Karlsruhe Institute of Technology

SCC
Steinbuch Centre for Computing

AI/ML-based support of satellite sensing for cloud cover classification

Master's Thesis of

Erik Wessel

at the Department of Informatics
Steinbuch Centre for Computing (SCC)

Reviewer:	Prof. Dr. Achim Streit
Second reviewer:	Prof. Dr. Jan Cermak
Advisor:	Dr. Ugur Cayoglu
Second advisor:	Dr. Markus Götz
Third advisor:	Dr. Babak Jahani

11. January 2023 – 11. July 2023

Karlsruher Institut für Technologie
Fakultät für Informatik
Postfach 6980
76128 Karlsruhe

Abstract

Accurate cloud cover assessment is crucial in weather forecasting, climate science, as well as agricultural and urban planning. Weather forecasting and climate science focus on cloud dynamics and analysis, while agricultural and urban planning are interested in cloud-free observations, requiring cloud detection beforehand. However, obtaining reliable cloud cover information in areas without local weather stations is a challenging task, due to limited data availability and spatial variability. This thesis evaluates the effectiveness of satellite sensing in complementing local weather station measurements to provide cloud cover assessments in data-scarce regions. Leveraging rapid machine learning advancements, a transfer learning based cloud coverage assessment model is developed for image classification, using satellite images as a data source. The methodology of this thesis involves gathering data from local weather stations and satellites observations from the Sentinel-2 mission, for which a dedicated microservice-architecture based system is built. Data is collected for selected regions in the U.S.A. that offer a high density of local weather stations, thus providing multiple examples of similar cloud conditions for training. Performance evaluation is conducted using a separate dataset for testing that the model has not previously trained with. Evaluation metrics, including loss, F1-score, and confusion matrices, assess the classification task for ranked classes of various cloud coverage ranges. Using local measurements for training yields an F1-score of around 0.5 to 0.6, with slight deviations of one cloud cover class rank for the majority of cases. Considering slight deviations, results in accuracies ranging from 93.5 % to 99.1 %. Using manual labeling of station related satellite observations for training improves the F1-score to around 0.75. Considering slight deviations, results in accuracies ranging from 98.4 % to 100 %. The findings demonstrate that satellite sensing effectively complements local weather station measurements, enabling accurate assessments of cloud coverage in areas without local weather stations. This research has practical implications, allowing for cloud cover assessment in regions without weather stations, extending monitoring of local condition and analysis of cloud dynamics. Improved accuracy in local cloud cover detection supports the forecasting of precipitation patterns and assists in early detection of extreme weather events, which is crucial in the context of climate change. Future research includes expanding to regions with fewer local weather stations and investigating additional data sources for comprehensive cloud information. In conclusion, this thesis shows the effectiveness of cloud cover assessment using satellite sensing alongside local measurements in data-scarce regions, providing valuable insights for research, decision-making, and improving the understanding of local cloud cover conditions.

