< 0,75
0,75 <	7/8	< 0,99
1 =	8/8	= 1

Table 6: Calculation of oktas by the ratio of pixels

5.2 Verification of cloud classification

To evaluate the classification models, it is always important to know how many samples from the analyses could be assigned to the true classes. A score to estimate it quantitative, is the accuracy in equation 6. [7]

$$\text{Accuracy:} \quad \frac{tp + tn}{tp + fn + fp + tn} \tag{6}$$

Variables: tp = true positive, tn = true negatives, fp = false positive, fn = false negatives

The accuracy quantifies the number of correct analyses (tp and tn) of the class that belongs to it. Take a cloud/sky classification model for example. If the accuracy is 0.9, 90% of the cases the model fits the observation.

One way of demonstrating the accuracy of a model overall the classes is to use a confusion matrix [16]. In Stanski et al. (1990) [30] this is also referred to as the contingency table. Each row of the matrix represents the analysed class (analysed label) of the model and each column represents the observation (true label). The boxes of the matrix show the value of the accuracy of the model. A perfect model will have nothing but values of one in the diagonal of the matrix.

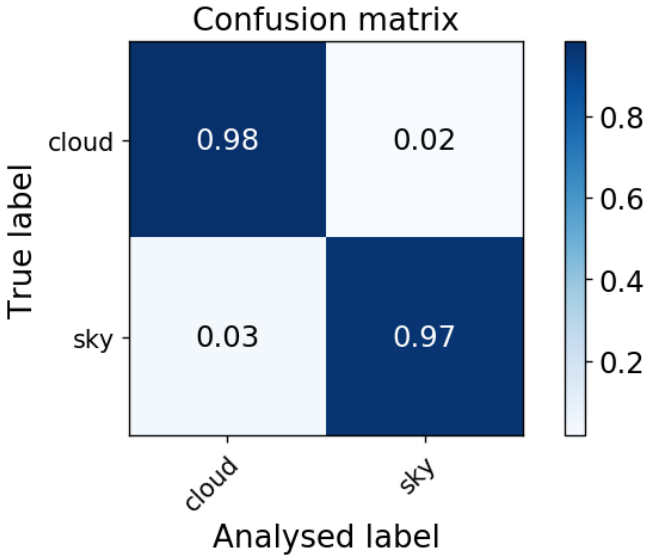


Figure 32: Confusion matrix with simple cloud/sky classification

An example of a binary confusion matrix is shown in figure 32. This is an example of a support vector machine analysing cloud and sky pixels. The accuracy of each label is illustrated in the boxes and the bluer a box is, the better the model performs. In this case, the accuracy of the model is 98% and 97%, respectively.

6. Results

In this chapter the results of algorithms, which are applied on the different datasets, are discussed. The results of the methods for determining cloud cover are described first, and two different approaches are discussed. One is the qualitative verification. Since the determination of cloud cover and cloud types by humans is based on visual observation, this technique can also be applied for the verification of the cloud cover determined by the methods.

On the other hand, to verify the whole dataset, qualitative verification is not suitable as it contains over 1000 images. In this case, quantitative methods are used to evaluate the data sufficiently. For this analysis, the methods and scores described in the previous chapters are used.

The second part of this chapter describes the results of the determination of the cloud types. The main verification tool is the confusion matrix for a deeper understanding of the different CNN models the training/validation accuracy and loss (learning curves) are also analysed.

6.1 Cloud cover

As mentioned above, the visual aspect is an important factor for an initial assessment of the functionality of the methods. The comparison between the analysed image with the real image or with the ground truth alone is sufficient to make a statement about the accuracy of the algorithm. In the following figures, the visual aspect is discussed first, followed by quantitative verification scores. First, some images from the SWIMSEG dataset are analysed. The main characteristic of this dataset is that ground truth images are already available. Second, few images are analysed, which contain not only clouds but also landscapes or infrastructure like buildings or power lines.

An example of the results from the different methods can be seen in figure 33. The first row shows the original image of the clouds from the SWIMSEG dataset and the corresponding ground truth. The second image shows the k-number of clusters of the analysis with the kmeans clustering method. The clustered image is further used for the Hybrid method. In the second row, one can see the two threshold methods, Euclidean Geometric Distance (EGD) and the Normalised Red Blue Ratio (NRBR). The last image shows the result of the k-nearest neighbour (KNN) method. The results of the hybrid method and the support vector machine (SVM) are illustrated in the last row. The analysed cloud cover is given in oktas in parentheses.

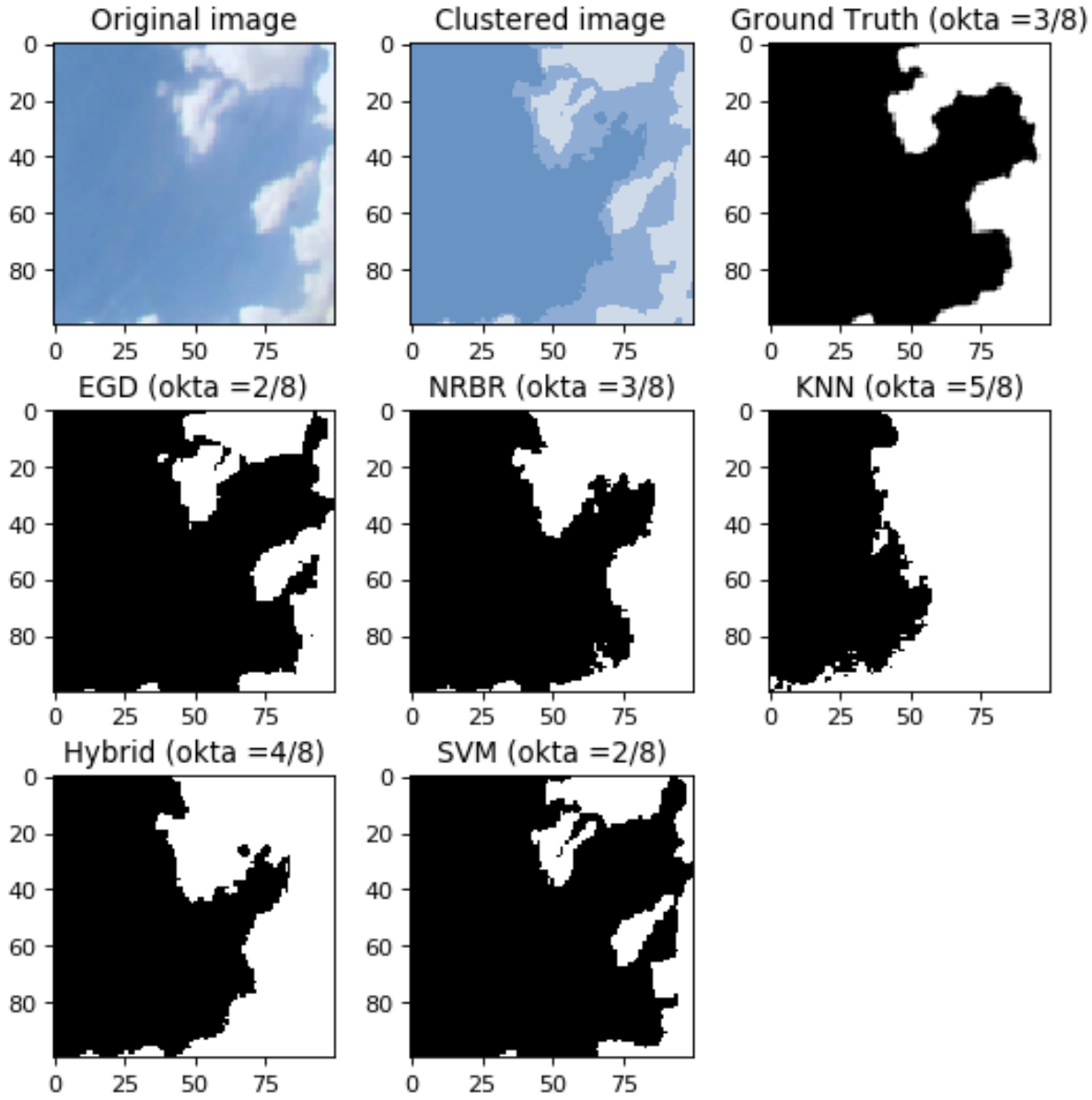


Figure 33: Example of the results from the cloud cover analysis and the analysed cloud cover in oktas by the different methods.

In the next seven sections, the results are shown as a function of the corresponding ground truth cloud cover. In the results the categories from one okta of cloud cover to seven oktas are included. The cloud cover of eight oktas is excluded, because there is no ground truth image where the whole image is covered by clouds (100% cloud pixels). There is also no image with zero okta, as explained in section 5.1.

6.1.1 SWIMSEG

One okta ground truth

Figure 34 shows the result of the different methods for the ground truth cloud over of one okta. The clustered image shows high agreement with the original image. This is already a first indication for the simplicity of the cloud image. In this image there are only two larger clouds and the contrasts between blue and white are distinct. In the original image, however, there is a small white shine in the left half of the image. The transition between sky and clouds is also affected by a light white background despite the high contrasts. This example is captured accurately by two out of five methods. EGD and SVM calculate the same number of oktas as the ground truth. NRBR and KNN have difficulties with the white shimmer, the clusters of the clouds are slightly to large and therefore the hybrid method overestimates the cloud cover.

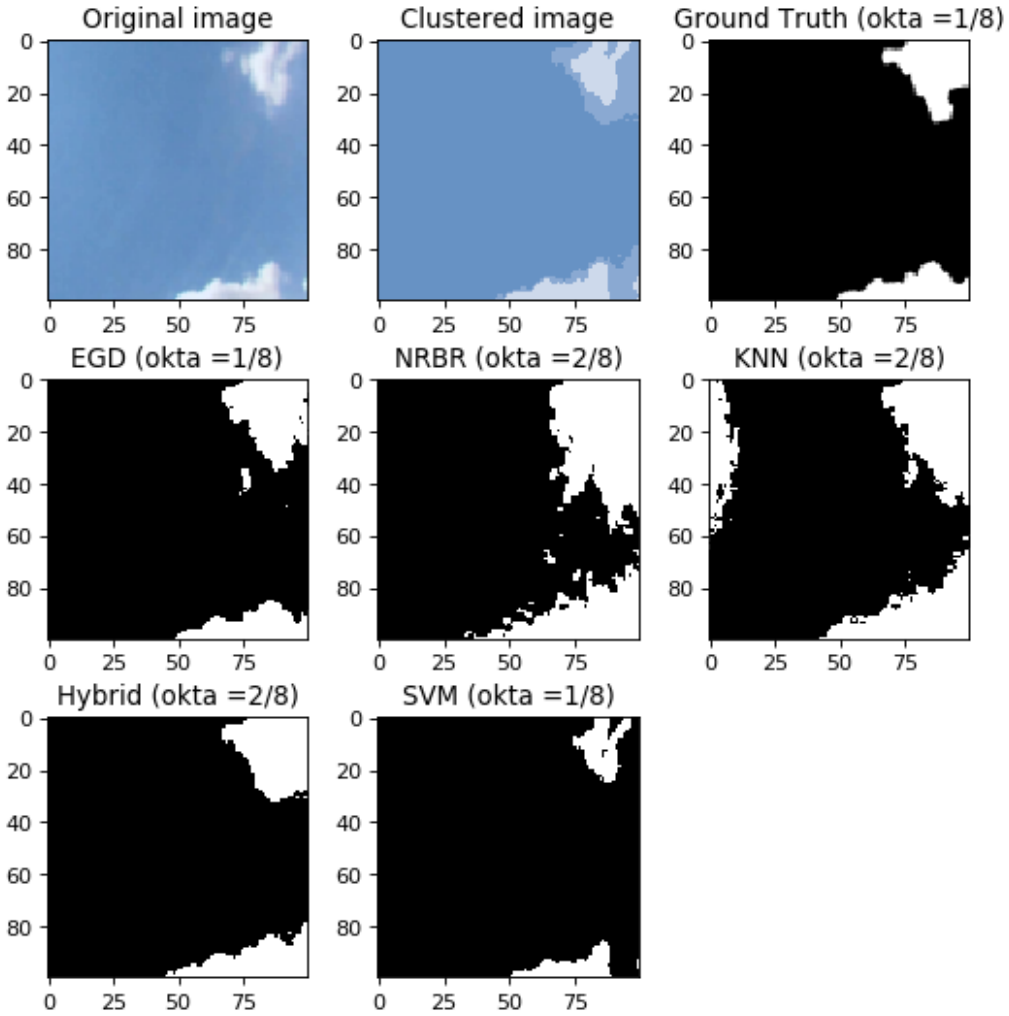


Figure 34: 1. example where ground truth = 1 (GT=1): three out of five methods analyse the true value, problems with light white shimmer in the left corner and with sky/cloud transition.

The next example is illustrated in figure 35. This image is an example, where the reflection of the sun rays in little cloud droplets distorts the results of the different methods. In the middle of the original image a small cloud and at the upper right edge a small cluster of a cloud-like structure can be seen. In addition, a white film is covering the right half of the image. This is due to the reflection of the sun rays on small water droplets. In the clustered image, this reflection becomes a major part of the whole image and therefore the hybrid method overestimates the cloud cover. Likewise, the KNN method detects most of the pixels as clouds. The two threshold methods also significantly overestimate cloud cover. Only the SVM method analyses the true value of the cloud cover correctly.

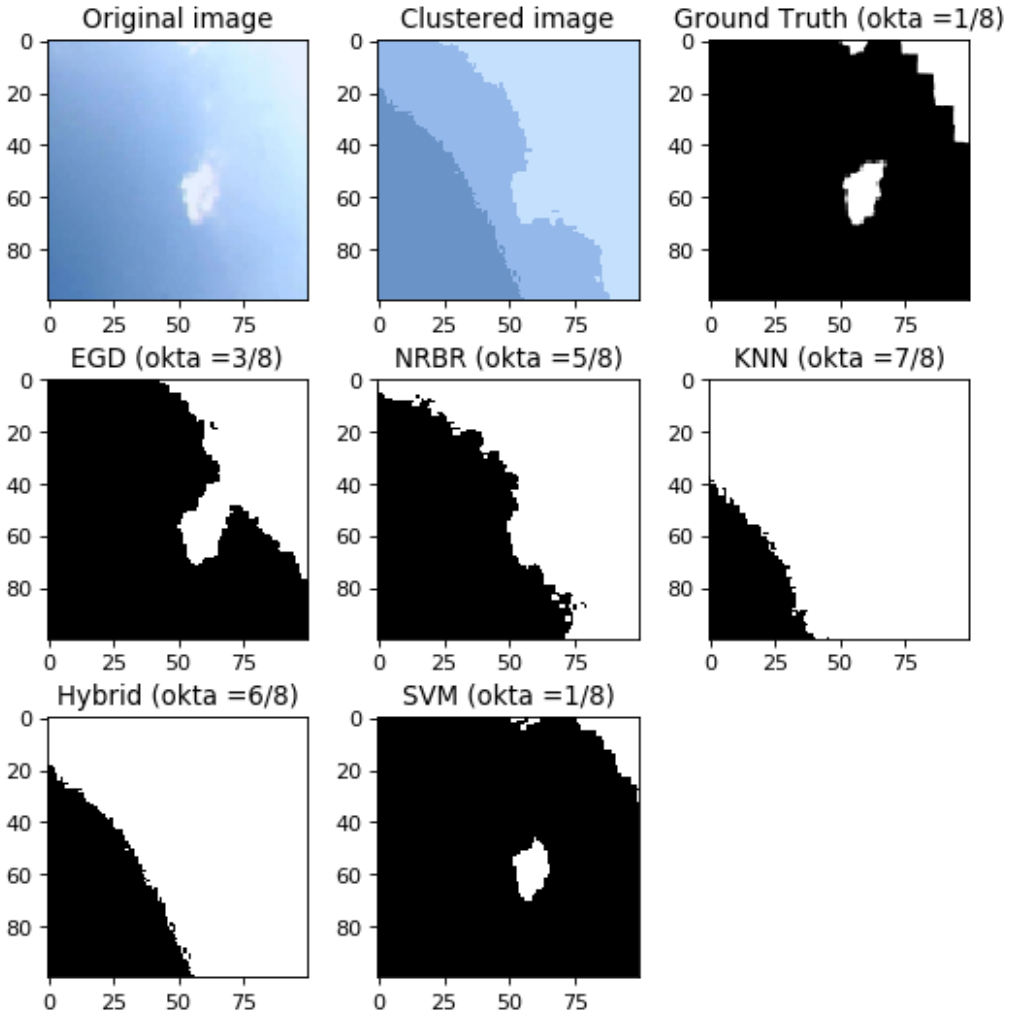


Figure 35: 2. result (GT=1): one out of five methods analyse the true value, great distortion of the cloud image by sun light reflected from small cloud droplets

The analysis of all images in this category (GT=1) can be seen in figure 36. It is particularly noticeable that SVM has assigned all images to the correct category. As a result, the analysed value from the SVM method is always the same as in the corresponding ground truth image of the original image. In addition, the two machine learning algorithms analyse the same values as the ground truth in the vast majority of cases. Whereas the two threshold methods overestimate the cloud cover, especially the NRBR method.

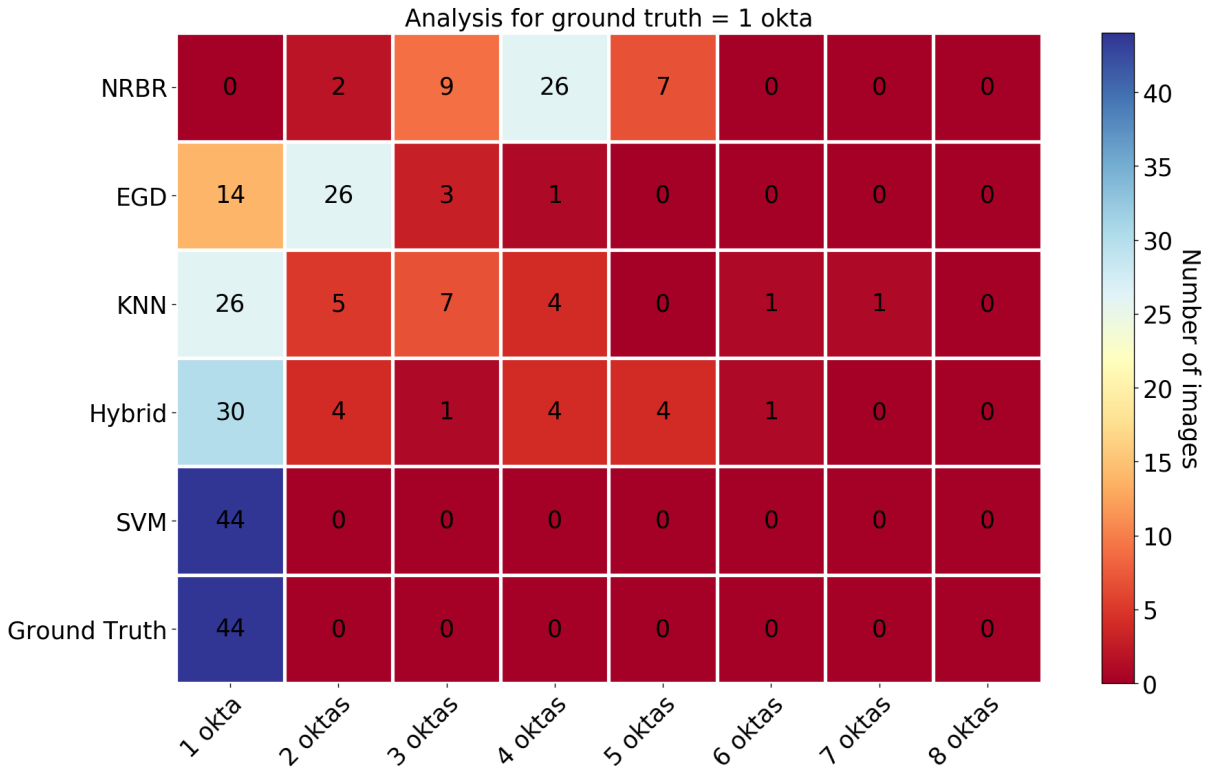


Figure 36: Heatmap for all methods where ground truth equals one okta. The total number of analysed images is 44.

Score	NRBR	EGD	KNN	Hybrid	SVM
MAE	0.128	0.012	0.047	0.052	0.002
RMSE	0.357	0.112	0.216	0.229	0.046
BIAS	0.343	0.082	0.096	0.079	-0.036

Table 7: Verification scores for one okta.

Table 7 shows the different verification scores of the individual methods with reference to this category (GT=1). It should be noted again that these scores are calculated using pixel ratios rather than oktas (section 5.1). The scores are consistent with the results shown previously. The verification scores confirm that SVM makes the best analysis in this category, while NRBR is the furthest from the observations. This analysis also shows a small difference from the previous analysis. While the hybrid method previously showed a high level of agreement, the verification scores show a different image. Back in figure 36, the hybrid

method analyses four oktas for four images, five oktas also for four images and even six oktas for one image. This incorrect analysis results in high MAE and even higher RMSE.

Two oktas ground truth

The next example where ground truth equals two oktas can be seen in figure 37. This image contains a group of clouds in the lower right corner, but in general there are many whitish cloud pixels in this image. This high proportion of white colour is also noticeable in the results of the individual methods. Already in the clustered image, the blueish portion of the three clusters appears to have a high white component. While the threshold methods as well as the SVM analyses the cloud cover relatively accurately, the KNN and Hybrid methods overestimate it significantly. It is only the EGD method which analyses the same value as the ground truth and is therefore in this image the best one. SVM slightly underestimates the cloud cover.