Zusammenfassung

Eine genaue Bewertung der Wolkenbedeckung ist für die Wettervorhersage, die Klimawissenschaft sowie die Agrar- und Stadtplanung von entscheidender Bedeutung. Die Wettervorhersage und die Klimawissenschaft konzentrieren sich auf die Wolkendynamik und Analyse, während die Landwirtschafts- und Stadtplanung an wolkenfreien Beobachtungen interessiert ist, die eine vorherige Wolkenerkennung erfordern. Die Beschaffung zuverlässiger Informationen über die Bewölkung in Gebieten ohne lokale Wetterstationen ist jedoch eine schwierige Aufgabe, aufgrund von begrenzter Datenverfügbarkeit und räumlicher Variabilität. In dieser Arbeit wird die Effektivität der Satellitenerkundung als Ergänzung zu den Messungen lokaler Wetterstationen bewertet, um die Bewölkung in Regionen mit geringer Datenmenge zu beurteilen. Unter Nutzung der rasanten Fortschritte im Bereich des maschinellen Lernens wird ein auf Transfer Learning basierendes Modell zur Bewertung der Wolkenbedeckung entwickelt, das Satellitenbilder als Datenquelle nutzt. Die Methodik dieser Arbeit umfasst die Sammlung von Daten von lokalen Wetterstationen und Satellitenbeobachtungen der Sentinel-2-Mission, für die ein spezielles, auf einer Microservice-Architektur basierendes System entwickelt ist. Die Daten werden für ausgewählte Regionen in den USA gesammelt, die eine hohe Dichte an lokalen Wetterstationen aufweisen und somit mehrere Beispiele für ähnliche Wolkenbedingungen für das Training liefern. Zur Leistungsbewertung wird ein separater Datensatz zum Testen verwendet, mit dem das Modell zuvor nicht trainiert wurde. Bewertungsmetriken, einschließlich Verlust, F1-Score und Konfusionsmatrizen, bewerten die Klassifizierung für geordnete Klassen verschiedener Stärke der Wolkenbedeckung. Die Verwendung lokaler Messungen für das Training führt zu einem F1-Score von etwa 0,5 bis 0,6, wobei in den meisten Fällen geringfügige Abweichungen von einem Rang der Wolkenbedeckungsklassen auftreten. Unter Berücksichtigung dieser geringen Abweichungen ergibt sich eine Genauigkeit von 93,5 % bis 99,1 %. Durch die manuelle Kennzeichnung von auf Wetterstationen bezogenen Satellitenbeobachtungen für das Training verbessert sich der F1-Score auf etwa 0,75. Unter Berücksichtigung geringer Abweichungen ergibt sich eine Genauigkeit von 98,4 % bis 100 %. Die Ergebnisse zeigen, dass Satellitenbeobachtungen die Messungen lokaler Wetterstationen wirksam ergänzen und genaue Bewertungen der Wolkenbedeckung in Gebieten ohne lokale Wetterstationen ermöglichen. Diese Forschungsarbeit hat praktische Auswirkungen, denn sie ermöglicht die Bewertung der Bewölkung in Regionen ohne Wetterstationen und erweitert die Überwachung der lokalen Bedingungen und die Analyse der Wolkendynamik. Die verbesserte Genauigkeit bei der Erkennung der lokalen Wolkenbedeckung unterstützt die Vorhersage von Niederschlagsmustern und hilft bei der frühzeitigen Erkennung von extremen Wetterereignissen, was im Zusammenhang mit dem Klimawandel von entscheidender Bedeutung ist. Zukünftige Forschungsarbeiten umfassen die Ausweitung auf Regionen mit weniger lokalen Wetterstationen und die Untersuchung zusätzlicher Datenquellen für umfassende Wolkeninformationen. Zusammenfassend zeigt diese Arbeit

die Effektivität der Bewertung der Wolkenbedeckung mithilfe von Satellitensensorik neben lokalen Messungen in datenarmen Regionen und liefert wertvolle Erkenntnisse für die Forschung, die Entscheidungsfindung und die Verbesserung des Verständnisses der lokalen Bewölkungsverhältnisse.

Contents

List of Figures

List of Tables

Glossary

AI Artificial Intelligence. 12

BOA Bottom of Atmosphere. 26

BT Brightness Temperature. 26

CNN Convolutional Neural Network. 20, 27, 28, 63

COTS Commodity-Off-The-Shelf. 4, 67

DBSCAN Density-based spatial clustering of applications with noise. 40, 41, 43

FLOPS floating point operations per second. 63

GMES Global Monitoring for Environment and Security. 3

GPU Graphical Processing Unit. 20, 57, 63

ICAO International Civil Aviation Organization. 9, 11, 28, 49

KDD knowledge discovery in data. 21

KIT Karlsruhe Institute of Technology. 63

LTA Long Term Archive. 32, 34, 40

MAC multiply-accumulate. 63

METAR Aerodrome routine weather report. vii, 4, 5, 9–12, 28, 29, 31–36, 40, 43, 46–49, 51–56, 58, 60–65, 67, 77