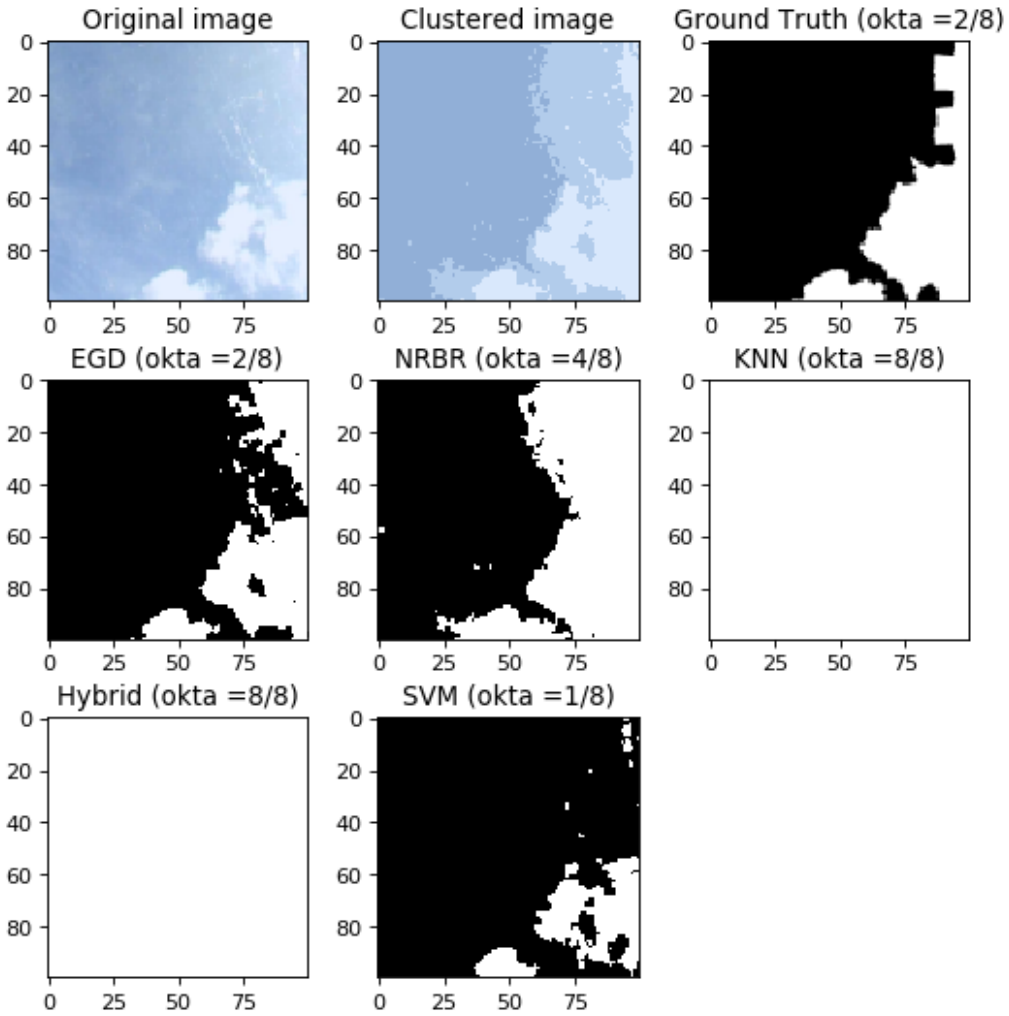


Figure 37: 1. result (GT=2): one out of five methods analyse the true value, colours of clusters are too whitish

The case in figure 38 is an example, where nearly all five methods analyse the cloud cover accurately. In the original image one can see a white cloud in the lower right corner and two small ones on the top right as well as in the left corner. The bottom corner of the image is covered by a thin white veil. This veil is not declared as a cloud by the ground truth and this fact is also assumed for most methods. Only the NRBR method analyses some clouds in this area.

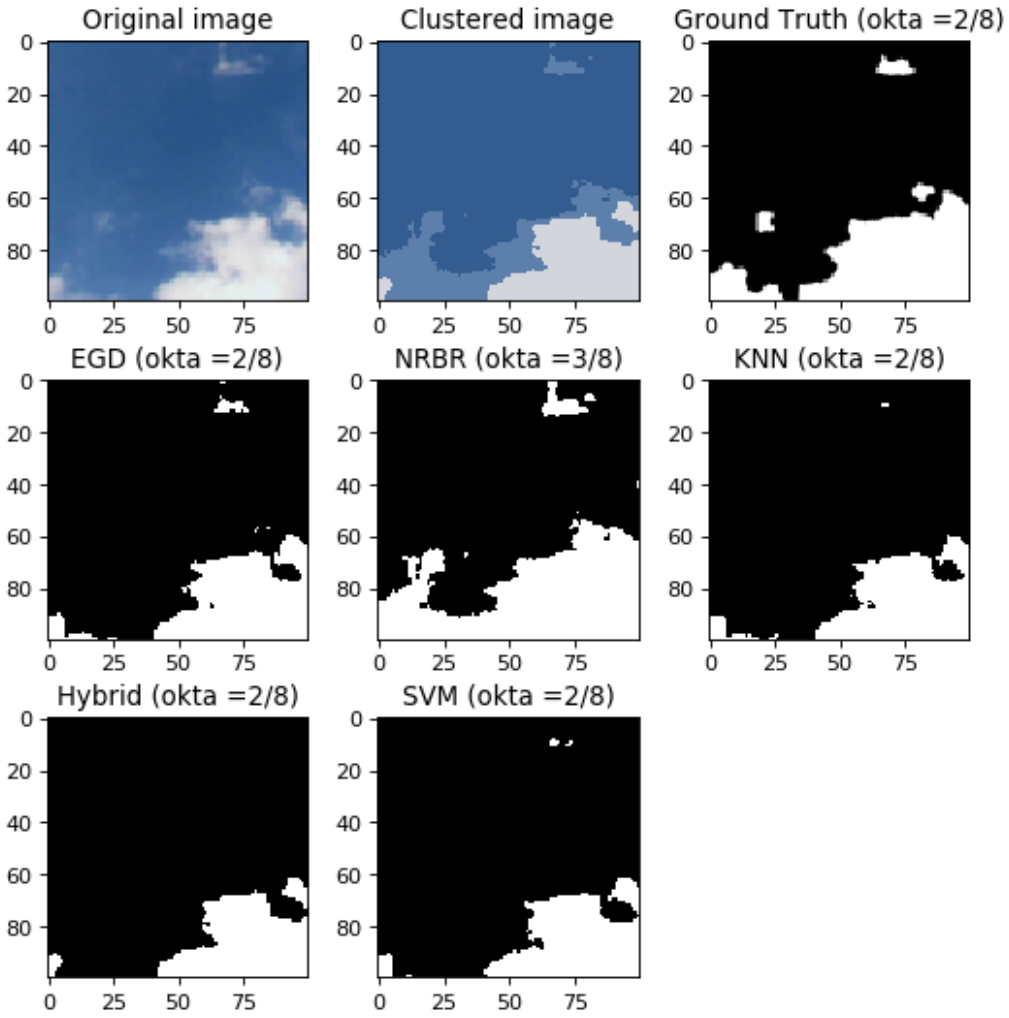


Figure 38: 2. result (GT=2): four out of five methods analyse the true value, good contrast between clouds and sky pixels

For a quantitative verification for the analysis of two oktas, figure 39 provides information. A total of 68 images are in the dataset, where the ground truth is equal to two oktas. A first look shows that especially the threshold method EGD performs particularly well, the analysis is correct for 61 images. The second most images are assigned to the same number of oktas by the SVM. Even though many images are in the wrong category, the deviation is low. In general, the machine learning methods underestimate the cloud cover. NRBR, on the other hand, always overestimates it significantly.

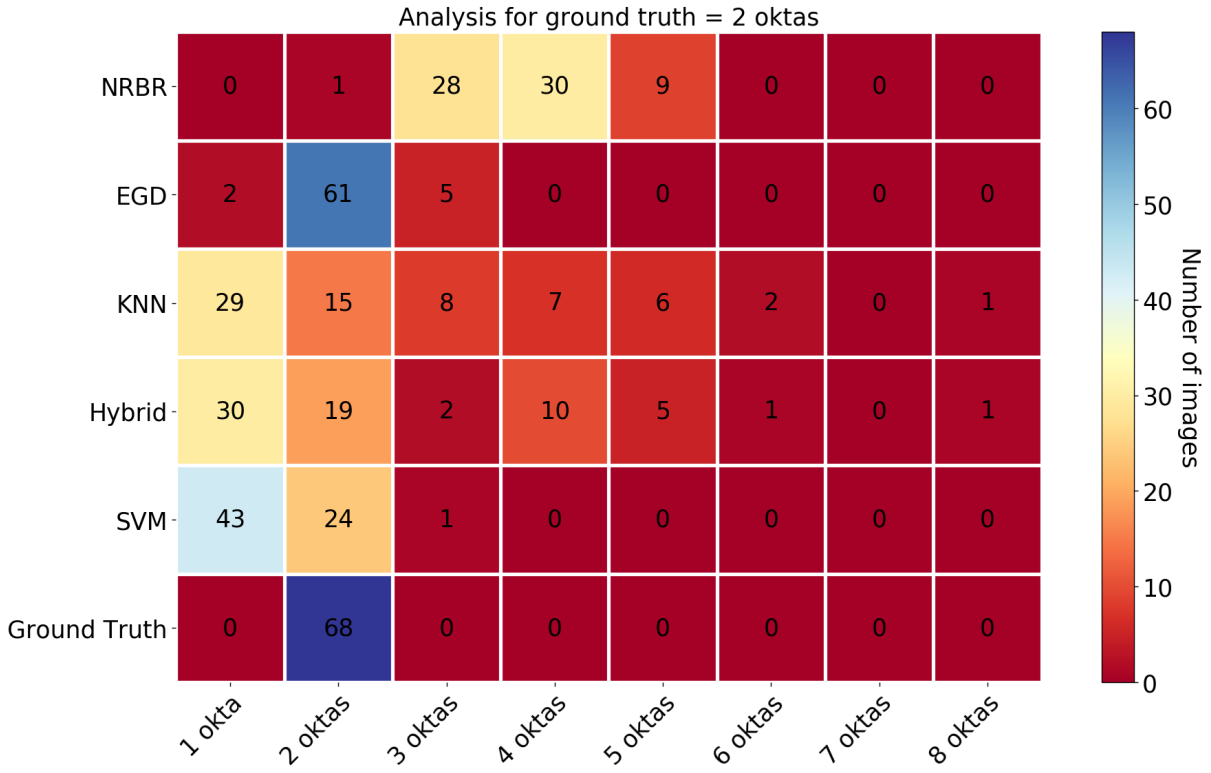


Figure 39: Heatmap for all methods where ground truth equals two oktas. There are a total of 68 images in this category.

Score	NRBR	EGD	KNN	Hybrid	SVM
MAE	0.053	0.002	0.045	0.043	0.013
RMSE	0.231	0.048	0.212	0.207	0.112
BIAS	0.213	0.001	0.032	0.005	-0.094

Table 8: Verification scores for two oktas.

Looking at the verification scores in table 8, also the EGD method shows the best performance. Whereas the other threshold method, NRBR, achieves the poorest scores. As also seen in the previous figure, the SVM underestimates the cloud cover, hence the negative BIAS. Worth mentioning is the fact, that the MAE from the SVM is very low. This shows that although many images were assigned to the wrong class, the deviation in terms of pixel ratio is low.

Three oktas ground truth

The next example of the analysis is illustrated in figure 40. In the bottom left corner is a patch of clouds with a greyish colour component. In the upper right part are no clouds, but the white part is caused by the scattering of sun rays. This example is captured quite accurately by the two threshold methods, especially the EGD algorithm resembles the Ground Truth well. In contrast to that, are the three machine learning techniques. The SVM analyses the

whitish part of the images as clouds and for the KNN and Hybrid method the image is to white at all.

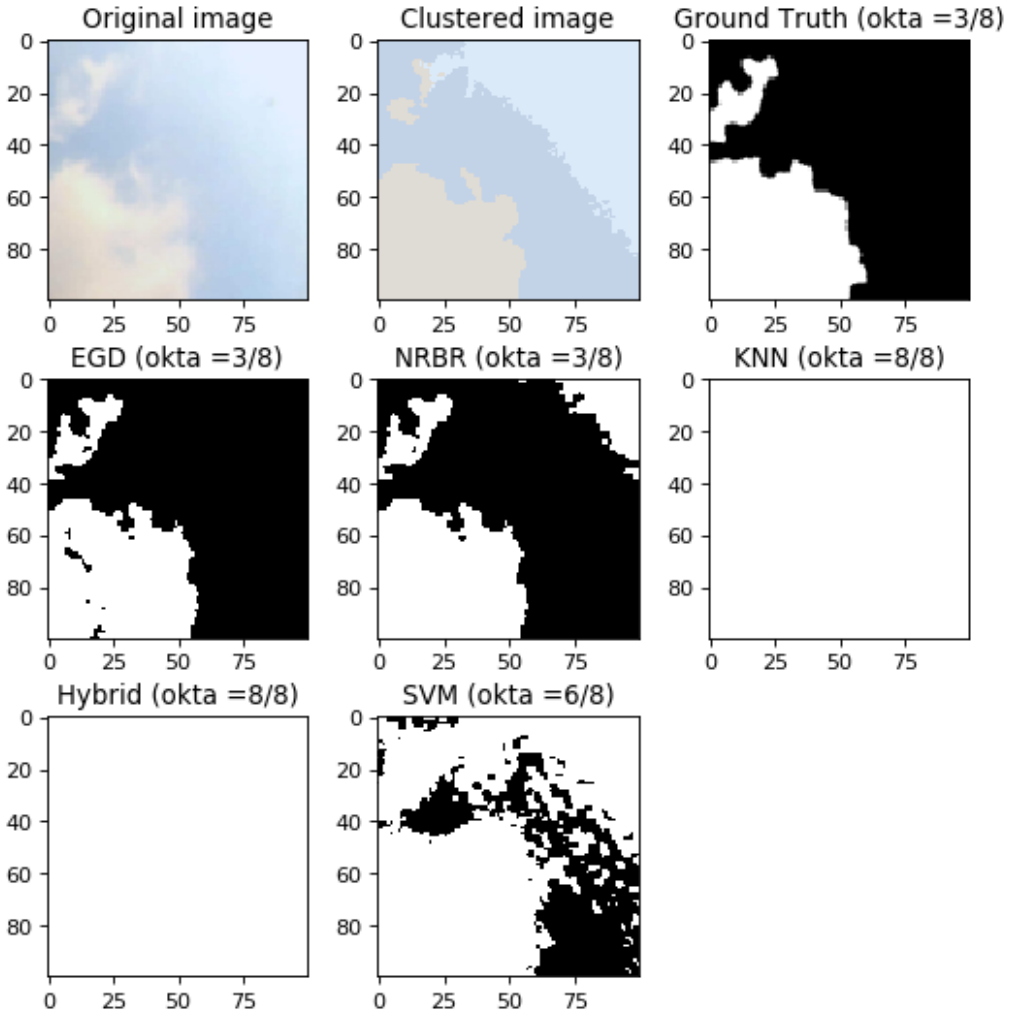


Figure 40: 1. result (GT=3): two out of five analyse the true value, large disturbance from sun light scattering

Figure 41 is an example where the contrast between cloud and sky is very sharp. The cloud is situated in the top middle of the image. Taking the analysis of the oktas, three out of five methods analyse the true value. However, since even the methods that are just barely off the ground truth, it is worth looking at the exact values of the pixel ratios.

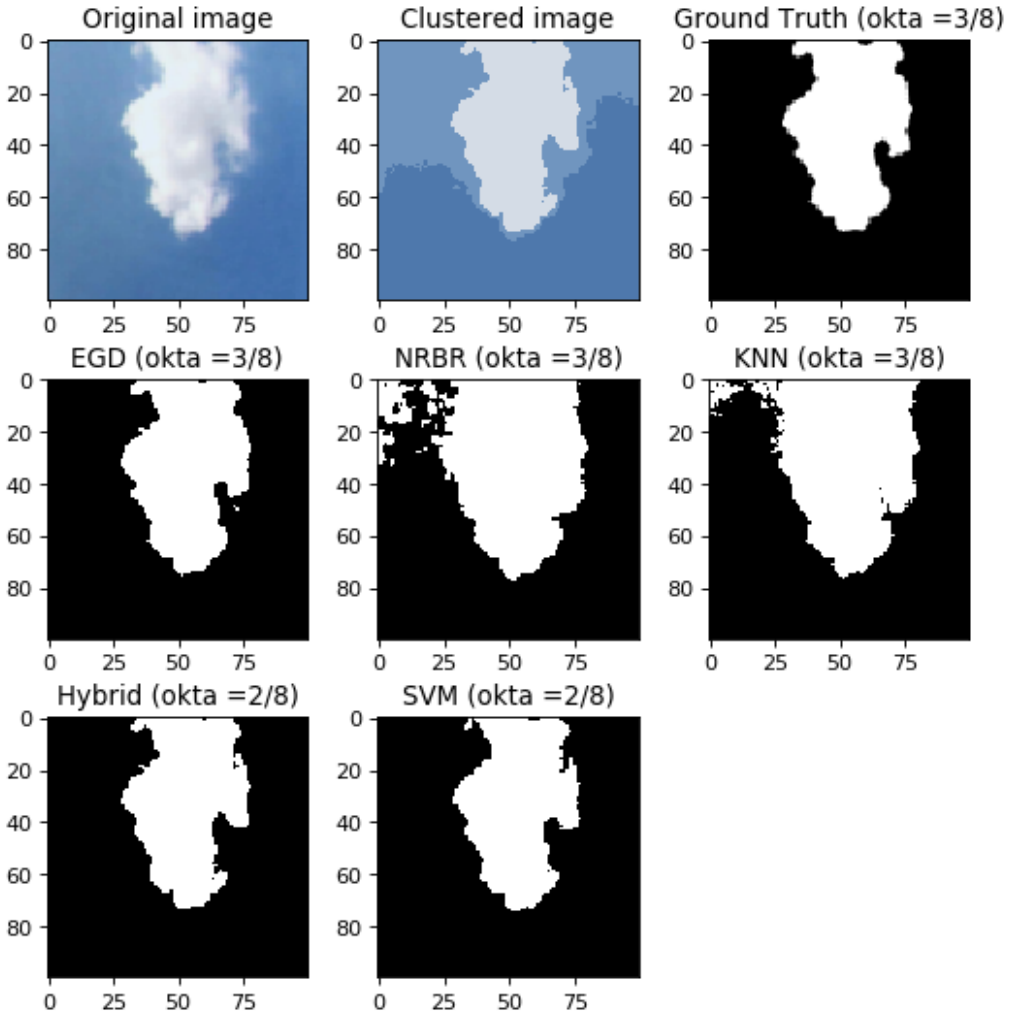


Figure 41: 2. result (GT=3): three out of five analyse the true value, high contrast between cloud and sky

GT	NRBR	EGD	KNN	Hybrid	SVM
0.26	0.35	0.25	0.33	0.24	0.23

Table 9: Cloud and sky pixel ratio for figure 41; green: analyse right okta value; red: wrong analysis

Table 9 shows the results of the individual methods in relation to the ratio of pixels. The methods that were correct, according to the okta analysis, are marked in green, those that analyse the wrong number of oktas in red. Taking the values from the table, the deviation of the two wrong methods is small. Compared to NRBR and KNN, the hybrid and SVM methods even performed better. But it is due to the threshold values for the of okta categories, that

Hybrid and SVM are assigned to another category. By comparing the values of Hybrid and SVM with table 6, these values are just slightly above threshold for the 2/8 oktas category.

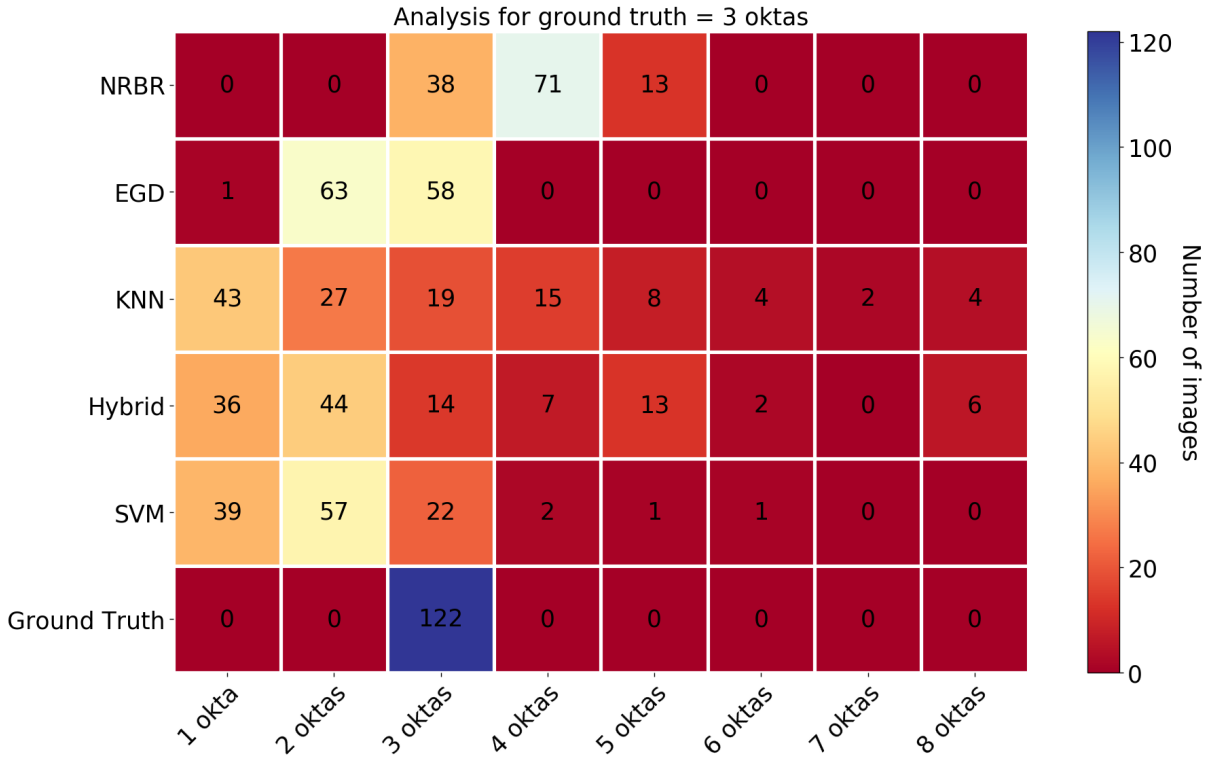


Figure 42: Heatmap for all methods where ground truth equals three okta. The total number of analysed images is 122.