MSE Mean Squared Error. 15, 17, 61, 62, 64

NDSI Normalised Difference Snow and Ice Index. viii, 26, 57, 59–62, 64, 67, 77

NDVI Normalised Difference Vegetation Index. 26, 27

NOAA National Oceanic and Atmospheric Administration. 12, 34, 49

SCC Steinbuch Centre for Computing. 63

SPECI Special Meteorological Report. 10, 28, 29

SVM Support Vector Machine. 12, 17, 27

SWIR short-wave infrared. 9, 26, 57

TAF Terminal Aerodrome Forecast. 10, 28, 29

TCI True Color Image. 46, 50, 63

TOA Top of Atmosphere. 10, 26

VNIR visible and near-infrared. 9, 26

WMO World Meteorological Organization. 9, 28, 49

1. Introduction

The importance of satellite sensing has increased significantly in research, as it captures vast areas with consistency, allowing for more comprehensive, global analysis of earth [Dub+21]. There are many fields of research, where satellite sensing can be applied, including oceanography [Fu+19], monitoring of climate and climate change [Ari+21; Mit23], agricultural monitoring [Atz13], urban planning [KMB16], and weather forecasting [Mit23]. This can be partially attributed to the open data policies from NASA's Landsat satellite program [NAS] started in 2008 and ESA's Copernicus program [ESAd] started in 2014 that built on the existing Global Monitoring for Environment and Security (GMES) program. Consequently, researchers now have access to a vast amount of high-quality data that has facilitated significant advancements in remote sensing [Zhu+19a]. The climate's chaotic nature makes simulation difficult, since small changes in initial conditions lead to vastly different outcomes [Ari+21]. These initial conditions are one of various factors of influence that affect the accuracy of climate models. Clouds in satellite imagery pose challenges in applications of remote sensing like urban planning, where a clear image of earth's surface and buildings is essential for accurate image analysis. Cloud masking algorithms generate masks that identify cloud and sometimes cloud shadow pixels, which are useful for distinguishing these areas from other features in satellite imagery. Aside from being undesired in ground based remote sensing analysis, clouds are important in the regulation of earth's climate and weather [Car+22; Nor+16]. The anthropogenic changes to earth's climate induce extreme weather events through a rise in global temperature that, in turn, increases the atmospheric moisture and thus leads to stronger precipitation intensities and possibly water related disasters such as flooding [Car+22; ONe+22; Nor+16]. Given the critical nature of climate change, its current impact, and associated risks that are described in the IPCC report for 2022 [Raw+22; ONe+22], it is essential to predict weather patterns and natural disasters like extreme weather events, tropical cyclones, wildfires, and heatwaves with speed and accuracy, while also monitoring long-term changes in cloud formation. Accurate identification of cloud coverage acts as a building block for further analysis and prediction in both cases.

The accurate assessment of cloud cover in data-sparse regions without local weather stations, poses a significant challenge for aforementioned fields, due to limited local data availability and spatial variability between locations of other weather stations. Satellite sensing shows potential for filling in missing local cloud coverage information in these regions, as several approaches for cloud cover detection in satellite images are already available. There are generally two types of algorithms for cloud cover detection. Single-image algorithms use only the information presented in a single satellite image to assess the given cloud coverage [Hol+96; ZW12; Ric04; Mai+17]. Multi-temporal algorithms use time-series of satellite images to assess the cloud coverage of another satellite image in the same

region [Lon+16; Hag+10]. Multi-temporal algorithms provide more accurate cloud cover detection, while single-image algorithms are less computationally expensive and need less data to provide results. Machine learning models learn from and complement these techniques, understanding hidden patterns and relationships, with the goal to produce cloud cover detection with higher accuracy in less time [Zup; Dom+21; Jep+19; Shi+16; Wan+22]. Both single-image and multi-temporal techniques commonly generate per pixel cloud masks for a given image. Although per pixel cloud masks are accurate, the corresponding algorithms are of high complexity to predict cloud presence per pixel of the image and can be computationally expensive. This makes these approaches unsuited for resource-constrained environments, near-real-time applications (like decision-making), continuous monitoring systems, and application to large datasets in a timely manner. Accurate cloud cover information for these applications depends on faster models (low-latency) and does not require classifications on a per pixel level. This highlights a gap in current research.

This thesis assesses the effectiveness of combining local ground-based measurements from weather stations on earths surface with satellite measurements over a larger area, aiming to derive the local cloud coverage conditions of data-sparse regions without local weather stations. Using local measurements show simplifications in the labeling of satellite imagery and shows potential for the assessment of cloud classification [Rum+15]. The approach developed in this thesis focuses on providing accurate but less complex, faster per-image cloud cover classification capabilities, allowing for efficient on-demand analysis and continuous monitoring in these regions with Commodity-Off-The-Shelf (COTS) hardware. Furthermore, using ordinal classes for cloud cover assessment provides improves interpretability of results by humans, a key aspect for informed decision-making. The model developed in this thesis uses a state-of-the-art deep-learning algorithm as the basis for transfer learning to handle the task of image classification and leverage knowledge of other domains. Conforming to an internationally standardized and commonly used format like METAR, offers potential for adoption by existing systems that rely on local weather station information, without additional effort [WMO22; Int21]. The satellite imagery is collected from the Sentinel-2 satellite mission of the Copernicus program [ESAd], whereas the METARs are gathered from the Iowa State University [IOW]. For each satellite image, relevant weather stations are selected based on criteria, like the difference in time of observation between the satellite image and local measurements, that are applied to the given METARs. If the METAR of a weather station is accepted, a region around its position is extracted from the raw satellite image, building the basis of the dataset. Images are systematically created from mathematical operations on spectral information.

While the model developed in this thesis has the potential to complement areas without local weather-stations for cloud coverage assessment, the applicability for short-term monitoring tasks, like weather forecasting, directly depends on the revisit frequency of the available satellite programs. The revisit frequency of the Sentinel-2 satellites together lies at around 5 days, which may not be enough for applications that need intraday information. To obtain intraday information, satellite programs with high total revisit

frequencies and correspondingly more satellites are required. A satellite program that offers these conditions is the ICEYE Constellation by ESA [ESAf], currently revisiting locations on earth twice a day, increasing to twice a day, when all 48 satellites of the program are in orbit.

Results show that the model is capable of predicting ordinal class-based cloud coverage without deviating from expected results by more than one class in the majority of cases. Accurately predicting the exact class proves more difficult, but also shows inaccuracies in the training data provided by local weather stations. Using manually labeled satellite images instead of weather station METARs, improves both exact predictions and predictions with one class deviations immensely. Moreover, the model is able to process many images in a given time window.

The thesis is structured as follows: chapter 2 breaks down necessary prerequisite knowledge about climate science and machine learning that is necessary for the understanding of this study. After that, chapter 3 provides information about existing studies relating to different aspects of this thesis, exploring cloud cover detection in satellite sensing approaches and applications of local weather station data. chapter 4 dives into the main topic of this thesis, starting with data collection, followed by information on dataset preparation, and finalized by the creation process of the machine learning model. In chapter 5, the machine learning model is evaluated, comparing results, and putting the performance into perspective. chapter 6 concludes the study by revisiting the initial research objective, assessing the contribution of this thesis, and introducing areas for future work.

2. Fundamentals

This chapter explains the fundamental concepts, necessary techniques and methods for the following chapters regarding machine learning and climate science. The aim is to give an introduction to the different topics that provide essential knowledge for the comprehension of decisions taken to develop the machine learning model of this thesis. This chapter is structured to start with the concepts associated with machine learning and is followed by the theory on climate science. Each section is structured in a way that first provides a brief overview on the topic and is then followed by explanations of important concepts in a way that successively builds up on previous concepts.

2.1. Climate Science

Climate science covers multiple disciplines that focus on the understanding of Earth's climate. This includes meteorology, oceanography, geology, ecology and atmospheric science. The climate system consists of the atmosphere (gases, clouds, and aerosols), the hydrosphere (lakes, rivers, ocean), the cryosphere (snow, glaciers, ice sheets, ice on sea), the biosphere (living organisms), and the lithosphere (land surface). All of these partial systems interact with each other, forming the climate system [Bos16]. Weather reflects short-term fluctuations of the climate system, which are characterized by conditions like temperature, humidity, wind, precipitation, and clouds. Climate, on the other hand, refers to long-term changes to the climate system over a period of multiple years. The climate varies naturally over different timescales, which can take shape in short-term events like El Niño and La Niña, where trade winds blowing to the east or west influence the flow of warm water through the Pacific Ocean [Bos16; ONb]. These irregular events usually last between 9 and 12 months and typically occur every 2 to 7 years. They also have global impacts on weather and can cause flooding, hurricanes, droughts, and other severe weather events, aside from which they also influence nutrient distribution in the ocean that is important for marine life. There are also long-term variations in the climate system like the occurrence of glacial periods (ice ages), referring to long periods of time in which a significant amount of the planet is covered by ice and glaciers [Bos16; WWU23]. The most prominent features of ice ages are the low overall temperatures and the expansion of ice sheets across the globe, far beyond the poles. The last glacial period lasted from around 115 thousand years ago to the beginning of the Holocene around 11,700 years ago.