Score	NRBR	EGD	KNN	Hybrid	SVM
MAE	0.014	0.007	0.06	0.069	0.028
RMSE	0.118	0.082	0.245	0.263	0.168
BIAS	0.099	-0.068	-0.052	-0.065	-0.132

Table 10: Verification scores for three oktas.

The overall statistics in figure 42 and in table 10 shows, that none of the methods used is in good agreement with the ground truth. The scattering of the methods is broad in the heatmap, only the individual verification scores indicate small differences. Taking the EGD method, more than half of all images are assigned to the wrong category. But, the verification scores MAE and RMSE have low values, which indicates rather good performance. The Hybrid method shows the poorest results in the overall statistics and it is again the SVM which underestimates the cloud cover.

Four oktas ground truth

In the next example (figure 43) a big grey cloud is situated in the top left corner of the images with small patches of clouds on the bottom. Four out of five methods analyses the true value of cloud oktas, only the EGD method analyses three oktas. The original image is very similar to the ground truth, but a few pixels are missing or have been assigned to the sky, especially in the upper right area.

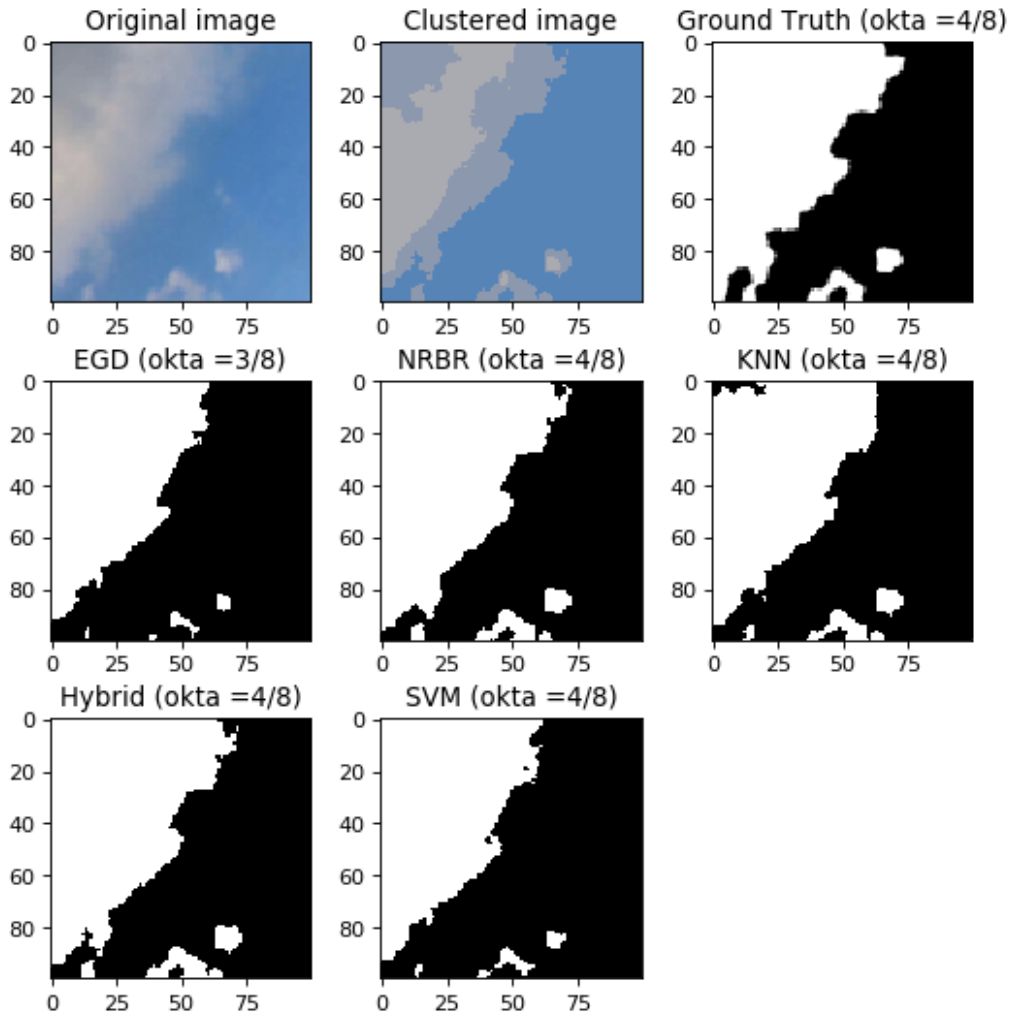


Figure 43: 1. result (GT=4): four out of five analyse the true value, good agreement with Ground Truth

In figure 44 half of the image is covered by clouds with saturated white colours. While cluster analysis agrees well with ground truth, the methods KNN and hybrid significantly overestimate cloud cover. Whereas the other three methods analyse the true value of oktas.

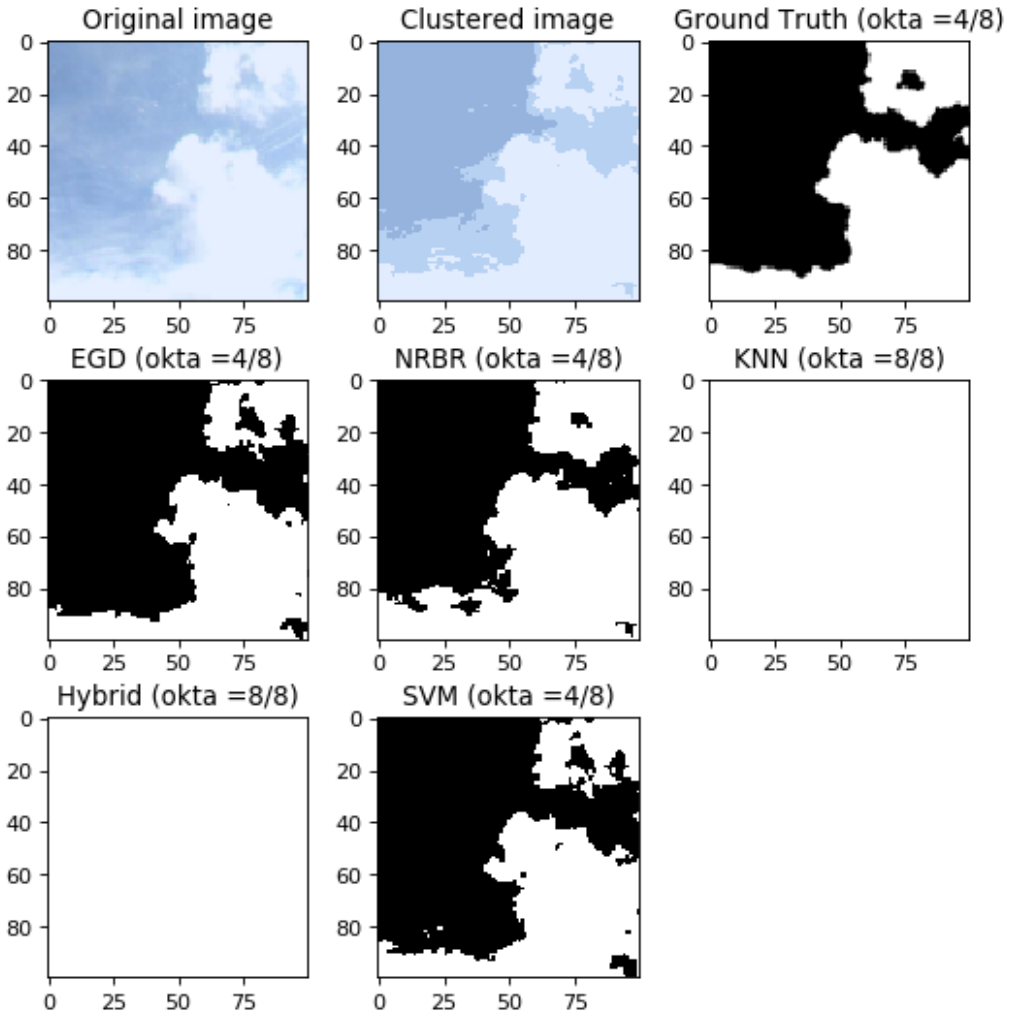


Figure 44: 2. result (GT=4): three out of five analyse the true value, KNN and Hybrid totally overestimate the number of oktas

The heatmap in figure 45 shows, that the threshold methods NRBR and EGD assign most of the images to one category. NRBR even to the same category as the ground truth. Whereas the distribution of the machine learning methods is widespread. Particularly the KNN and hybrid methods analyse a lot of images with seven and eight oktas, like the example above in figure 44. The SVM method underestimates the cloud cover in most of the images. The underestimation also reflects in the BIAS of the method, which can be seen in 11. As expected, the NRBR and the EGD have the lowest values of verification scores of all the methods.

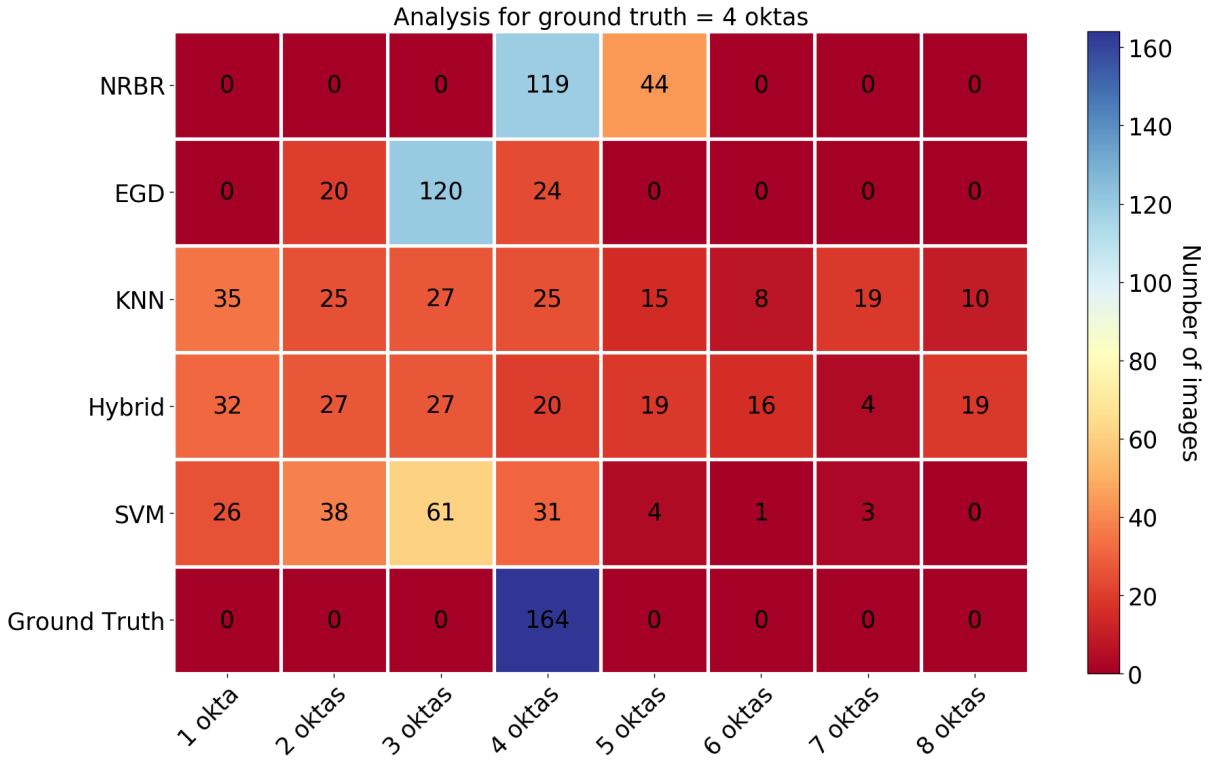


Figure 45: Heatmap for all methods where ground truth equals four oktas. There are a total of 164 images in this category.

Score	NRBR	EGD	KNN	Hybrid	SVM
MAE	0.004	0.019	0.088	0.098	0.045
RMSE	0.064	0.137	0.297	0.312	0.212
BIAS	0.034	-0.124	-0.041	-0.04	-0.161

Table 11: Verification scores for four oktas.

Five oktas ground truth

In figure 46 the clouds are situated in the top left corner of the image. None of the five methods analyse the true value of the cloud oktas. Upon closer inspection, the analysed images of SVM and EGD agree quite accurately with the original image. But the Ground Truth does not match the original image very well instead. The hybrid and KNN methods do not detect the dark cloud pixels as clouds and therefore underestimate the cloud cover overall.

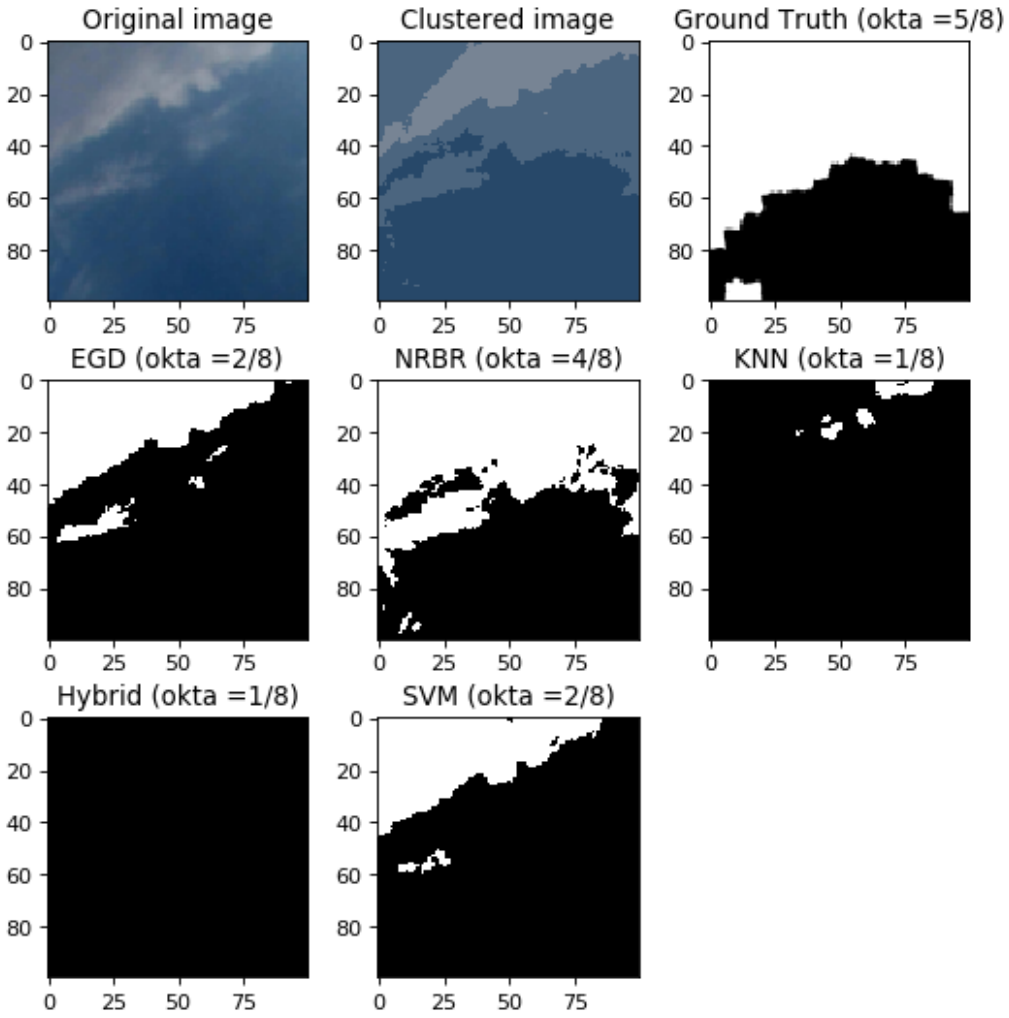


Figure 46: 1. result (GT=5): zero out of five analyse the true value, colours to dark to analyse them as cloud pixels

In the next example, which can be seen in figure 47, thick patches of clouds are scattered all over the image. The contrast between the sky and clouds is clear. But overall, the image is covered by a light white shimmer. Two out of five methods analyse the true value compared to the ground truth. The analysis of EGD is in good agreement with the original image and with the ground truth, the value of the ratio is slightly above the threshold for five oktas. The KNN and Hybrid methods overestimate the cloud cover significantly. The analysis of the SVM resembles the ground truth.

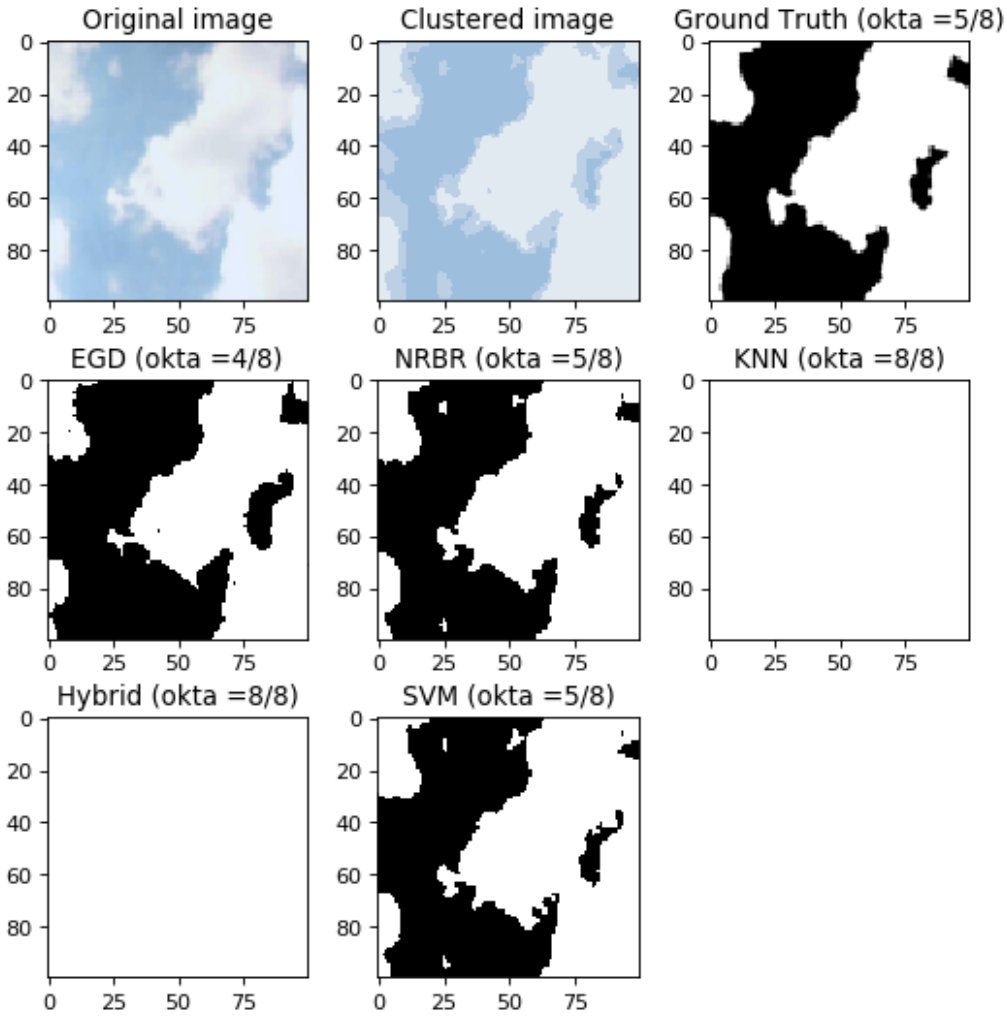


Figure 47: 2. result (GT=5): two out of five analyse the true value, mostly good contrast between clouds and sky, high white content

The heatmap (figure 48 and verification scores in table 12 show, that the EGD method performs better than the other methods. In total 137 images are assigned to the right category. This also reflects in the low MAE, RMSE and Bias of the NRBR method. The low BIAS of the EGD method is due to the assignment of most of the images in the category of three and four oktas. Again, KNN, hybrid and SVM are scattered over many categories, especially the first two methods assign a lot of images to the seven and eight oktas category, hence the high RMSE.

Score	**NRBR**	**EGD**	**KNN**	**Hybrid**	**SVM**
MAE	0.005	0.041	0.097	0.108	0.059
RMSE	0.071	0.201	0.312	0.329	0.244
BIAS	-0.029	-0.184	-0.062	-0.05	-0.182

Table 12: Verification scores for five oktas.