However, humanity itself also impacts the climate system in its own way. The current climate crisis is unequivocally driven by human activities, such as the large scale emission of greenhouse gases like carbon dioxide (CO_2), methane (CH_4), and nitrous oxide (N_2O) from the burning of fossil fuels like coal, oil, and natural gas [Ari+21]. The emitted greenhouse

gases trap heat in the atmosphere, leading to a greenhouse effect and global warming. Over different timescales, both carbon dioxide and methane are absorbed from the atmosphere. While methane's lifetime in Earths atmosphere is around 12 years, carbon dioxide resides there around 300 to 1,000 years [AN]. In comparison to the emission of carbon dioxide, the same amount of methane is more than 25 times as potent at trapping heat in the atmosphere. Additionally, other human influences on climate change are deforestation, since trees absorb carbon dioxide through photosynthesis, industrial activities like cement, steel, and chemical production that release additional greenhouse gases into the atmosphere [Ari+21]. Furthermore, the transformation of land for agriculture or urbanization changes the landscape, thus releasing stored greenhouse gases and removing capabilities for absorbing them from the atmosphere. Human activities have far outpaced the natural absorption capacities of greenhouse gases in the climate system. Continued emissions are going to increase global warming effects, leading to more extreme weather events more frequently, melting ice from the poles and glaciers, a rise in sea-level, changes to precipitation patterns [ONe+22]. Resulting risks include the loss of biodiversity, economic impacts, increased poverty, decline in food security, and water scarcity. Reducing the consequences of global warming requires broad and rapid reductions of greenhouse gas emissions. The highest potentials for reducing the impact of the current climate crisis lie in the transition to clean energy, mainly from solar and wind, as well as the reduced transformation of land and corresponding ecosystem restoration.

Clouds play an important role in the climate system by being part of the water cycle, and shifting the energy balance of the planet [Bos16; Ari+21]. They reflect sunlight back into space, effectively cooling the planet, while also trapping heat that is reflected from Earth's surface and in turn warming the planet. This reflectance contributes to Earth's overall reflectance, called albedo. This balance is a delicate mechanism of the climate system that controls temperature and precipitation distributions. In turn, the rising global temperatures affect the jet streams through more severe warming in the polar regions [Man+18; ONe+22]. The lack of temperature difference leads to a slowdown and general disruption of the jet streams, which leads to longer lasting local weather conditions and potentially extreme weather events. Climate change has already led to various consequences, including rising temperatures, extreme weather events, and a rise in global sea-level, which is why it is crucial to observe and react to current changes [Ari+21; ONe+22]. While the mitigation of additional greenhouse gas emissions is central to reduce the impact of climate change, it is also important to tackle current and future climate related challenges. Observing the amount of cloud cover and its variability is essential to assess the planet's reflectance or albedo and observe feedback mechanisms, as well as extreme weather events like hurricanes and thunderstorms. Through observation and analysis, climate models with higher accuracy can be developed to predict extreme weather events earlier than current techniques.

Capturing and analyzing large amounts of data about clouds enables further exploration of the inner workings of earths climate, as seen from both abstract, global and detailed, local viewpoints [Mit23]. Remote sensing technologies offer great capabilities for monitoring

and analyzing global cloud coverage using satellite imagery. Remote sensing makes it possible to acquire information from various locations around the world over a distance, without having to travel there. This is especially useful to collect information about earths atmosphere like cloud coverage, since it provides a vast amount of data on a global scale, thus accounting for temporal and regional differences. Another advantage over in situ observations is the ability to cover areas with little to no human interaction, as is the case for polar regions and oceans. Due to modern satellite missions offering updates every few days to as frequent as every 15 minutes, tracking of changes in the atmosphere improves, which builds a fundamental data source for weather forecasts and climate models. Usually, satellite-based remote sensing yields data for multiple spectral bands that cover a section of the electromagnetic spectrum, such as visible light and infrared. This allows further analysis of interrelationships of different wavelength reflectance or radiance values for different atmospheric phenomena. Apart from remote sensing, observations from weather stations offer local, point-based measurements from a combination of instruments that register information like temperature, cloud cover, wind speed, and atmospheric pressure that can be used to enhance information from satellite images by adding contextual information. An example for weather information from local stations is the METAR format that is standardized by the World Meteorological Organization (WMO) in correspondence with the International Civil Aviation Organization (ICAO) [WMO22; Int21].

2.1.1. Sentinel-2 Mission

The Sentinel-2 mission is a satellite-based Earth observation program developed by the European Space Agency (ESA) for the Copernicus program. Its objective is to systematically collect high-resolution images on a global scale using two satellites that orbit Earth on opposing sides. The quality of the imagery is coined by a high revisit-frequency of 2 to 5 days and multi-spectral instruments. The spectral bands that are captured by the Sentinel-2 instruments are show in Table 2.1. Overall, the multi-spectral instruments measure 13 spectral bands, covering the range from visible to short-wave infrared light at different resolutions. The highest resolution of 10 meters per pixel is reserved for the visible and near-infrared light bands. Additionally, there are six bands at a resolution of 20 meters per pixel, with four in the visible and near-infrared (VNIR) range and two in the short-wave infrared (SWIR) range for applications in the detection of clouds, snow, and ice as well as the observation of vegetation. The remaining three bands are provided at 60 meters per pixel and focus on the applications of cloud screening, atmospheric correction and cirrus detection. The bands are structured in a way that a higher band number corresponds to a higher Spectral bands 2, 3, and 4 represent the usual color channels blue (around 493 nanometers), green (around 560 nanometers), and red (around 665 nanometers) respectively. Aside from the rest, band 10 provides images that focus only on cirrus clouds without underlying terrain. There are many processing levels for the Sentinel-2 data that process data from the previous level: Level-0, Level-0 Consolidated, Level-1A, Level-1B, Level-1C, and Level-2A. However, the most attention of this thesis lies on the Level-1C processing level. It already encompasses all previous processing steps, such as radiometric corrections, refinement of geometric viewing model, resampling with geometric interpolation, conversion of values to reflectances, and generation of masks.

	S2A		S2B		
Band Number	Central wavelength (nm)	Bandwidth (nm)	Central wavelength (nm)	Bandwidth (nm)	Spatial resolution (m)
1	442.7	20	442.3	20	60
2	492.7	65	492.3	65	10
3	559.8	35	558.9	35	10
4	664.6	30	664.9	31	10
5	704.1	14	703.8	15	20
6	740.5	14	739.1	13	20
7	782.8	19	779.7	19	20
8	832.8	105	832.9	104	10
8a	864.7	21	864.0	21	20
9	945.1	19	943.2	20	60
10	1373.5	29	1376.9	29	60
11	1613.7	90	1610.4	94	20
12	2202.4	174	2185.7	184	20

Table 2.1.: Spectral bands of the two Sentinel-2 satellites A and B taken from [ESAg].

Level-1C does not include scene classification and atmospheric correction to convert Top of Atmosphere (TOA) to surface reflectance, as this is provided by Level-2A. All images are restricted to the boundaries of grid tiles that are geospatially fixed, but do not always cover the tiles completely. Each grid tile is covers around 10,000 square kilometers, while the swath width is as large as 290 kilometers. [ESAg]

2.1.2. METARs

METAR is an internationally standardized format for the reporting of weather observations at airports and aerodromes. They concisely report current weather conditions, focussing on essential information that are relevant to pilots and meteorologists. The format enables efficient communication of crucial weather information that is for example linked to flight safety. However, METARs hold potential for use cases outside of flight operation [Rum+15; NBM17; Gor+22]. The name METAR is an abbreviation or code name that stands for an aerodrome routine weather report or a meteorological aerodrome report. It is usually issued hourly or half-hourly and provides information on various local environmental conditions. To better illustrate the content of a METAR, the following example is broken down into their individual parts:

METAR KSFO 281356Z 26011KT 10SM FEW002 BKN006 OVC008 12/10 A2993

- METAR indicates that the weather report uses the METAR format and not other formats like Special Meteorological Report (SPECI) or Terminal Aerodrome Forecast (TAF).