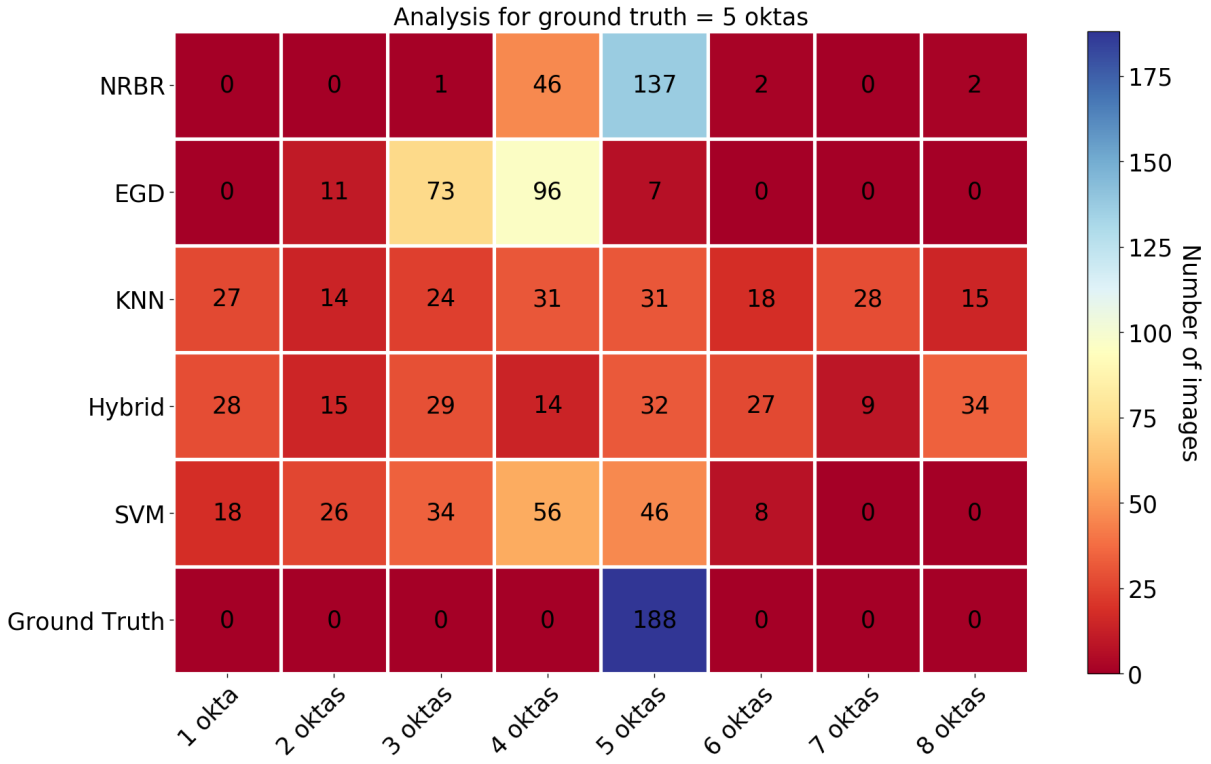


Figure 48: Heatmap for all methods where ground truth equals five oktas. In total there are 188 images in this category.

Six oktas ground truth

In the first case, figure 49, large parts of the image are covered by thick white cirrus clouds. On the left edge the cloud is also a little frayed. The ground truth does not show it as part of the clouds. In general, the different methods agree well with the ground truth, only EGD and Hybrid underestimate the cloud cover to some extent.

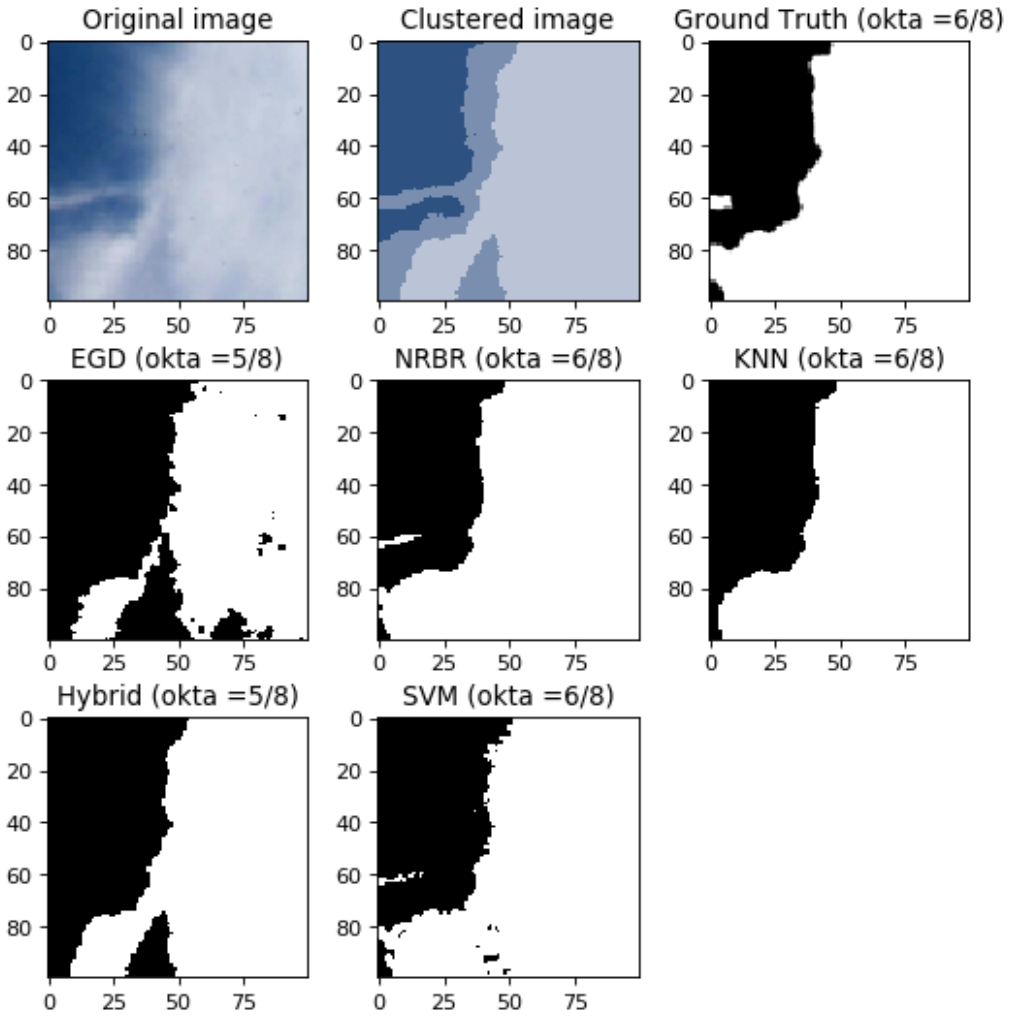


Figure 49: 1. result (GT=6): three out of five analyse the true value, thin cirrus clouds with good contrast, slight underestimation of EGD and Hybrid

In the second case (figure 50), the sky is covered by many small patches of clouds. This is a typical example of altocumulus clouds. Three out of five methods capture most of the cluster accurately, whereas for the KNN and Hybrid method the colours are to whitish to distinct between sky and clouds.

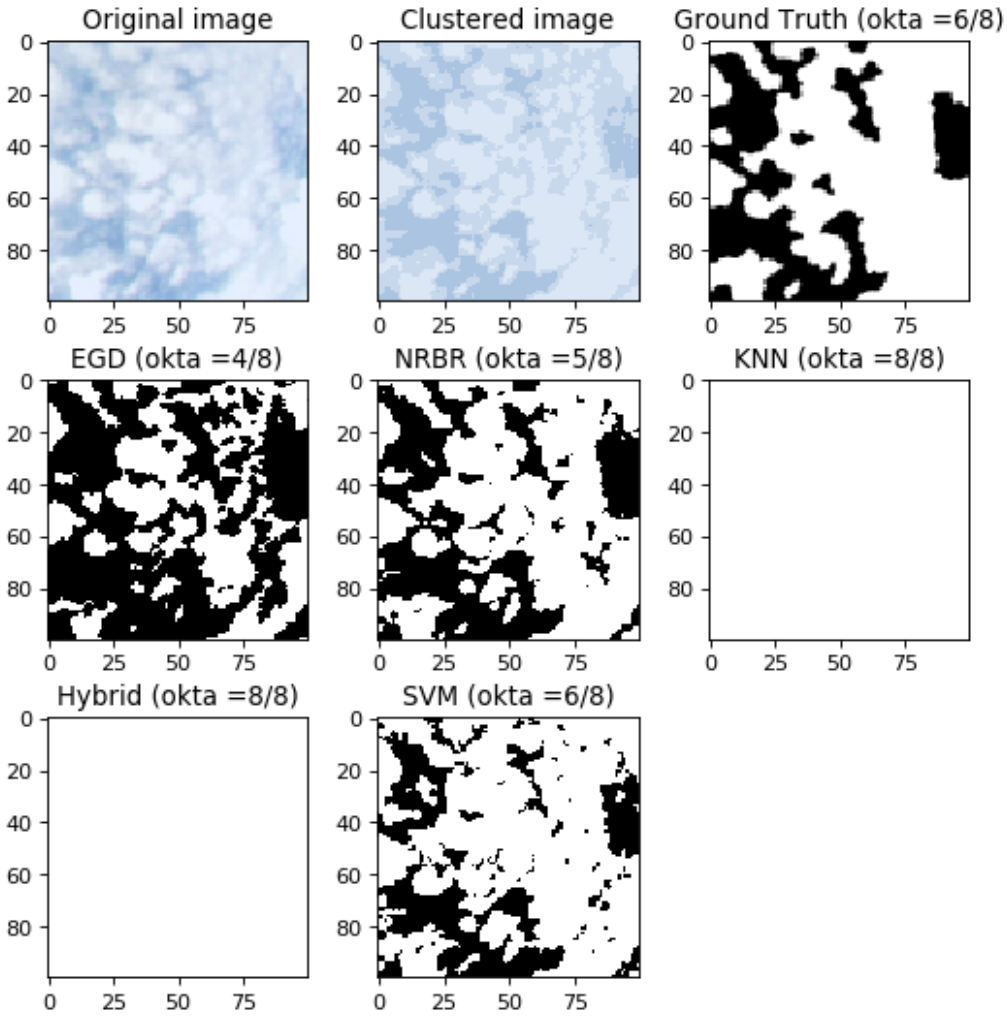


Figure 50: 2. result (GT=6): one out of five analyse the true value, small clusters of clouds

Figure 51 and table 13 show the statistical verification, and the conclusion is similar to that of the previous categories. The two threshold methods show the best verification scores, although a lot of images are assigned to the wrong category. But, the scattering of these two methods is not as big as the one from all the machine learning methods. Significant is also the negative bias of all methods.

Score	**NRBR**	**EGD**	**KNN**	**Hybrid**	**SVM**
MAE	0.015	0.088	0.092	0.094	0.065
RMSE	0.122	0.296	0.303	0.307	0.255
BIAS	-0.087	-0.282	-0.095	-0.098	-0.147

Table 13: Verification scores for six oktas.

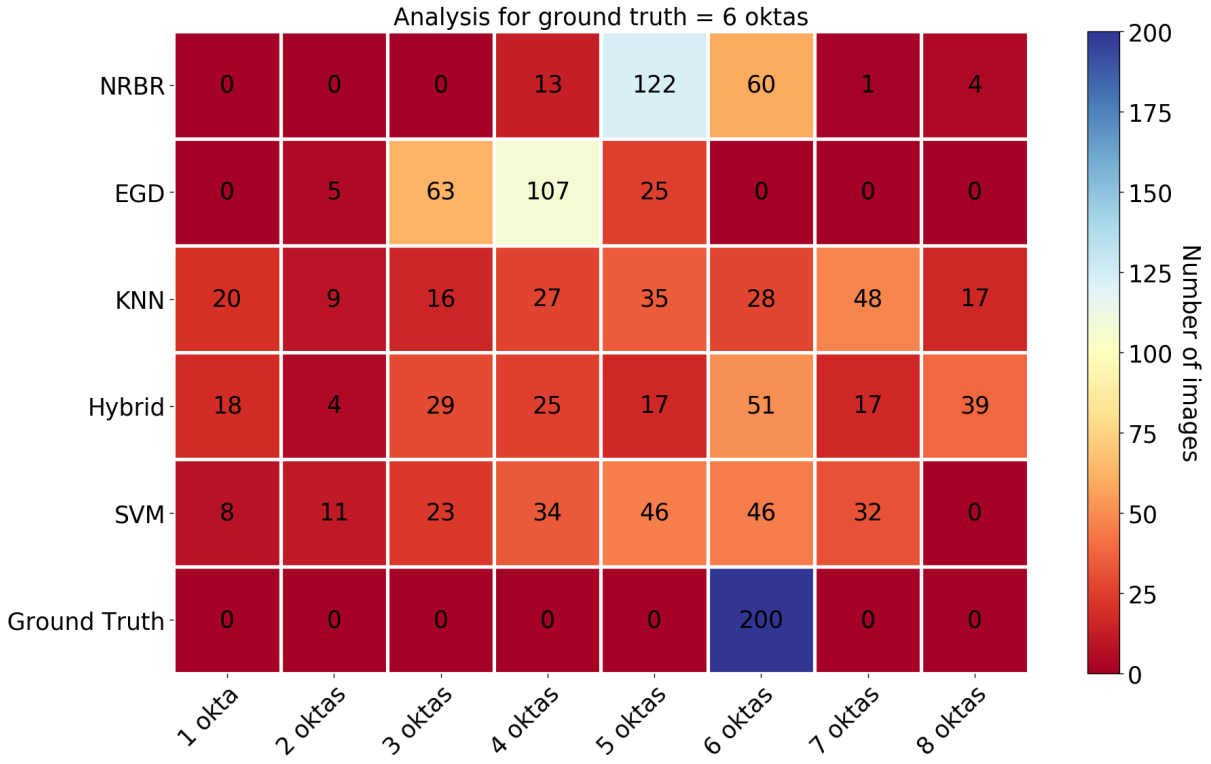


Figure 51: Heatmap for all methods where ground truth equals six oktas. There are a total of 200 images in this category.

Seven oktas ground truth

In figure 52 the image shows a thick dark cloud in the top half. In addition, there is a shiny white part of the cloud and a small part of the sky is visible. Overall, most parts of the image are covered by clouds. Only two methods analyse the true value of oktas. The SVM analysis is in good agreement with the ground truth. KNN analyses the correct number of oktas, the analysed part of the sky is in a completely different place. In this case, despite the correct okta, it is a miss. Hybrid and NRBR overestimate the cloud cover and also the analysis of the EGD method is wrong. The analysis from EGD shows a cloud in a completely different part of the image.

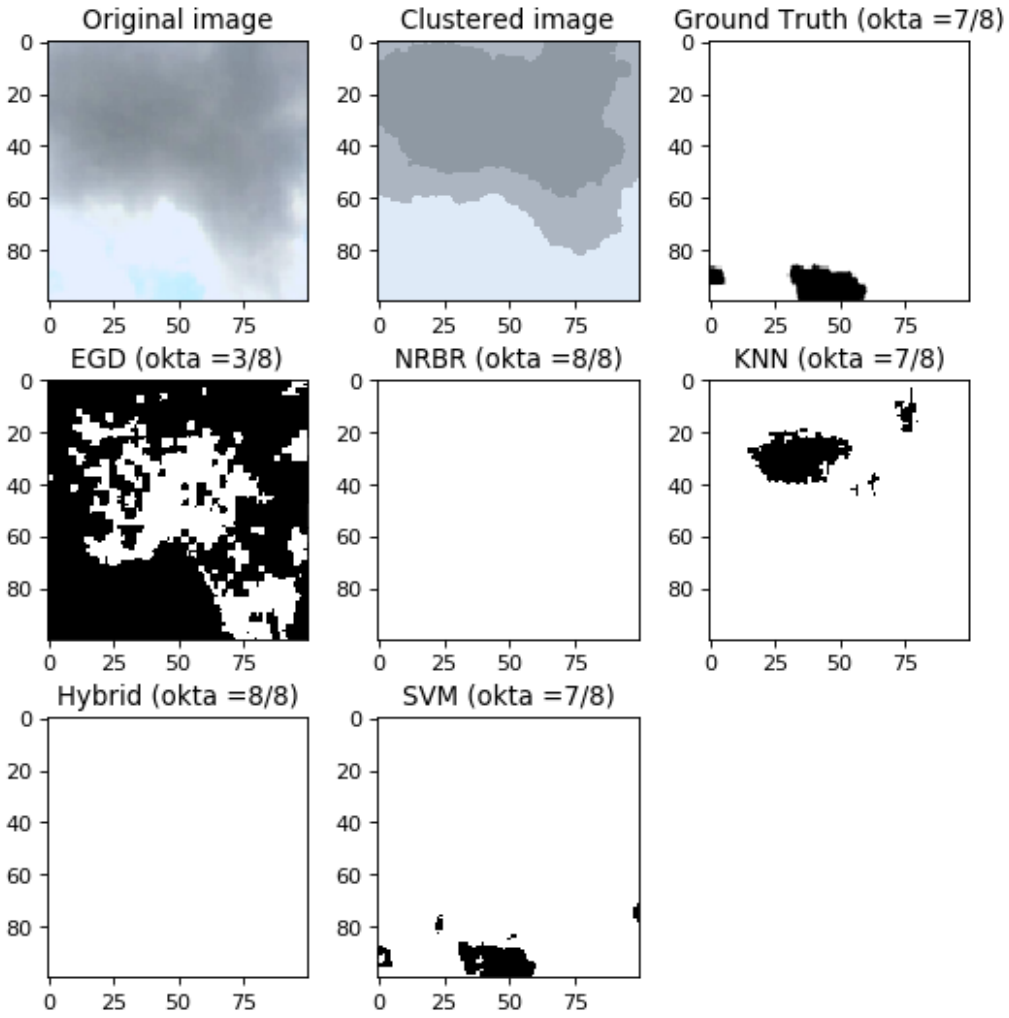


Figure 52: 1. result (GT=7): two out of five methods analyse the true value, dark cloud, high whitish intensity of sky

The next example (figure 53) shows patches of clouds with a small fraction of clear sky in the image. Particularly the right half of the image is completely covered by clouds as seen in the ground truth. Only one method analyses the right value. The analysis for the NRBR, KNN and Hybrid method results in eight oktas. Whereas EGD completely underestimates the cloud cover.

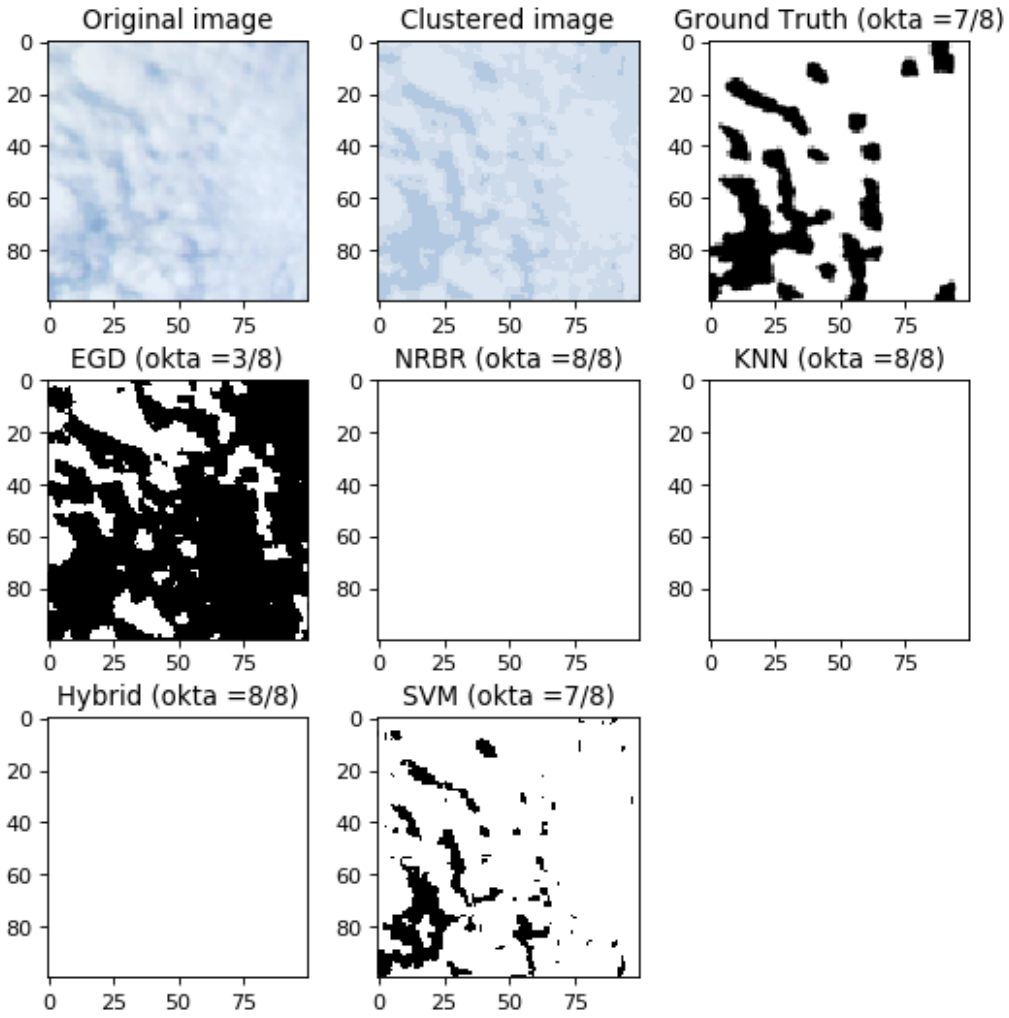


Figure 53: 2. result (GT=7): one out of five analyse the true value, a lot of clusters of white clouds, generally high intensity of white in the image

In the heatmap in figure 54 the SVM gives the best overall result. 145 images are assigned to the same category and 36 in one category lower. EGD and NRBR have quite high confidence, as most of the images are assigned to one category and variability is not as high as in the KNN and Hybrid method. These results are supported by the verification scores in table 14. SVM has the lowest values of the verification score, where EGD results the poorest verification scores.

Score	NRBR	EGD	KNN	Hybrid	SVM
MAE	0.047	0.288	0.129	0.137	0.046
RMSE	0.218	0.536	0.36	0.371	0.215
BIAS	-0.153	-0.52	-0.184	-0.18	-0.097

Table 14: Verification scores for seven oktas.

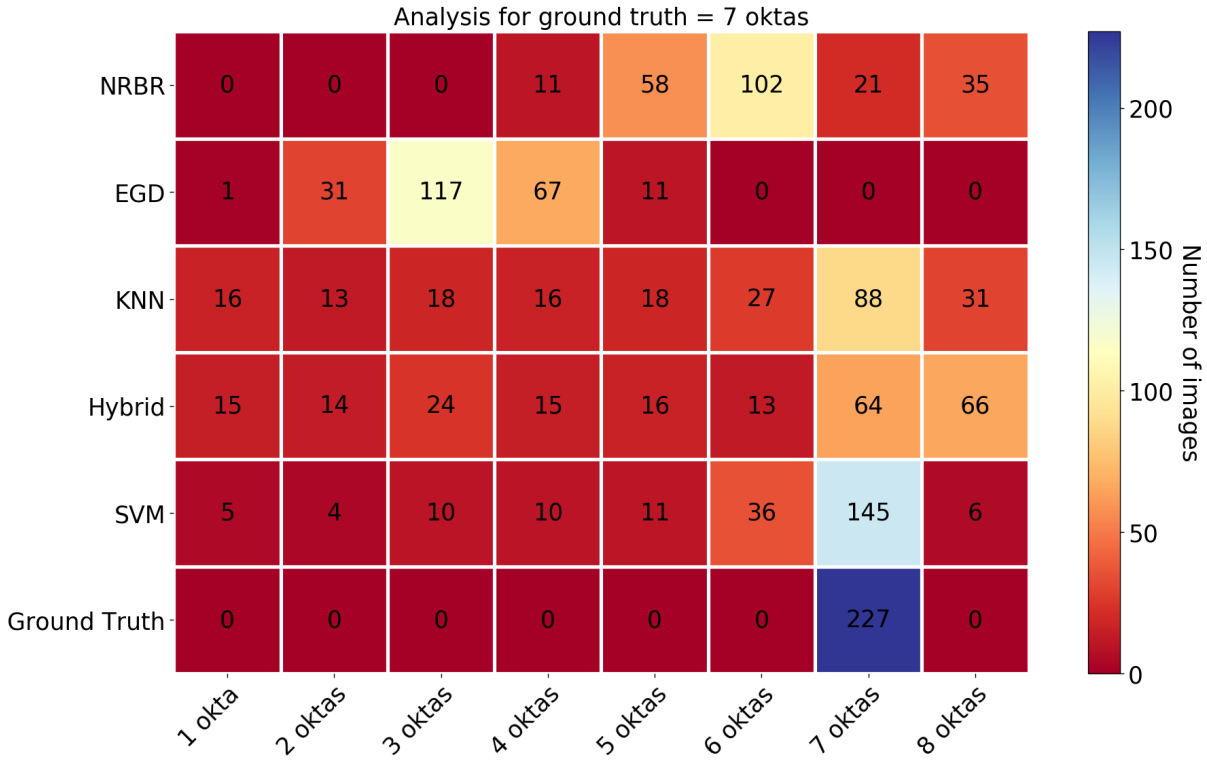


Figure 54: Heatmap for all methods where ground truth equals seven oktas. There are 227 images in total in this category.

Overall verification of cloud cover

For the analysis of all categories it is useful to combine all verification scores in one figure. The RMSE, MAE and Bias of all cloud okta categories are combined in one plot. The results can be seen in figures 55, 56 and 57.

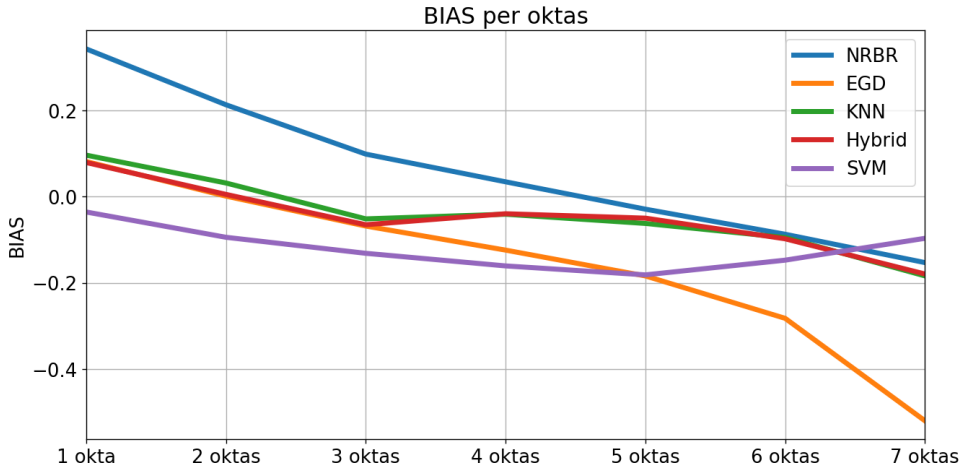


Figure 55: BIAS of all categories. BIAS is plotted on the ordinate and the respective cloud cover category on the abscissa.

In figure 55, initially the BIAS is positive for most of the methods, only the SVM underestimates the cloud cover by a little. In the further analysis the BIAS tends towards negative values for increasing cloud cover. The two threshold methods show a linear progression in the direction of the negative bias. Especially the EGD method shows a strong trend towards the negative range in the last category. The hybrid and KNN methods tend towards a bias of zero at three to four oktas. Compared to the other methods, the SVM has an opposite trend at the end and again shows a bias towards zero.

MAE and RMSE scores show (equation 4) that only the magnitude distinguishes these two verification scores. The trend in the two figures for all methods is the same. In the first category, the SVM and EGD method have low values of MAE and RMSE. Whereas the other scores and especially the NRBR show high values of RMSE. During the analysis all methods except for the NRBR have a trend towards higher RMSE and MAE. In the last category, the MAE and RMSE is increasing for all methods expect for the SVM.

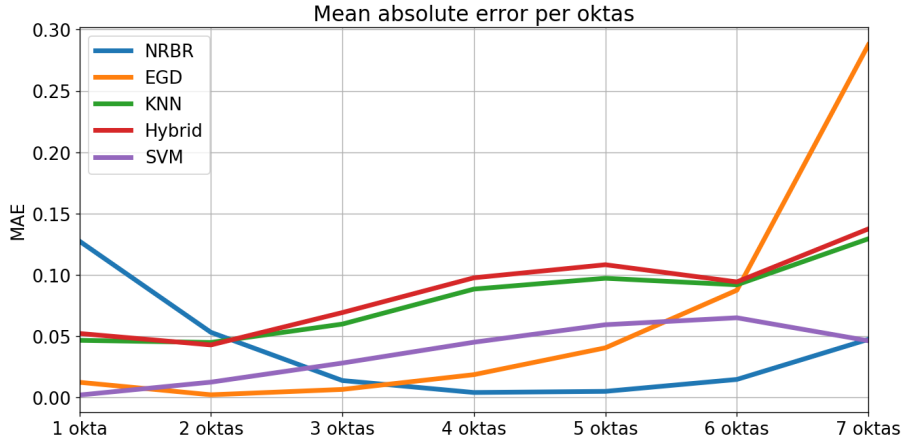


Figure 56: MAE of all categories. MAE is plotted on the ordinate and the respective cloud cover category on the abscissa.

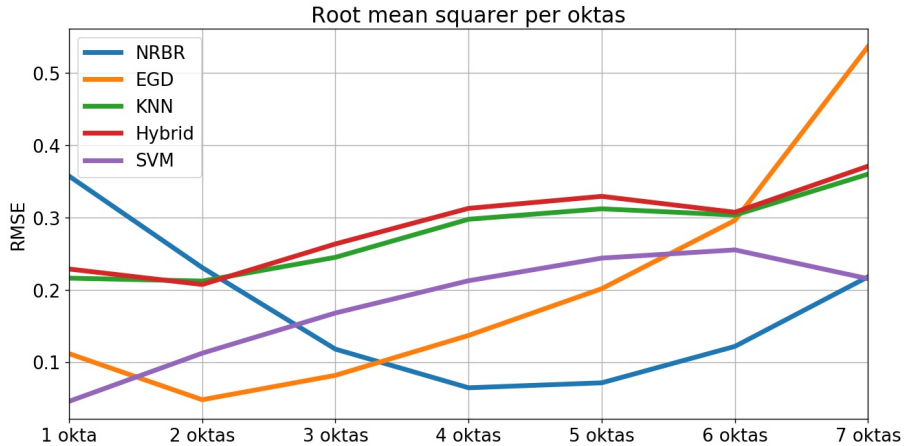


Figure 57: RMSE of all categories. RMSE is plotted on the ordinate and the respective cloud cover category on the abscissa.

6.1.2 Cloud images with landscape

The images of the SWIMSEG and SWIMCAT dataset are taken from a sky view imager. This measuring instrument is directed vertical towards the sky and therefore there is usually no piece of landscape or infrastructure in the images. But, as the basis idea is to use images taken from smartphones, it cannot be avoided that other objects than clouds and clear sky are in the images. The next few examples contain images from the CCSN database and some images which were sent from students during the H-View project. The verification of these images can only be done on a visual basis, as there is no ground truth to compare with.

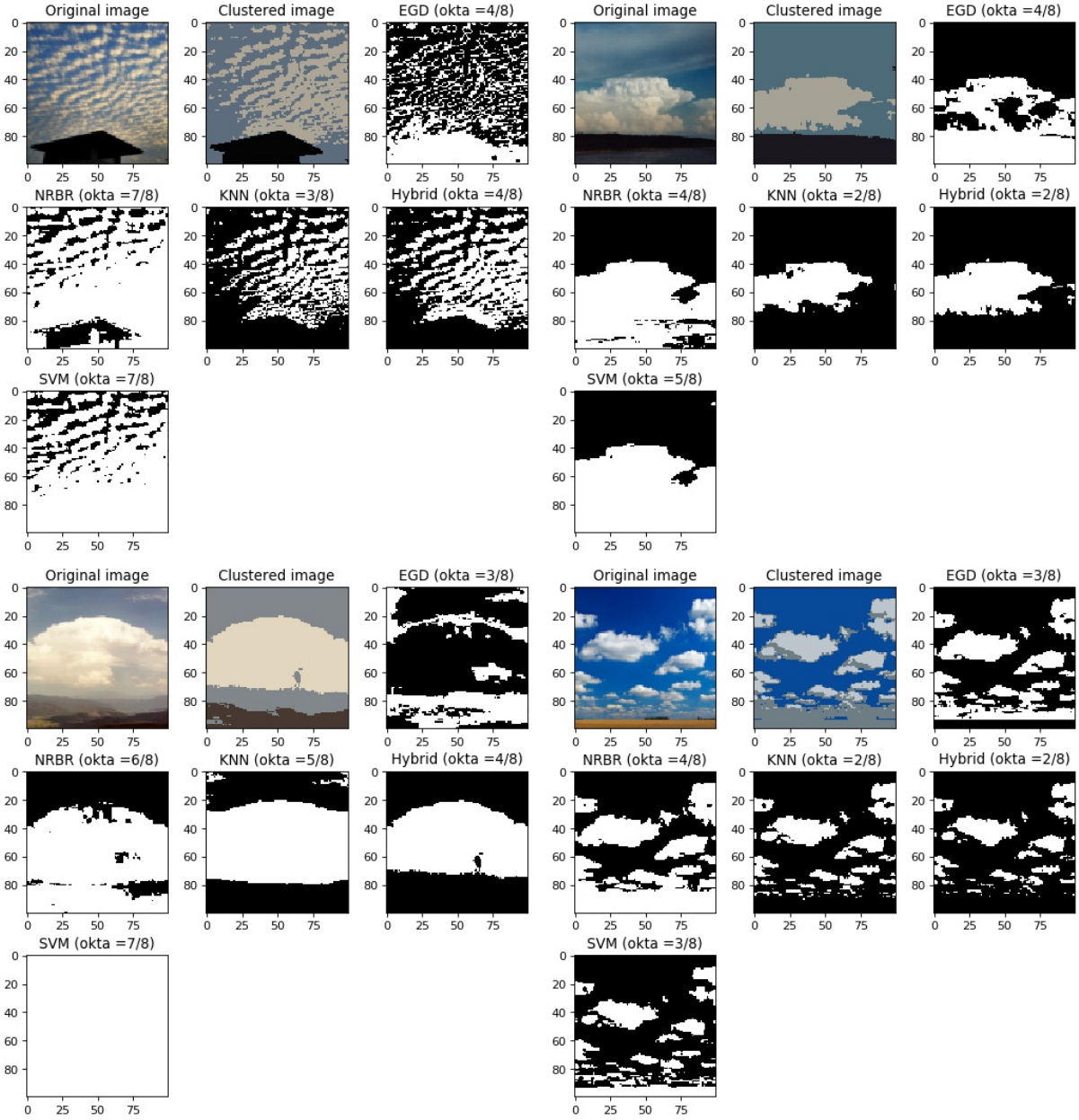


Figure 58: Images from the CCSN dataset with landscape

Four examples where landscape is included in the cloud images can be seen in figure 58. These examples are selected from the CCSN dataset. In the top left image are altocumulus

clouds and in the bottom of the image is some kind of shed. Three out of the methods determine the shed as part of the clouds, whereas the machine learning algorithms KNN and Hybrid declare it as sky (cloud-free) pixels. The structure of the clouds is captured by most of the methods.

A cumulonimbus cloud can be seen in the background of the next image on the top right corner. The bottom half of it is covered by landscape. Again the two machine learning algorithm isolate the cloud from the landscape while the other methods include it in the analysis. The same applies for the two images in the bottom half of figure 58. KNN and Hybrid detect the landscape, while the other methods do not consider the landscape or other disturbances.

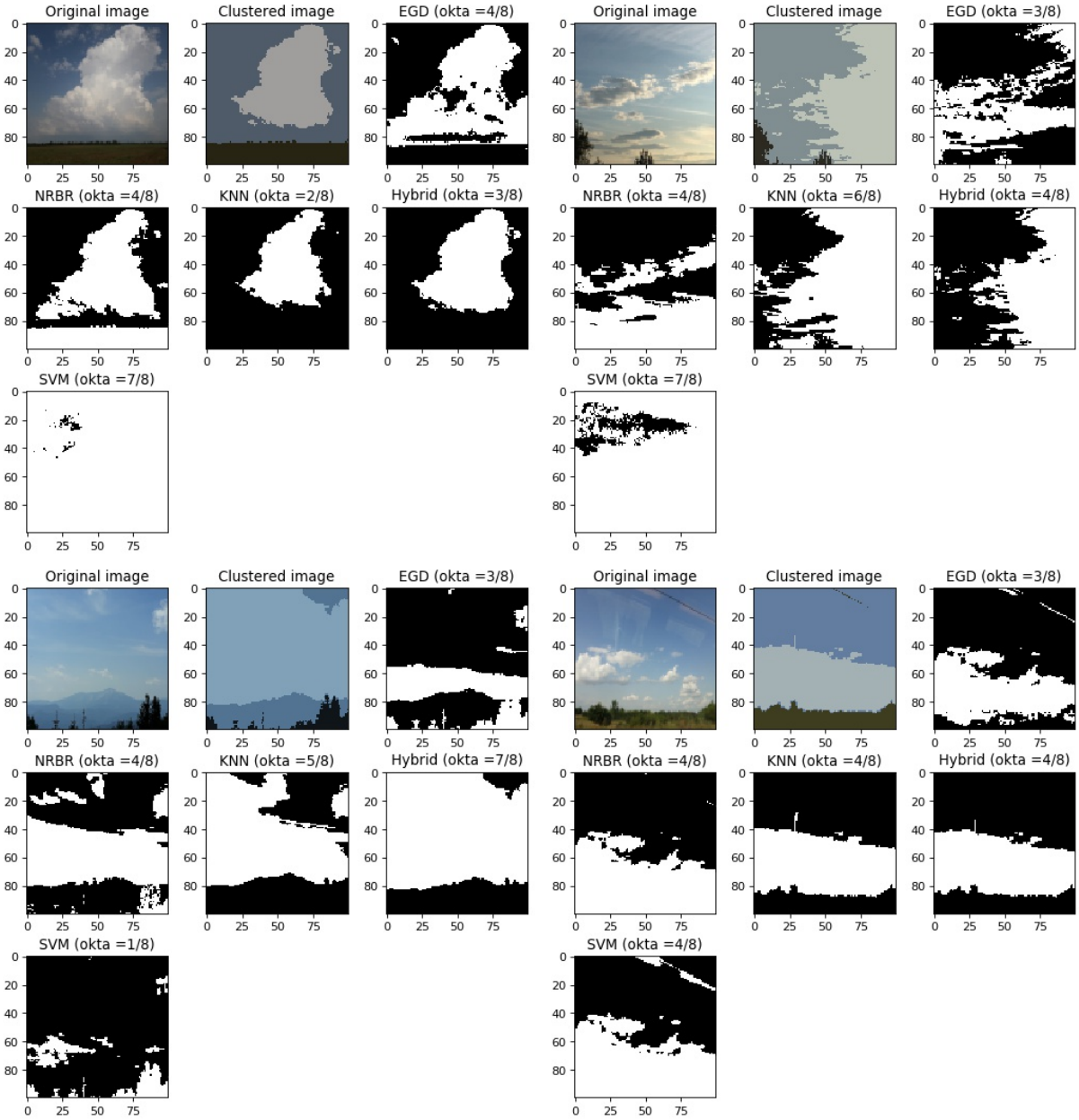


Figure 59: Images taken from participants of the H-View project

The next four examples are images taken by smartphones and were collected during the H-View project. The analysis in figure 59 shows, that again the KNN and Hybrid method isolate the landscape from the rest of the image for most parts and only analyse the clouds. The three other methods (EGD, NRBR and SVM) included the landscape in the analysis and therefore overestimates the cloud cover in most of the images.

6.2 Cloud classification

This chapter describes the results of the cloud classification of the two methods CNN and SVM. The results are described with the help of the scores, which have been defined in chapter 5.2. The architecture and the setup of the individual parameters for the description of the models can be found in chapter 3.2.4 and 3.2.5. First, the final results of the SVM are described and afterwards the result of the different CNN models. These results are then compared to the results of Dev et al. (2015) [6], Rhee et al. (2018) [21] and Zhang et al. (2018) [7]. All models are applied to the SWIMCAT and CCSN dataset.

6.2.1 SWIMCAT dataset

The SWIMCAT dataset is a small database containing five categories of different cloud types. Some example images and a detailed description can be found in chapter 4.1.1.

Cloud classification SVM models

As described in 3.2.4 the support vector machine can also solve complex classification problems. In this case the SVM is trained on a training set and then verified on the test dataset. There are many different settings which can results in a variety of different results. The results depends on the kernel of the SVM and other important variables. These results are presented in this chapter and then compared to the results of Dev et al. (2015) [6], Rhee et al. (2018) [21] and Zhang et al. (2018) [7]. The models in these three papers used the SWIMCAT dataset. In Zhang et al. (2019), the CCSN dataset was also applied to the CNN model.

The first results for the classification of the SWIMCAT dataset can be seen in figure 60. In this example, the SVM uses a linear kernel. The class A-sky, which includes only clear sky images, is classified by the SVM 100% correctly. Also the category C-thick-dark and E-veil is classified accurately by this model. In the other two categories (B-pattern,D-thick-white), the SVM shows some problems. In B-pattern 30% of the images are assigned to the category of E-veil. The assignment of the category D-thick-white is scattered towards three other categories. It is also noticeable, that the different settings of the model do not change the overall results a lot. Only the modification of the parameter C from 1 to 10 has an impact in the category D-thick-pattern. The other changes of C have no influence on the results.