- KSFO is the ICAO specified four-letter code for the location of the observation, usually containing an abbreviation of the airport. In this case, KSFO refers to the San Francisco International Airport.
- 281356Z encodes the date and time of the observation, where the first two digits stand for the day of the month. The following digits relate to the hours and minutes. The Z at the end is used to indicate Zulu time or Greenwich Mean Time (UTC). So 281356Z translates to the 28th day of the month at 13:56 (military time), using Zulu time.
- 26011KT encodes wind direction in degrees in the first three digits, starting north and moving clockwise, while the rest represents the wind speed. In this example, the wind is coming from 260 degrees (almost completely from the west) at a strength of 11 knots or around 20 kilometers per hour.
- 10SM stands for 10 statute miles or around 16 kilometers and refers to the horizontal visibility from the station, indicating that the visibility is clear in this case.
- FEW002, BKN006, and OVC008 all relate to cloud coverage on different heights. The first three letters indicate how much of the sky is covered by clouds, while the last three digits indicate the base cloud height in 100 feet (ca. 30 meters) above ground level. In the example, there are few clouds at 200 feet (ca. 61 meters), broken clouds at 600 feet (ca. 183 m), and an overcast sky at 800 feet (ca. 244 m).
- 12/10 indicate temperature and dew point, showing it is 12 °C warm, while the dew point lies at 10 °C.
- A2993 is the atmospheric pressure given in inches of mercury (inHg) in this case. The example shows an atmospheric pressure of 29.93 inHg or around 1013.5 hectopascal (hPa).

There are other variations and unmentioned aspects that are not relevant in relation to this thesis or would cause unnecessary elongation of the explanation, which is why the explanation is limited to the given example.

Cloud coverage information is provided by METARs using oktas (1/8). It is described using standardized codes, indicating the portion of the sky that is covered by clouds [WMO22]. Following international standards, the data is collected in the area within a radius of around 16 kilometers of the weather station. There are small regional differences

Code	CLR	FEW	SCT	BKN	OVC
Meaning	Clear sky	Few clouds	Scattered clouds	Broken clouds	Overcast sky
Sky cover	0/8	1/8 to 2/8	3/8 to 4/8	5/8 to 7/8	8/8

Table 2.2.: METAR codes for different conditions regarding cloud coverage and their decoded meanings.

on the implementation of these standards. This thesis focuses on the conventions that are used by the United States according to the information provided by National Oceanic and Atmospheric Administration (NOAA) [ONa], as show in Table 2.2 METARs indicate cloud cover in ascending order until the first layer of overcast clouds is reached. There are special cases that need to be considered when working with oktas for drawing the border between categories:

- CLR is the code for clear sky, used in reports that are made by automatic stations. There is also the code SKC, which also stands for clear sky, but is used for observations by manual stations instead. Since there is no difference in the values they encode, all abbreviations for clear sky are mapped to CLR. The sky is considered to be clear only if no cloud is observed.
- FEW, standing for few clouds, in turn also includes the presence of clouds below 1/8 sky cover
- BKN, represents broken clouds and includes sky coverage larger than 7/8 but smaller than 8/8

2.2. Machine Learning

Machine learning is a part of Artificial Intelligence (AI), computer science, and statistics that focuses on extracting knowledge from input data and making predictions on unseen data without explicit instructions how to do this [MG17]. The models should be able to learn generalizable rules that are also applicable to data that varies from the training input to some degree. To achieve this, machine learning models detect and analyze patterns and relationships that are present in the given data. Machine learning models improve over time and provide scalable ways for automation of complex tasks that might not be as efficiently or effectively automatable by conventional means. There are multiple algorithms that can be used for machine learning like decision trees, Support Vector Machines (SVMs), and neural networks. In the context of the decisions taken in this thesis, neural networks will be explained in more detail than other algorithms and other concepts will focus more on approaches using neural networks. A subfield of neural networks is deep learning that handles tasks with very high complexity especially well. Machine learning models and neural networks are a great fit for the problem of cloud classification, as they are able to learn complex patterns and relationships from data such as images from satellites.