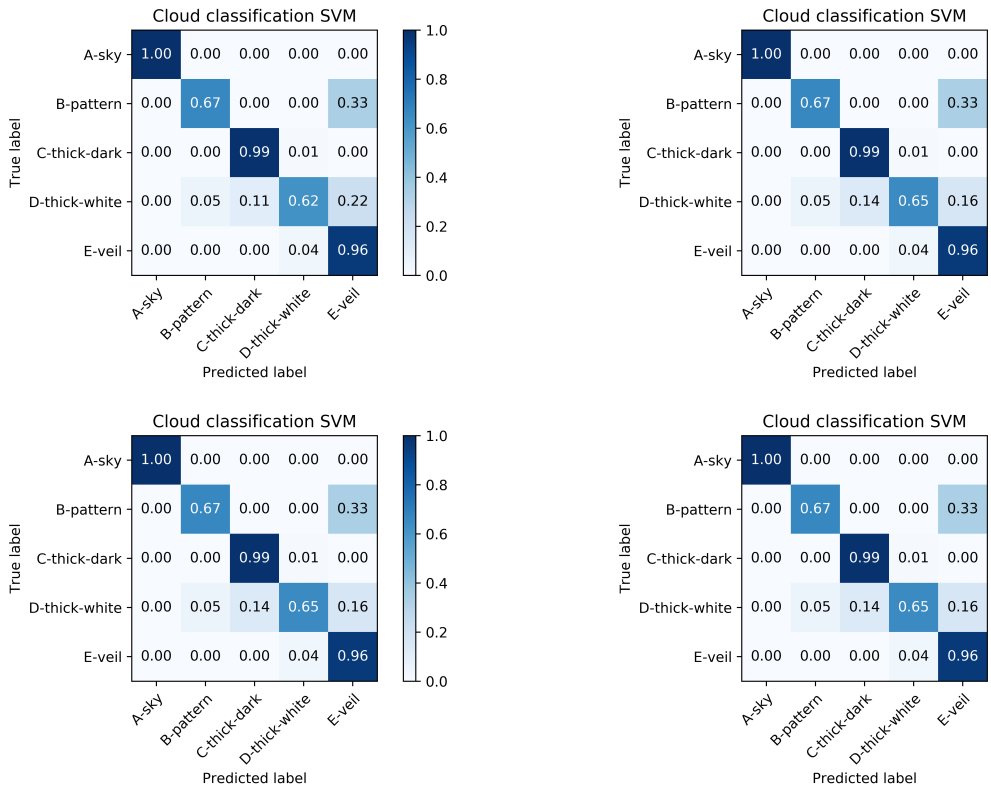


Figure 60: SVM with linear kernels; Parameter c: top to bottom, left to right: [1,10,100,1000]

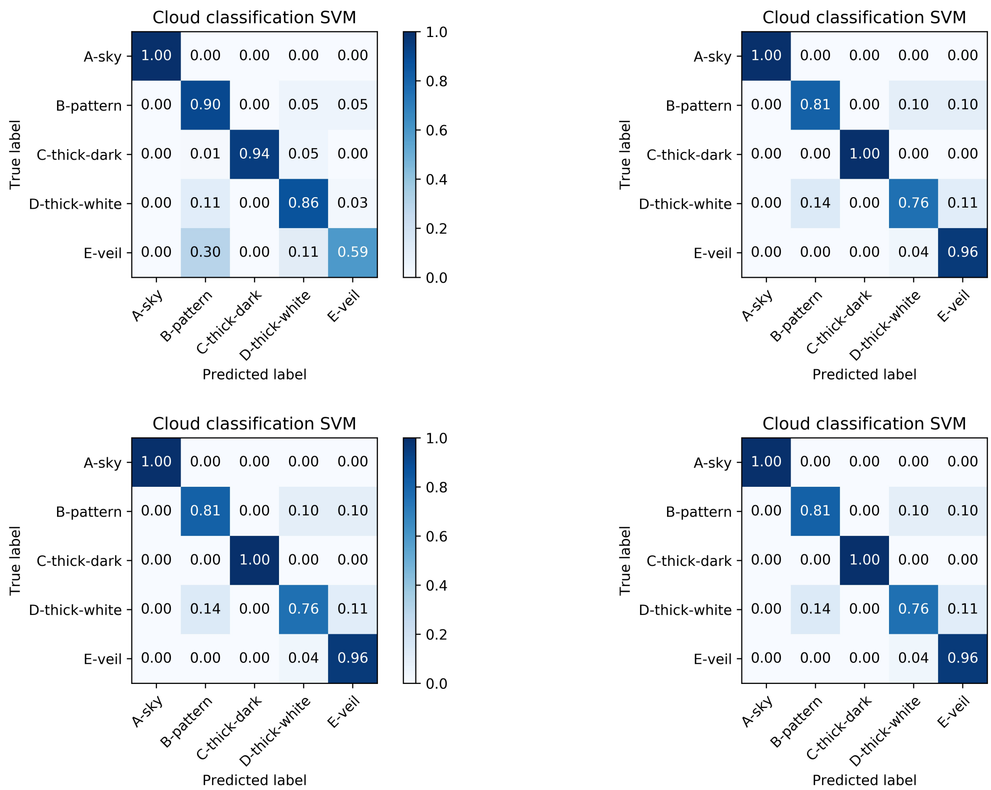


Figure 61: SVM with RBF kernels; Parameter c: top to bottom, left to right: [1,10,100,1000]; Parameter gamma: 0.01

Figure 61 shows the results from the SVM model with a radial basis function (RBF), a non-linear kernel. The parameter C varies from 1 to 1000 and the parameter gamma is set to 0.01. The models where the parameter C equals 10, 100 and 1000 assign all images in two classes out of five to the true category (100% accuracy). But, also in the category E-veil, 90% of the classifications are true. In the top left image, were C equals 1, the results from B-pattern and D-thick-white shows better percentage than in the other three models. However, in the last category E-veil only 60% of the images are assigned to the correct category.

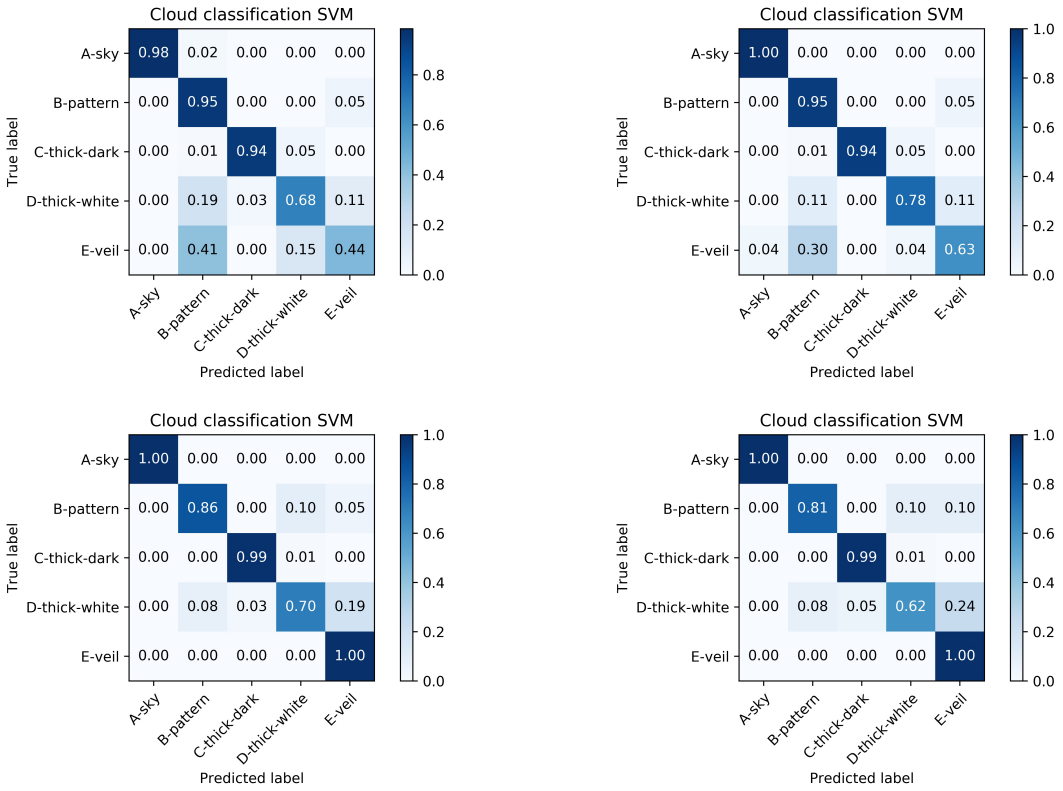


Figure 62: SVM with RBF kernels; Parameter c: top to bottom, left to right: [1,10,100,1000]; Parameter gamma: 0.001

The next model in figure 62 is also calculated with a RBF kernel and the parameter C ranges from 1 to 1000. The parameter gamma is set to 0.001. The results for this models are illustrated in figure 62. Image one and two in the top row show quite similar results. The first three categories are assigned rather accurately while the last two categories show some weaknesses. Especially in the left top model, the scattering in the last category is high. 40% of the E-veil images are classified as B-pattern images. But, also in the model in the top right corner, 30% of the images of the same category are assigned wrong. On the other hand, the two models in the bottom row of the figure, where C equals 100 and 1000, classify all images of the E-veil category correctly. In the other four categories most of the images are assigned to the correct category.

Overall models, the best results are achieved by the RBF kernel where the parameter C is above 10 and gamma equals 0.01. As this is the best result, this one will be compared to the results the other three models from the literature.

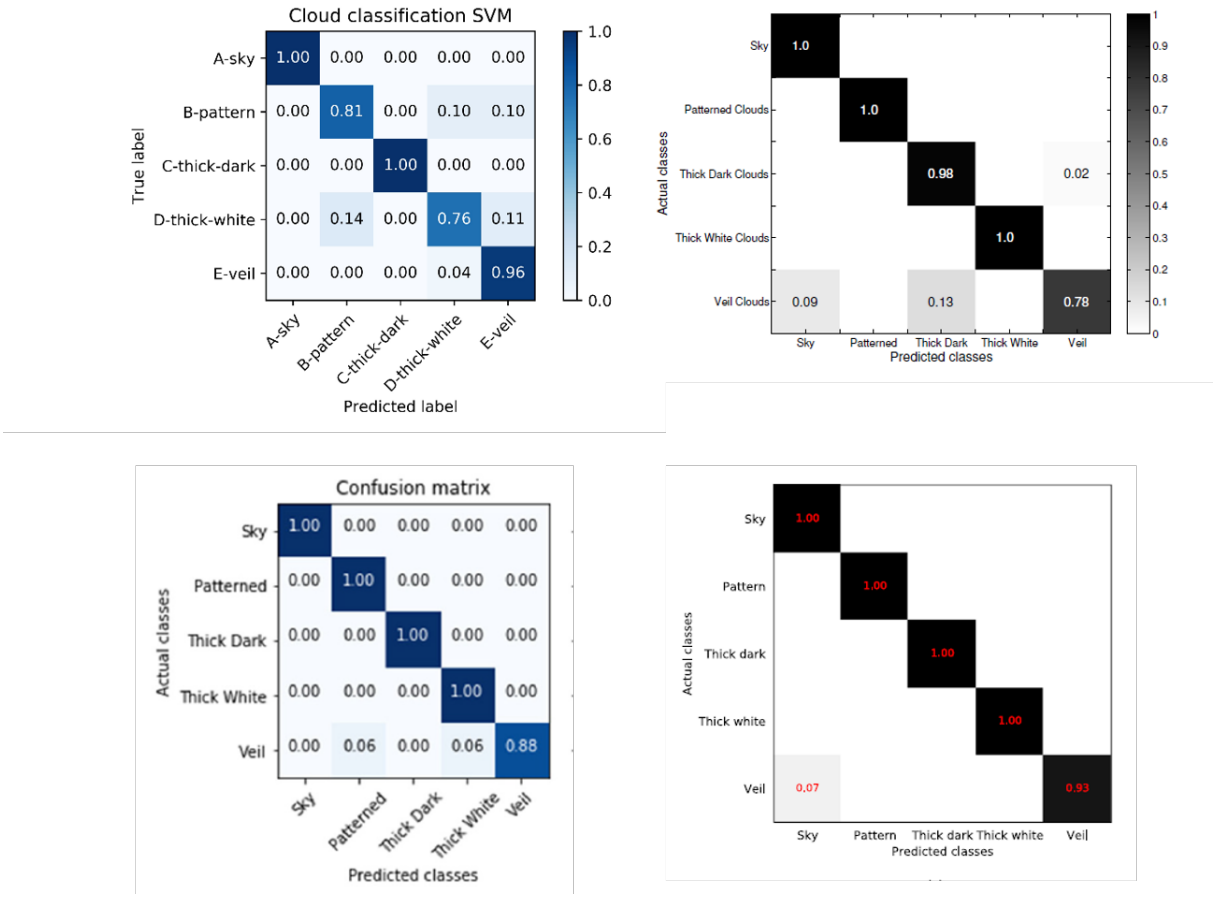


Figure 63: Top left: Results SVM (RBF Kernel, c = 10, gamma = 0.01); Top right: Results from Dev et al. (2015) [6]; Bottom left: Results from Rhee et al. (2018) [21]; Bottom right: Results from Zhang et al. (2018) [7]. The top right and bottom right illustrations are different types of confusion matrix, but the values are interpreted in the same way.

A comparison of the results from the SVM model with results from different approaches in the literature can be found in figure 63. The results in the top right matrix comes from an improved texton-based approach by Dev et al. (2015) [6]. This model classifies mostly all images in four categories correctly, only 0.02% in the category of thick-dark of the images are assigned into the wrong one. However, the model shows weakness in the categorisation of veil clouds, this error also is discovered in different settings of the SVM model used in this thesis. The other two results from the literature come from a CNN model. The first one, bottom left image, is by Rhee et al. (2018) [21]. In this model the classification of all images in four categories are 100% correct. But it also has a weakness in the last category. The same applies to the model by Zhang et al. (2018) [7]. In four categories the classification is correct, whereas in the last category 0.07% of the images are assigned wrong.

Cloud classification CNN models

In this chapter the results of the different convolutions neural network models are presented. Three different settings of the models are applied to the two datasets. The accuracy and loss as well as the classification results are shown for each model and for the different settings of the models. The architecture and the summary of the models can be looked up in chapter 3.2.5. The number of epochs for all models is 1000 and each model is calculated once with and once without data augmentation.

Model by Chollet, 2018

The results from the first CNN model are illustrated in the next few figures. The exact design of the model by Chollet, 2018 [8] can be found in chapter 3.2.5. The two confusion matrices in figure 64 represent the same type of matrix as seen in the chapters before. Both results are from the third fold out of five folds with the k-cross validation method.

The improvement of the model during the training process can be seen in the two plots of the model accuracy and loss in figure 65. Initially, the accuracy and loss of the model are high, but with time (number of epochs) the loss decreases while the accuracy increases quite rapidly. Although there are a few small outliers, the accuracy remains high and the error small until the end. The discrepancy between validation loss and train loss is a typical sign of a slight overfitting of the data.

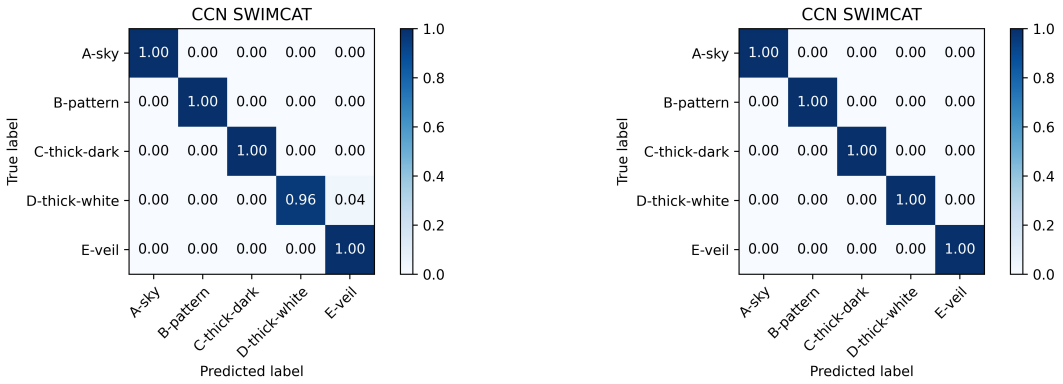


Figure 64: Confusion matrix of the model by Chollet, 2018 with SWIMCAT dataset. Left: model without data augmentation. Right: Model with data augmentation.

The results for each category is illustrated in the two matrices in figure 64. In both model runs the classification is almost perfectly correct. Especially the model where data augmentation is applied shows perfect scores, as all images are assigned to the correct category. Only the model without data augmentation misclassifies 0.04% of the images in the category D (thick white).

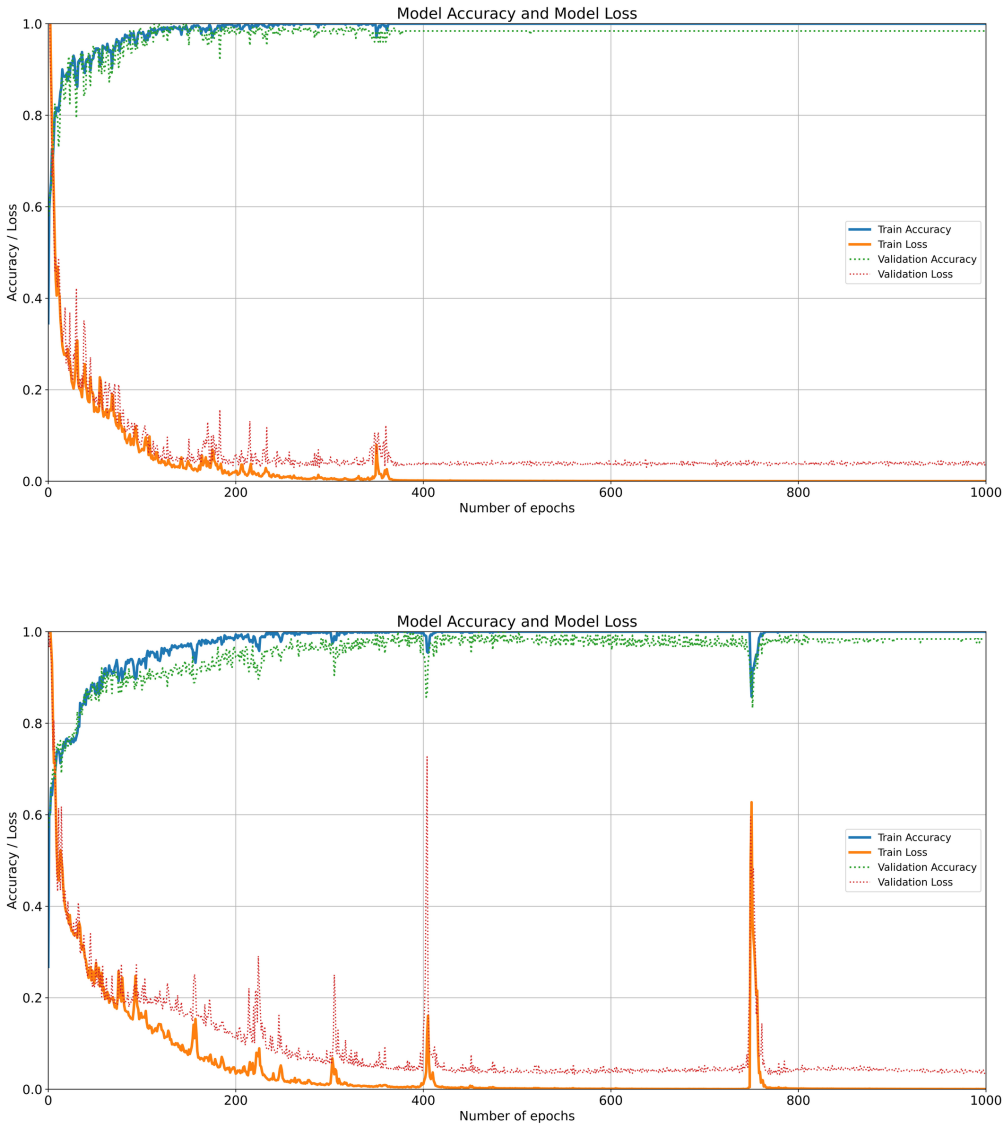


Figure 65: Cost and loss function of the model by Chollet, 2018. The solid/dotted lines show the accuracy and loss while training/validation. Top: model without data augmentation. Bottom: model with data augmentation (rotation, zoom, horizontal/vertical flip, etc.).

The results over all five folds for each model are shown in figure 66. The model without data augmentation has an overall accuracy of 97% and a loss of under 0.2%. Also the accuracy of the model with data augmentation reaches over 90% and the loss is around 2%. Here, however, it is noticeable that the first run of the model had a significantly worse performance. On the other hand, fold three shows 100% accuracy of the model. This fold is also the one, where all images are assigned to the correct category.

```
------------------------------------------------------------------------
Score per fold
------------------------------------------------------------------------
> Fold 1 - Loss: 0.20949824154376984 - Accuracy: 95.54139971733093%
------------------------------------------------------------------------
> Fold 2 - Loss: 0.3244226574897766 - Accuracy: 94.90445852279663%
------------------------------------------------------------------------
> Fold 3 - Loss: 0.03177828714251518 - Accuracy: 99.3630588054657%
------------------------------------------------------------------------
> Fold 4 - Loss: 0.03059222549200058 - Accuracy: 98.08917045593262%
------------------------------------------------------------------------
> Fold 5 - Loss: 0.02381971850991249 - Accuracy: 98.07692170143127%
------------------------------------------------------------------------
Average scores for all folds:
> Accuracy: 97.19500184059143 (+- 1.6887142937330046)
> Loss: 0.12402222603559494
------------------------------------------------------------------------
```

```
Score per fold
------------------------------------------------------------------------
> Fold 1 - Loss: 0.6268424987792969 - Accuracy: 68.15286874771118%
------------------------------------------------------------------------
> Fold 2 - Loss: 0.08502843230962753 - Accuracy: 99.3630588054657%
------------------------------------------------------------------------
> Fold 3 - Loss: 0.008785702288150787 - Accuracy: 100.0%
------------------------------------------------------------------------
> Fold 4 - Loss: 0.16843369603157043 - Accuracy: 96.81528806686401%
------------------------------------------------------------------------
> Fold 5 - Loss: 0.16954822838306427 - Accuracy: 96.79487347602844%
------------------------------------------------------------------------
Average scores for all folds:
> Accuracy: 92.22521781921387 (+- 12.10639912958583)
> Loss: 0.21172771155834197
------------------------------------------------------------------------
```

Figure 66: Results for all folds. Left: no data augmentation; Right: with data augmentation.

Model by Rhee et al. (2018)

A detailed description of the model by Rhee et al. (2018) [21] can be found in chapter 3.2.5. Fold five and fold two achieve the best results in the two models, so these two folds are chosen for figure 68 and 67.

Again, the model classifies all the data accurately with an overall accuracy of more than 90%. This fact is shown in figure 69 in the two different setups. The accuracy of the runs is also confirmed by the examples in the confusion matrices (figure 67). In both models, the learning curve has a progression towards high accuracy and low loss. Especially in the model without data augmentation the accuracy and loss functions approach the line of 1 and 0, respectively, rather quickly. In the other model, however, we see that the learning curve is quite high at the beginning, but then stagnates for some time from epoch 30 to 300. From epoch 400 the loss and accuracy decreases and increases more significant again and the two functions approach 0 and 1, respectively. The two confusion matrices show, that the results from the model with data augmentation are slightly better compared to the the one without it. The accuracy over all folds however is higher in the model without data augmentation.

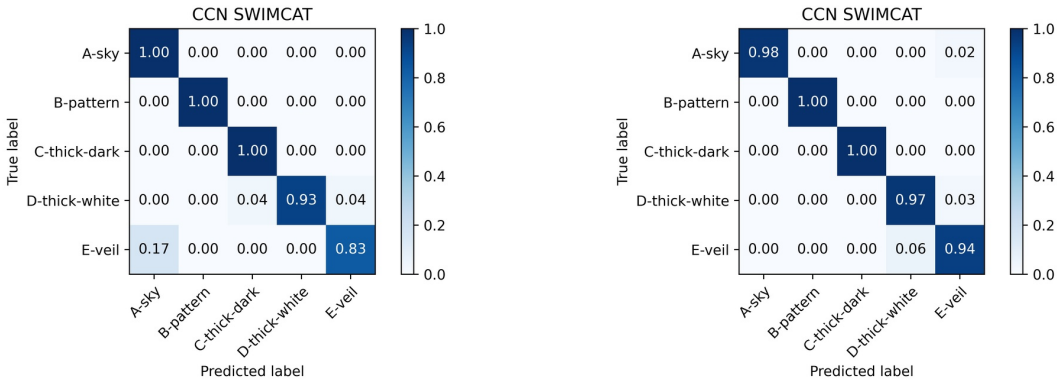


Figure 67: Confusion matrix of the model by Rhee et al. (2018) with SWIMCAT dataset. Left: model without data augmentation; Right: Model with data augmentation.

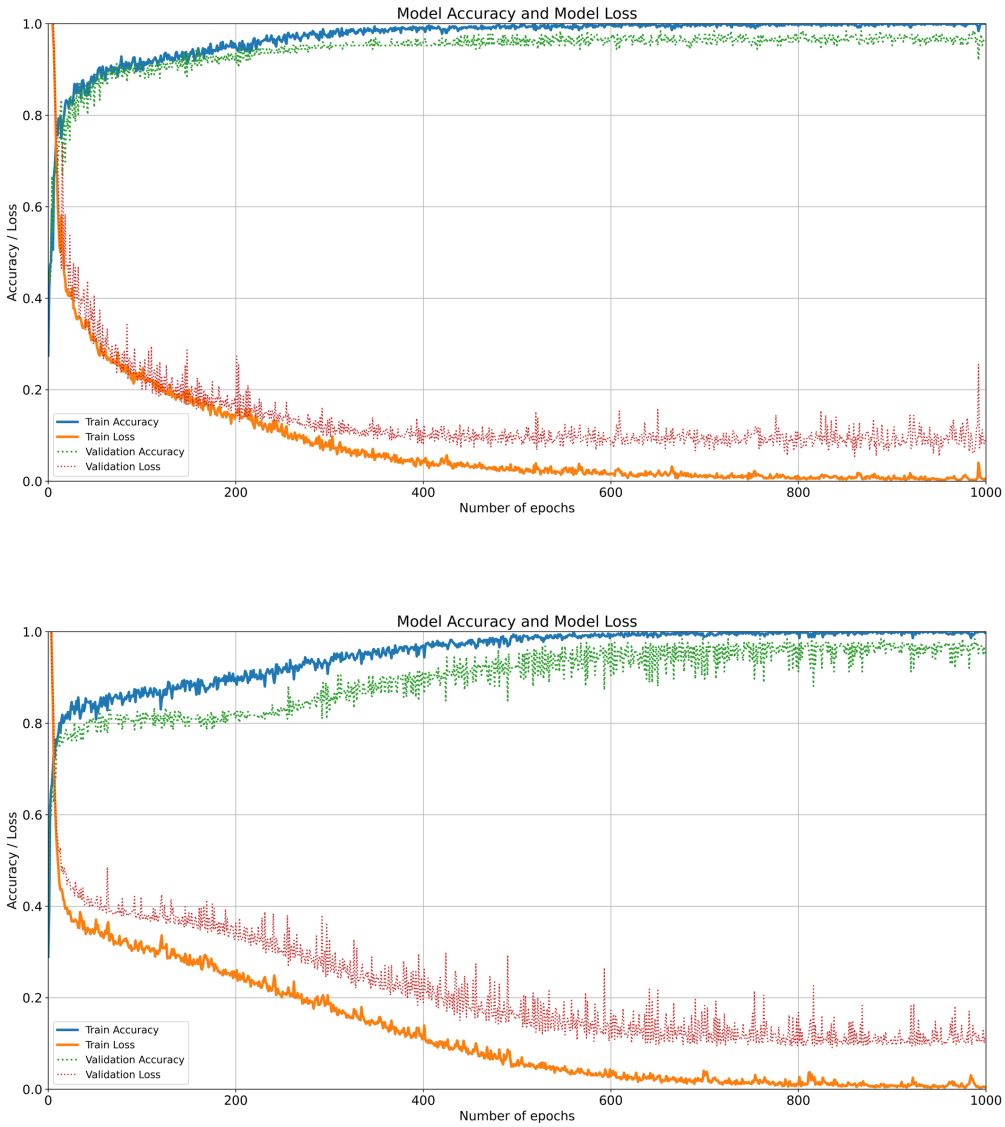


Figure 68: Functions of loss and accuracy of model by Rhee et al. (2018.The solid/dotted lines show the accuracy and loss while training/validation. Top: model without data augmentation. Bottom: model with data augmentation (rotation, zoom, horizontal/vertical flip, etc.).

```
Score per fold
------------------------------------------------------------------------
> Fold 1 - Loss: 0.2818168103694916 - Accuracy: 96.17834687232971%
------------------------------------------------------------------------
> Fold 2 - Loss: 0.14046505093574524 - Accuracy: 96.17834687232971%
------------------------------------------------------------------------
> Fold 3 - Loss: 0.1598242074251175 - Accuracy: 96.81528806686401%
------------------------------------------------------------------------
> Fold 4 - Loss: 0.09510541707277298 - Accuracy: 95.54139971733093%
------------------------------------------------------------------------
> Fold 5 - Loss: 0.10327859967947006 - Accuracy: 96.79487347602844%
------------------------------------------------------------------------
Average scores for all folds:
> Accuracy: 96.30165100097656 (+- 0.4723306629718884)
> Loss: 0.15609801709651946
------------------------------------------------------------------------
```

```
Score per fold
------------------------------------------------------------------------
> Fold 1 - Loss: 0.09790670871734619 - Accuracy: 96.81528806686401%
------------------------------------------------------------------------
> Fold 2 - Loss: 0.187403604388237 - Accuracy: 98.08917045593262%
------------------------------------------------------------------------
> Fold 3 - Loss: 0.3310513198375702 - Accuracy: 92.99362897872925%
------------------------------------------------------------------------
> Fold 4 - Loss: 0.2322351038455963 - Accuracy: 92.35668778419495%
------------------------------------------------------------------------
> Fold 5 - Loss: 0.13425134122371674 - Accuracy: 96.7948734760284%
------------------------------------------------------------------------
Average scores for all folds:
> Accuracy: 95.40992975234985 (+- 2.2905120738545537)
> Loss: 0.19656961560249328
------------------------------------------------------------------------
```

Figure 69: Results for all folds; left side: no data augmentation, right side: with data augmentation

Model by Zhang et al. (2018)

The last model which is tested on the dataset is by Zhang et al. (2018) named CloudNet from the authors. The architecture is by far the most complex compared to the other models. It can be looked up again in chapter 3.2.5. The graphs and matrices in figure 70 and 72 are both from the fifth fold of the models.

The confusion matrix in figure 70 on the left side shows a perfect score for the model without data augmentation, all test images are assigned to the right category. In the overall score (figure 71 becomes evident, that 100% is also achieved by the first fold. Although the learning curve in both models in figure 72) shows some outliers, the accuracy and loss functions approach the thresholds of 0 and 1 in the end. But, especially the learning curve in the bottom model shows strong signals of an overfit of the data, as the gap between the validation and training loss increases with time.

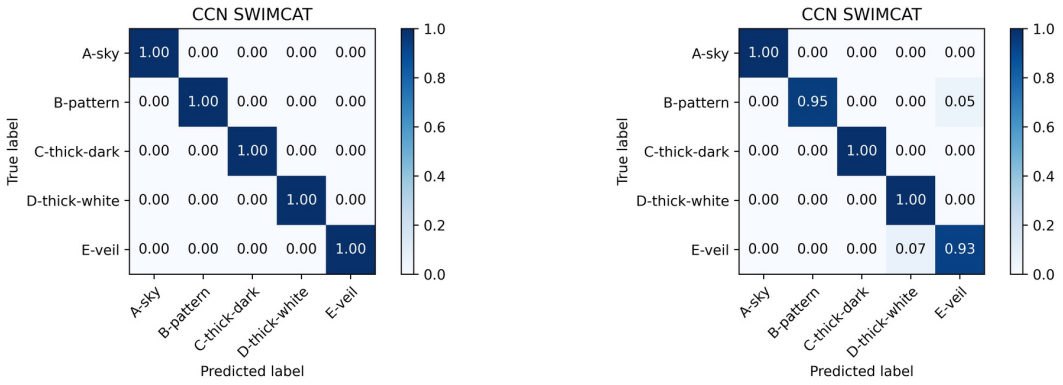


Figure 70: Confusion matrix of the model Zhang et al. (2018) with SWIMCAT dataset. Left: model without data augmentation; Right: Model with data augmentation.

```
------------------------------------------------------------------------
Score per fold
------------------------------------------------------------------------
> Fold 1 - Loss: 3.6980840377509594e-05 - Accuracy: 100.0%
------------------------------------------------------------------------
> Fold 2 - Loss: 0.15673014521598816 - Accuracy: 96.17834687232971%
------------------------------------------------------------------------
> Fold 3 - Loss: 0.17395693063735962 - Accuracy: 98.7261176109314%
------------------------------------------------------------------------
> Fold 4 - Loss: 0.09054741263389587 - Accuracy: 97.45222926139832%
------------------------------------------------------------------------
> Fold 5 - Loss: 0.003677325788885355 - Accuracy: 100.0%
------------------------------------------------------------------------
Average scores for all folds:
> Accuracy: 98.47133874893188 (+- 1.4855920438196029)
> Loss: 0.0849897590233013
------------------------------------------------------------------------
```

```
Score per fold
------------------------------------------------------------------------
> Fold 1 - Loss: 0.24889369308948517 - Accuracy: 96.17834687232971%
------------------------------------------------------------------------
> Fold 2 - Loss: 0.3669222295284271 - Accuracy: 91.08280539512634%
------------------------------------------------------------------------
> Fold 3 - Loss: 0.2015610784292221 - Accuracy: 93.63057613372803%
------------------------------------------------------------------------
> Fold 4 - Loss: 0.13520243763923645 - Accuracy: 98.08917045593262%
------------------------------------------------------------------------
> Fold 5 - Loss: 0.07015471905469894 - Accuracy: 98.71794581413269%
------------------------------------------------------------------------
Average scores for all folds:
> Accuracy: 95.53976893424988 (+- 2.846667501490937)
> Loss: 0.20454683154821396
------------------------------------------------------------------------
```

Figure 71: Results for all folds; left side: no data augmentation, right side: with data augmentation

The comparison with the three confusion matrices from the literature back in figure 63 shows, that all three models achieve the same or even better results. A perfect score, like in figure 69, for all categories cannot be found in the literature for the SWIMCAT dataset.

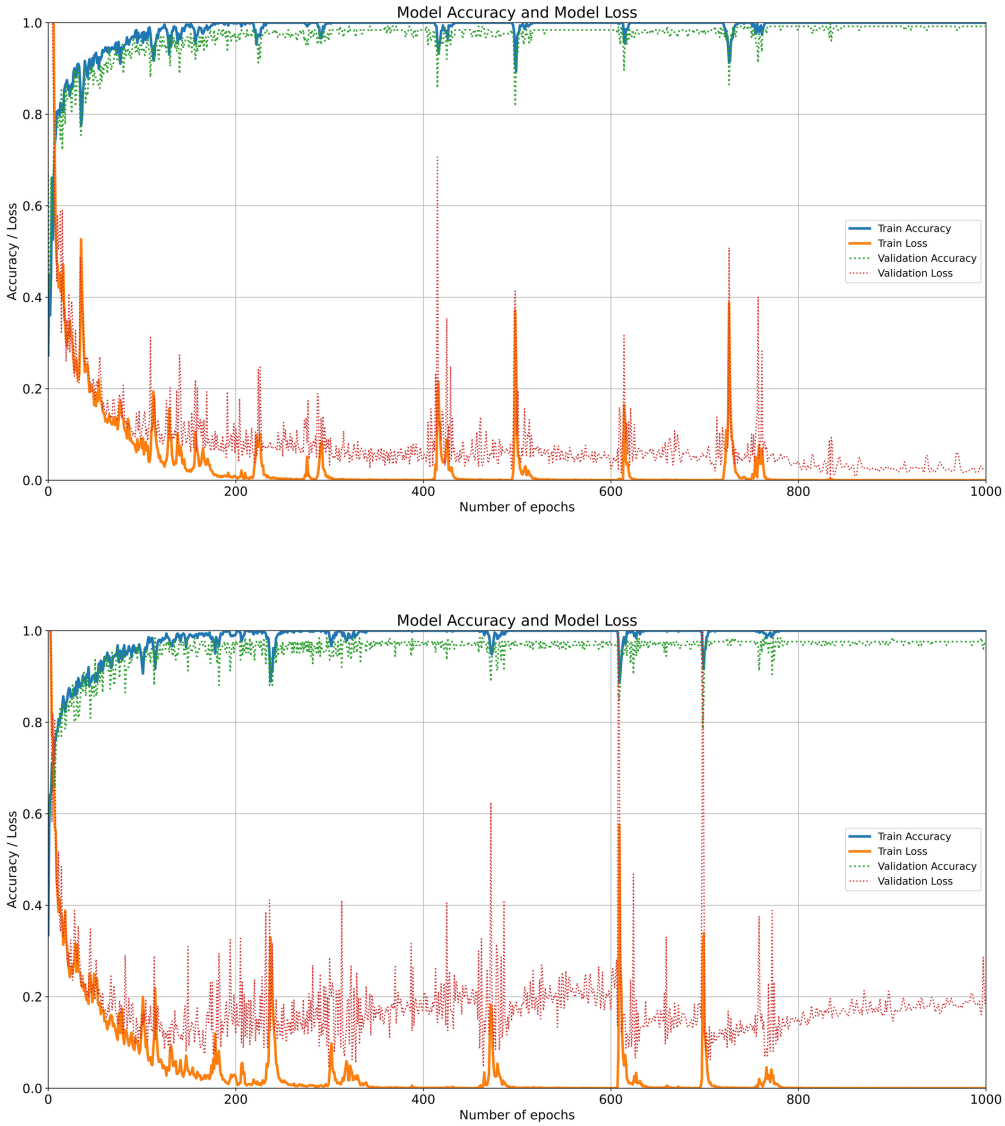


Figure 72: Functions of loss and accuracy of model by Zhang et al. (2018). The solid/dotted lines show the accuracy and loss while training/validation. Top: model without data augmentation. Bottom: model with data augmentation (rotation, zoom, horizontal/vertical flip, etc.).

6.2.2 CCSN dataset

The SVM and the three different CNN models CNN are also applied on the CCSN dataset.

The learning curves for the three models are illustrated in figures 73, 74 and 75. A first look at the three curves already shows, that the functions of loss and accuracy do not follow the expected learning curve for the significantly more complex dataset. While the loss and accuracy functions in the two figures in the top row decrease and increase respectively during training, is this not the case during validation. At a certain point during the training of the model, the loss function of the validation increases sharply, and the validation accuracy stagnates. This is a sign that the learned patterns from the model cannot be applied on the validation dataset. The graph of the model from Zhang et al. (2018) [7] does not show a trend at all. The four functions are already stagnating from the beginning. This indicates that the gradient method cannot minimise its cost function from the beginning and a learning plateau has been reached. Therefore, the model does not learn during the training process at all.

It should also be mentioned that the three models were automatically terminated after a certain time because there was no learning success. Therefore, the number of epochs varies.

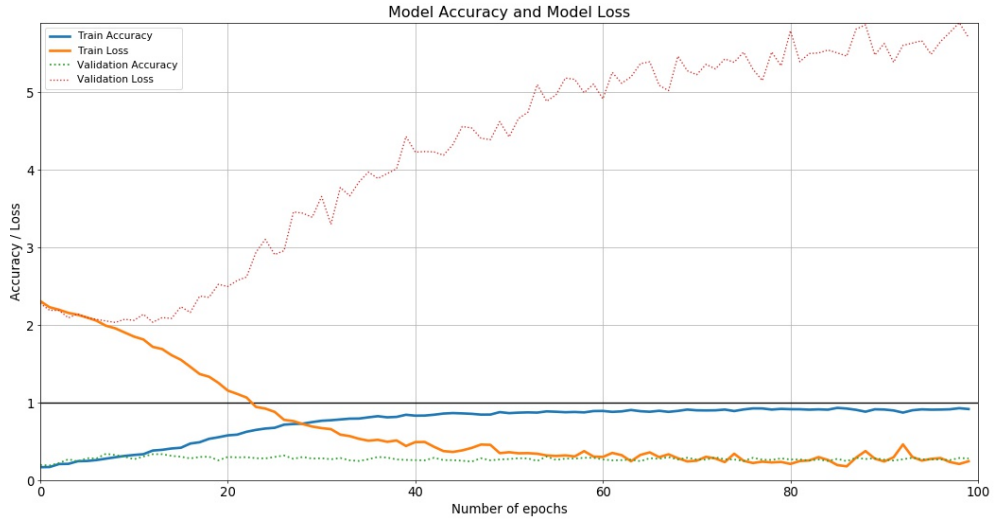


Figure 73: Accuracy and loss function of model by Chollet, 2018.

The confusion matrices for all four models are illustrated in figure 76. The results from the three CNN models are consistent with the results from the learning curves of the models. None of the three models could roughly solve the classification problem. In particular, the model of Zhang et al. (2018) does not even show scattering over other classes. Only in the confusion matrix of the SVM the diagonal can be guessed at least a little.

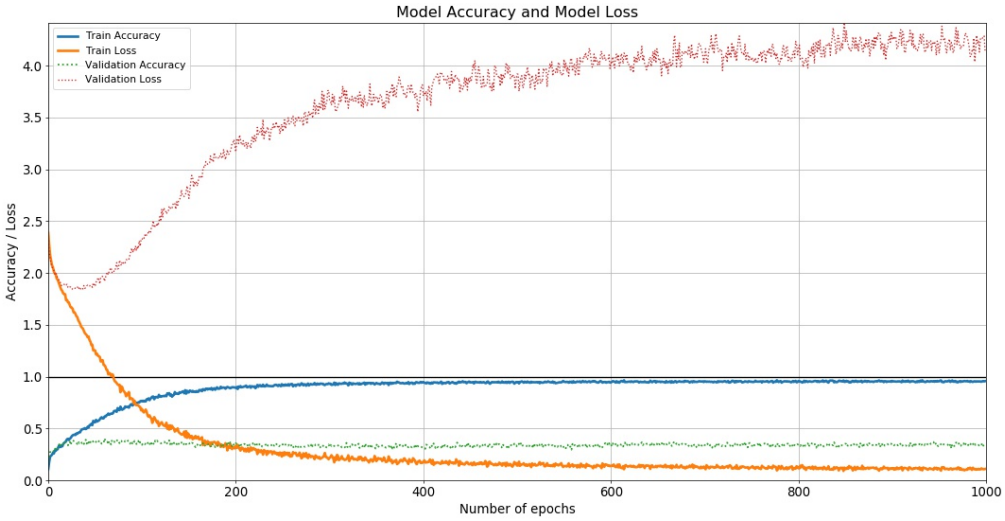


Figure 74: Accuracy and loss function of model by Rhee et al. (2018).

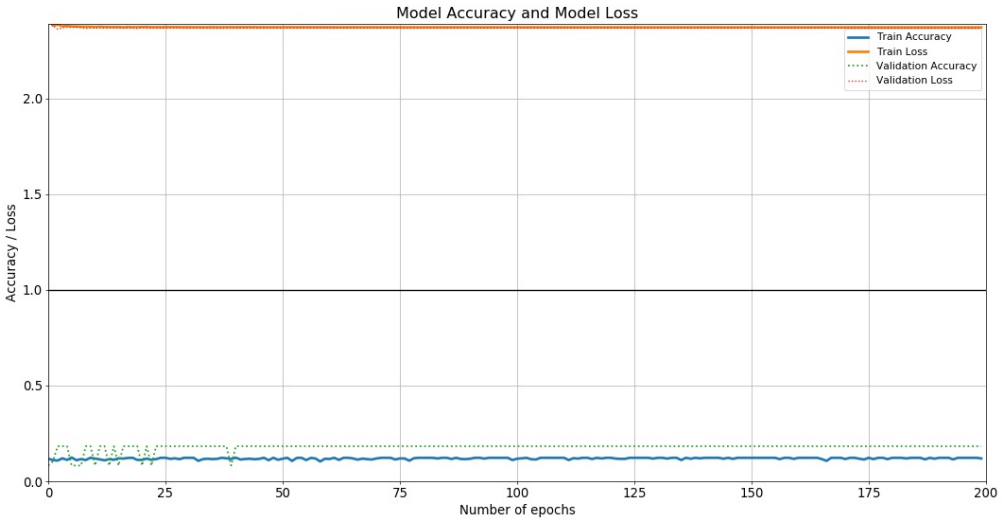
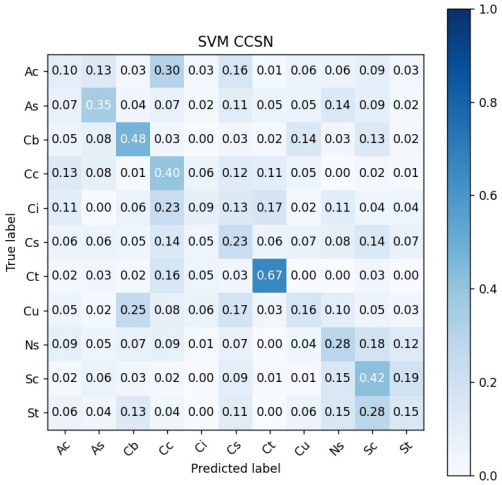
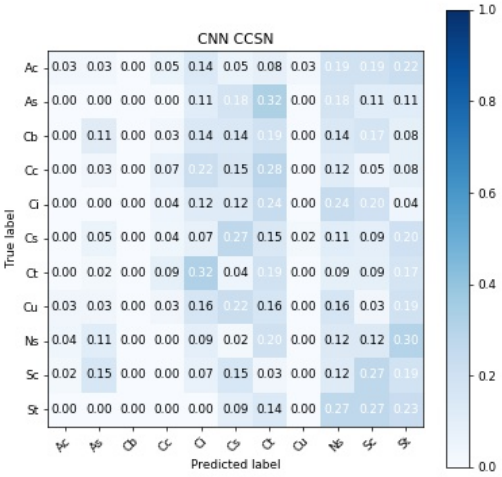


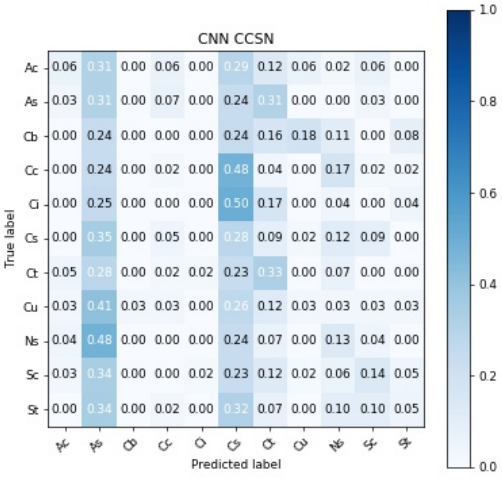
Figure 75: Accuracy and loss function for model by Zhang et al. (2018)



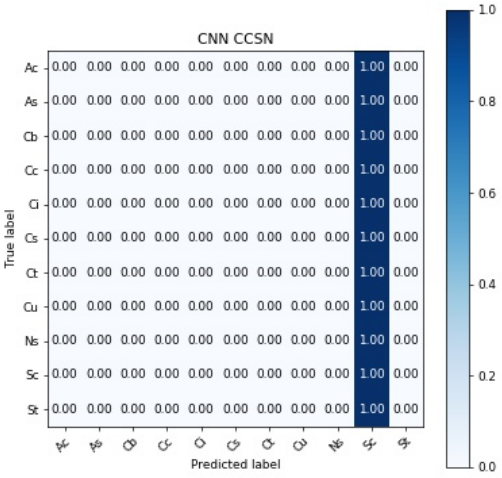
Model: SVM (RBF Kernel)



Model: Chollet, 2018



Model: Rhee et al. (2018)



Model: Zhang et al. (2018)

Figure 76: Confusion matrices for the four models with CCSN dataset

7. Conclusion

In the preceding thesis, the problems with automatic and manual weather stations and observer were described first. As stated before, the number of manual weather stations and therefore human observers is decreasing. To overcome this problem, the idea was developed to determine whether it is possible to replace this kind of observations. The automatic weather stations measure temperature, wind, moisture, and many other parameters. Also, the cloud cover can be determined by some modern techniques, but when it comes to cloud classifications, a human observer is always needed. Hence, the fundamental question was whether these manual observations could be automated using current methods of science.

7.1 Problem: cloud cover

The first problem is the cloud cover. There are several ways to approach this problem. On the one hand, threshold methods are used and on the other hand, methods from the machine learning area are applied.

The basic idea of the threshold methods is quite simple. All pixels of the image are analysed, and a threshold value is generated based on the distribution of the spectrum, which is then used to classify the different pixels. In the results section, it was shown that for some types of images these two methods achieved very good results. Especially when the contrast of clouds and clear sky was rather high, the NRBR and EGD methods classified most of the images correctly. With respect to the individual degrees of coverage, NRBR showed high agreement where the ground truth equals two oktas and EGD at three and four oktas. This is also reflecting in the MAE and RMSE across all categories. But a closer look at the other images that were analysed shows that these are mainly random hits. If we look at the heatmaps of the other categories, the EGD analysis always accumulate in the categories of three and four oktas. These false analyses are probably made because the colour grey is not recognised by the algorithm as white or as a cloud. Thus, this algorithm underestimates the degree of cloud cover, especially in a sky with many dark clouds. In the categories with a lower degree of cloud cover, the two algorithms perform better, since the images here mostly show white clouds with clear blue sky and thus have a good contrast. An overall problem, also stated by Tulpan et al. (2017) [24], is the area near the sun or generally the pollution of bright sun light in the images. In the analysis, it must also be considered that the determination of the threshold for the NRBR has been simplified in contrast to the paper of Li et al. (2011) [2].

The KNN and Hybrid methods determine the colour of the pixel with a list of the RGB values of each colours. The SVM however, learns the colours from the SWIMSEG dataset. This learning process is the great advantage of this technique with images containing only clouds.

In the results for the SWIMSEG dataset, the KNN and Hybrid methods perform relatively poor. The overall MAE and RMSE (figure 56 and 57 indicate, that these two method perform better for images with low cloud cover. This reflects in an increase in the error (MAE and RMSE) towards higher number of oktas. The BIAS shifts from positive values towards negative ones. This is because the outliers in the lower categories are only possible upwards (towards higher oktas) and in the upper categories only downwards. The main problem of these two methods can be identified in the categorisation of the colours. An example of the colour list is given in figure 17. The spectrum of this list covers eight colours, the colour grey is excluded. Several test runs have shown that if grey is in the list of colours, most pixels will be assigned to this category. The RGB values of the colour grey always have the same proportion of each colour, for example [125,125,125] or [90,90,90]. As the KNN method picks the colour in the list in a three dimensional room (RGB cube) which is next to the pixel values, grey is in most cases the obvious pick. The solution to this problem would be to declare grey as a colour for clouds as well, but then the method would overestimate cloud cover in most cases because blue pixels were classified as grey in several tests. Another problem is that in some images the two algorithms analysed eight oktas, as the images are too bright overall. In these cases the blue pixels of the images are to whitish, so the algorithm declares them as white and therefore as clouds.

As already stated before, the SVM algorithm works a bit different. This algorithm learns from the SWIMSEG dataset and therefore the colours and RGB values are more specific for this data. In the MAE and RMSE across all categories, this method performs best. The error increases with the number of oktas. This is to be expected, as the number of possible analyses is highest in categories three or four oktas. Therefore, in the area of the highest categories the error decreases again. The BIAS is lowest with this method compared to the others. However, this method generally slightly underestimates the cloud cover. The SVM was trained on 5 images and these means 120000 data points. The computational effort would increase significantly if the number of images and their sizes were increased, but the results would almost certainly improve.

Figure 77: Example of the SWIMSEG dataset where ground truth analysis doesn't match cloud image.

In this thesis, the ground truth analysis from the SWIMSEG dataset is assumed to be the correct solution. Due to the lack of verification data, this is appropriate. Figure 77 and the analysis in figure 37 shows for example, that in this dataset the analyses do not always correspond to the exact cloud image. Thus, the entire evaluation must be treated with a certain degree of caution. In general, however, the error can be neglected due to the amount of data.

In contrast to the analysis and verification with the SWIMSEG dataset the results for the images with landscape are slightly different. The Hybrid and KNN show the best results. Although the clouds are not analysed quite accurately, the algorithms mostly isolate the landscape. This is due to the fact, that also other colours, not only white and blue, are in the training data. Only the number of clusters shall be reconsidered, because the higher the variety of the colours in the image the higher the number of clusters. The SVM and the two threshold methods do not detect the landscape. The explanation for this is rather simple. As the SVM is learning from the SWIMSEG dataset and this database only contains images with clouds and sky, the model cannot detect other colours than these two. The threshold methods and the corresponding threshold are also designed for the determination of clouds in the sky. Hence, these methods have difficulties detecting other colours.

7.1.1 Outlook

There are several options especially in the field of machine and deep learning to improve this analysis. First, as stated above, the training dataset for the SVM can be extended, also to images with landscape (if there is a corresponding ground truth). Of course, the computing time for the entire model rises significantly, but the quality of the analyses should increase. Another approach with a deep neural network was done by Dev et al. (2019) [31]. In this thesis, a CNN is used to encode the structures of the cloud image and this image is then decoded back by up-sampling the image to an output image. Also, another technique is to train a neural network instead of the SVM on some example images. However, the potential of threshold methods is, in my opinion, very much exhausted. Since the machine and deep learning methods are much more powerful and functional.

7.2 Problem: cloud classification

Since the classification of clouds is sometimes difficult even for meteorologists, the solution with computer models is not entirely trivial.

We can see from the results that for relatively simple datasets like SWIMCAT, all models are able to solve the problem in a reasonable amount of time (1000 epochs) and with a high level of accuracy. Although more complex models are used in the literature, the SVM performs relatively well in comparison.

The application of the three different CNN models shows that even a simple architecture is sufficient to solve the classification problem of the SWIMCAT dataset. But, more complex models like the CloudNet model by Zang et al. (2018) [7] can even achieve a perfect score with 1000 epochs. However, it takes some practice and experience to find the right setting and the right hyperparameters for the respective model. The problem with all datasets is, that all deep learning models tend towards slight overfitting. However, this is mainly due to the rather small dataset. This can be seen in the learning curves (figure 65, 68 and 72) when the loss of the validation dataset reaches higher values than the loss function of the training dataset. In general, however, the classification problem of the first dataset (SWIMCAT) is adequately solved by all models.

The difference, on the contrary, is with the CCSN dataset. The learning curves show that two of the three models could not apply the learned patterns on the validation set during training and also on the test dataset. This a sign that the models are not able to predict on new data (validation and training dataset), indicating that the models are starting to fit on noise and are beginning to overfit. The model by Zhang et al. (2018) generally shows no learning rate in the learning curves. This is an indication that the model is stuck at a loss plateau during the gradient descent method.

The four confusion matrices (figure 76) demonstrate that the problem cannot be solved with the used models and defined hyperparameters. The three deep learning models do not generalise at all, and most of the test images are assigned to one class. With the SVM, the situation is slightly different, as the diagonal can be predicted to some extent. These factors now point to a major overfitting issue. This is the case with the majority of deep learning models, and there are several approaches to solving it.

First of course the architecture. More complex architecture will lead to easier overfitting of the model. Reducing the number and depth of layers can prevent overfitting. Other ways to avoid overfitting are collected in the term "regularisation" [32]. Two techniques, already implemented in the models, are the dropout layers and the batch normalisation [33]. These two regularisation methods slightly reduced overfitting, but not sufficient enough.

Another way to avoid overfitting in models, however, is also to improve the dataset. It turns out that there are cloud images in certain categories in the CCSN dataset that do not actually

fit there. Some examples of incorrectly classified images can be seen in figure 78. The first cloud image was categorised as cirrus, although it is more likely to be altocumulus. In the next example it is also more likely to be cumulus or again altocumulus clouds. Example number three should be in the category of cumulonimbus or cumulus. The fourth image is also more in the cumulus category. These are only a few examples of many in the dataset.

But how did the authors of Zhang et al. (2018) [7] achieve such good results with their model? The solution probably lies in the extremely low learning rate of the model. A learning rate of 0.001 is quite normal for deep learning models. But in their model, the learning rate decays with every iteration and is additionally reduced by a factor of 10 every 5000 epochs [7]. The whole model is also trained with 20000 epochs. This is a very large number for deep learning models. Even with a high-end GPU, the model probably needed several days of computing time. It is therefore not surprising that the model is able to learn the structures of the cloud images with such a low learning rate and such a long training period. The only question is whether it can now be applied to images outside the dataset.

When the CCSN dataset is used for the model by Zhang et al. (2018) it is stuck in a loss plateau during the training process. Verifying the learning rate can be a solution to this problem. Another approach is described in *Cyclical Learning Rates for Training Neural Networks* by Smith et al. (2015) [34]. Training with cyclical learning rates instead using fixed or monotonically learning rates achieve higher classification accuracy and can be a way out of learning plateaus.

Figure 78: Incorrectly categorised cloud types: 1. cirrus, 2. nimbostratus, 3. stratocumulus, 4. stratus. Correct categories: 1. altocumulus, 2. cumulus/altocumulus, 3. cumulus/cumulonimbus, 4. cumulus

7.2.1 Outlook

The results for the second dataset (CCSN) are not satisfying. A next step could be to improve the dataset by re-categorising the cloud images. Another approach could also be to create an own dataset. The Karlsruher Cloud Atlas provides images that are already categorised. Another way to solve this problem could be semi-supervised models. As these models use labelled and unlabelled data, the CCSN could provide the labelled images and images from the internet the unlabelled dataset. A semi-supervised deep learning model is the generative adversarial network (GAN) by Goodfellow et al. (2014) [35]. This model is often used

for classification of images. The GAN generates images by learning the structure from the already existing cloud images. Other models, such as AlexNet, VGG-19, pretrained models, and a variety of others, could be used to solve this classification problem. There are hardly any limits in the field of deep learning and more and more models are becoming available due to the popularity of neural networks.

In this thesis, an overview should be given of the possibilities that exist today to determine cloud cover and cloud types with automatic methods. Especially in the field of machine and deep learning, there are countless methods to solve this problem. However, I see the biggest problem in obtaining and categorising the data. As mentioned in the beginning, even among meteorologists there is sometimes disagreement about the determination of cloud types. This was also evident in the creation of the CCSN dataset, as there were also difficulties with categorisation. The methods from AI achieve good results even with smaller datasets (SWIMCAT), but the dataset must be sufficiently well categorised. The tools and models to analyse the data and to provide a solution for a classification problem described in the thesis are already available. In the end, however, the models can only be as good as the dataset they are based on.

Bibliography

[1] O. Boucher, D. Randall, P. Artaxo, C. Bretherton, G. Feingold, P. Forster, V.-M. Kerminen, Y. Kondo, H. Liao, U. Lohmann, P. Rasch, S. K. Satheesh, S. Sherwood, B. Stevens, and X. Y. Zhang. *Clouds and aerosols*, pages 571–657. Cambridge University Press, Cambridge, UK, 2013. doi:10.1017/CBO9781107415324.016.

[2] Qingyong Li, Weitao Lu, and Jun Yang. A hybrid thresholding algorithm for cloud detection on ground-based color images. *Journal of atmospheric and oceanic technology*, 28(10):1286–1296, 2011. doi:10.1175/jtech-d-11-00009.1.

[3] M Costa-Suros. Cloud cover estimation based on ceilometer measurements: a comparison with visual observations. *17th International Conference on Clouds and Precipitation*, 2016. doi:10.13140/RG.2.1.4433.1769.

[4] R Chauvin, J Nou, S Thil, A Traoré, and S Grieu. Cloud detection methodology based on a sky-imaging system. *Energy Procedia*, 69(C):1970–1980, 2015. doi:10.1016/j.egypro.2015.03.198.

[5] Soumyabrata Dev, Yee Hui Lee, and Stefan Winkler. Color-based segmentation of sky/cloud images from ground-based cameras. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 10(1):231–242, 2016. doi:10.1109/jstars.2016.2558474.

[6] Soumyabrata Dev, Yee Hui Lee, and Stefan Winkler. Categorization of cloud image patches using an improved texton-based approach. In *2015 IEEE International Conference on Image Processing (ICIP)*, pages 422–426, 2015. doi:10.1109/ICIP.2015.7350833.

[7] Jinglin Zhang, Pu Liu, Feng Zhang, and Qianqian Song. Cloudnet: Ground-based cloud classification with deep convolutional neural network. *Geophysical Research Letters*, 45(16):8665–8672, 2018. doi:https://doi.org/10.1029/2018GL077787.

[8] François Chollet. *Deep Learning with Python*. Manning, November 2018.

[9] George Spencer Young and Sue Ellen Haupt. Teaching artificial intelligence to meteorology undergraduates. *85th AMS Annual Meeting, American Meteorological Society - Combined Preprints*, pages 4961–4964, December 2005. 85th AMS Annual Meeting, American Meteorological Society - Combined Preprints ; Conference date: 09-01-2005 Through 13-01-2005.

[10] Petrina Papazek, Irene Schicker, Claudia Plant, Alexander Kann, and Yong Wang. Feature selection, ensemble learning, and artificial neural networks for short-range wind

speed forecasts. *Meteorologische Zeitschrift*, 29, 10 2020. doi:10.1127/metz/2020/1005.

[11] Haochen Li, Chen Yu, Jiangjiang Xia, Yingchun Wang, J. M. Zhu, and Pingwen Zhang. A model output machine learning method for grid temperature forecasts in the beijing area. *Advances in Atmospheric Sciences*, 36:1156–1170, 2019. doi:10.1007/s00376-019-9023-z.

[12] Aurélien Géron. *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow: Concepts, tools, and techniques to build intelligent systems*. O'Reilly Media, 2019.

[13] T Kanungo, D.M Mount, N.S Netanyahu, C.D Piatko, R Silverman, and A.Y Wu. An efficient k-means clustering algorithm: analysis and implementation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(7):881–892, 2002. doi:10.1109/tpami.2002.1017616.

[14] C. Cortes and V. Vapnik. Support vector networks. *Machine Learning*, 20:273–297, 1995. doi:10.1007/BF00994018.

[15] Intisar Shadeed Al-Mejibli, Jwan K Alwan, and Hamed Abd Dhafar. The effect of gamma value on support vector machine performance with different kernels. *International Journal of Electrical and Computer Engineering*, 10(5):5497, 2020. doi:10.11591/ijece.v10i5.pp5497-5506.

[16] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, 12:2825–2830, 2011. doi:10.5555/1953048.2078195.

[17] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[18] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. doi:10.1109/5.726791.

[19] Saad Albawi, Tareq Abed Mohammed, and Saad Al-Zawi. Understanding of a convolutional neural network. In *2017 International Conference on Engineering and Technology (ICET)*, pages 1–6, 2017. doi:10.1109/ICEngTechnol.2017.8308186.

[20] J. Brownlee. *Machine Learning Mastery with Python: Understand Your Data, Create Accurate Models and Work Projects End-to-end*. 2016. URL: https://books.google.at/books?id=85V6nQAACAAJ.

[21] Eun Joo Rhee et al. A deep learning approach for classification of cloud image patches on small datasets. *Journal of information and communication convergence engineering*, 16(3):173–178, 2018. doi:10.6109/jicce.2018.16.3.173.

[22] Alon Halevy, Peter Norvig, and Fernando Pereira. The unreasonable effectiveness of data. *IEEE Intelligent Systems*, 24:8–12, 2009. URL: http://www.computer.org/portal/cms_docs_intelligent/intelligent/homepage/2009/x2exp.pdf, doi:10.1109/mis.2009.36.

[23] Michele Banko and Eric Brill. Scaling to very very large corpora for natural language disambiguation. In *Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics*, pages 26–33, Toulouse, France, July 2001. Association for Computational Linguistics. URL: https://www.aclweb.org/anthology/P01-1005, doi:10.3115/1073012.1073017.

[24] Dan Tulpan, Cajetan Bouchard, Kristopher Ellis, and Cyrus Minwalla. Detection of clouds in sky/cloud and aerial images using moment based texture segmentation. In *2017 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 1124–1133. IEEE, 2017. doi:10.1109/ICUAS.2017.7991380.

[25] Sylvio Luiz Mantelli Neto, Aldo von Wangenheim, Enio Bueno Pereira, and Eros Comunello. The use of euclidean geometric distance on rgb color space for the classification of sky and cloud patterns. *Journal of Atmospheric and Oceanic Technology*, 27(9):1504–1517, 2010. doi:10.1175/2010jtecha1353.1.

[26] Ahmet Özlü. Color recognition, 2018. URL: https://github.com/ahmetozlu/color_recognition.

[27] Tse-Wei Chen, Yi-Ling Chen, and Shao-Yi Chien. Fast image segmentation based on k-means clustering with histograms in hsv color space. In *2008 IEEE 10th Workshop on Multimedia Signal Processing*, pages 322–325. IEEE, 2008. doi:10.1109/MMSP.2008.4665097.

[28] David Arthur and Sergei Vassilvitskii. K-means++: The advantages of careful seeding. In *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '07, page 1027–1035, USA, 2007. Society for Industrial and Applied Mathematics. doi:10.5555/1283383.1283494.

[29] M. Lothon, P. Barnéoud, O. Gabella, F. Lohou, S. Derrien, S. Rondi, M. Chiriaco, S. Bastin, J.-C. Dupont, M. Haeffelin, J. Badosa, N. Pascal, and N. Montoux. Elifan, an algorithm for the estimation of cloud cover from sky imagers. *Atmospheric Measurement Techniques*, 12(10):5519–5534, 2019. URL: https://doaj.org/article/74d5edbff03c49e0b42743aa949672d1, doi:10.5194/amt-12-5519-2019.

[30] Stanski, Laurence Wilson, and William Burrows. Survey of common verification methods in meteorology. *WMO World Weather Watch Tech. Rep.*, 8, 01 1990.

[31] Soumyabrata Dev, Atul Nautiyal, Yee Hui Lee, and Stefan Winkler. Cloudsegnet: A deep network for nychthemeron cloud image segmentation. *IEEE Geoscience and Remote Sensing Letters*, 16(12):1814–1818, 2019. doi:10.1109/lgrs.2019.2912140.

[32] Christian Garbin, Xingquan Zhu, and Oge Marques. Dropout vs. batch normalization: an empirical study of their impact to deep learning. *Multimedia Tools and Applications*, pages 1–39, 2020. doi:10.1007/s11042-019-08453-9.

[33] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In Francis Bach and David Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 448–456, Lille, France, 07–09 Jul 2015. PMLR. URL: https://proceedings.mlr.press/v37/ioffe15.html, doi:10.5555/3045118.3045167.

[34] Leslie N. Smith. Cyclical learning rates for training neural networks. *2017 IEEE Winter Conference on Applications of Computer Vision (WACV) (pp. 464-472)*, June 2015. arXiv:1506.01186, doi:10.1109/WACV.2017.58.

[35] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'14, page 2672–2680, Cambridge, MA, USA, 2014. MIT Press. URL: https://arxiv.org/abs/1406.2661.

List of Figures