2.2.1. Workflow

Developing a machine learning model for a specific task usually follows a typical workflow [MG17]. Initially, it is important to collect relevant data from various sources that is suitable for the task. The collected data is then preprocessed, which includes a subset of multiple techniques like selecting the most relevant features from the available data, handling of missing data, and transforming the data to meet assumptions of the machine learning mode. The data is then split into training, validation, and test sets. The training

dataset is used to illustrate to the model what the desired outputs are, as well as learn any underlying patterns and relationships of the data. The validation dataset is used to compare the models' predictions with the expected outcome and adapt the training process based on how wrong the predictions are. First training is done, which is followed by validation. This procedure repeats itself for however long the model should learn. The test dataset on the other hand is not used until the model has finished learning. This makes it possible to handle the test dataset as new, unseen data that the performance of the model can be evaluated on in the end. Because of this, it is especially important to pay attention to separating the datasets precisely, preventing information leaking from one dataset into another. When the data is ready, the model is created or selected and adapted based on the task, data, and requirements. It is also possible to combine different specialized models in an ensemble. Following the creation of the model is the training process that repeats the actions of learning structures in the data and evaluating predictions over multiple iterations, called epochs. Training is done using batches of data that are processed in one step, adapting the throughput of data to the capacity of the processing units. Before doing training, adjustment or tuning hyperparameters like learning rate and batch size takes place. Finally, the performance of the model is assessed based on the test dataset. This provides insights into what possible changes to the structure of the model or tuning of hyperparameters might further improve performance. After evaluating the model, adjustments are made before starting the training from the beginning again. Under optimal circumstances, this continues until satisfactory performance is achieved.

2.2.2. Paradigms

There are different paradigms to guide machine learning models in their search for knowledge. They all cover fundamental questions that shape what the model is able to learn and can be seen as a design decision for a machine learning task. Machine learning paradigms are not mutually exclusive and may be combined to improve a model.

Supervised Learning One of the most common approaches is called supervised learning, where the model learns from labeled data in which the labels represent the desired output [MG17]. It answers questions of how to predict future values, or how to classify data into existing classes. During training, the model uses the input-output pairs to derive underlying patterns and relationships. Its performance is validated and then optimized during training by comparing predicted and actual label of the data. Once the model is trained, it can make predictions on unseen data by using the learned patterns and relationships. It works well to categorize input data or to predict continuous values. Areas of application include image and speech recognition, as well as medical diagnosis. A downside of supervised learning is that it requires a well-labeled dataset, where the labeling accuracy has repercussions on the performance of the model.

Unsupervised Learning Another approach is unsupervised learning, where the model does not have information on what it is expected to find mappings for [MG17]. It answers questions regarding what natural clusters and groups reside in the data, or whether the

data can be expressed in a more concise form. It needs to detect patterns, structures, and relationships on its own. This type of machine learning algorithm is applicable when the goal is to discover hidden patterns or groups in the data that are unknown at the time of model creation. Such a way of information gathering is especially useful for clustering of data, where similar data is grouped together, and for dimensionality reduction that tries to reduce the number of input features without removing relevant information.

Semi-Supervised Learning Semi-supervised learning combines the approaches of supervised learning and unsupervised learning by only requiring a smaller portion of labeled data and a larger portion of unlabeled data [Hea16]. The idea is to let the model learn on the labeled data, in order to provide general guidelines of what is of interest, and train on unlabeled data to find hidden patterns and relationships. The approach thus tries to leverage the benefits of both supervised and unsupervised learning for achieving better generalization. The effectiveness of using semi-supervised learning over its component paradigms varies based on multiple factors. Semi-supervised learning introduces pseudo-labels to the training process that are predictions made from learned patterns and relationships of previous training on the labeled data. By combining these pseudo-labels with actual labels, the dataset can be processed like in supervised learning. However, the quality of predictions has a direct feedback on its training and might hinder performance.

Reinforcement Learning Reinforcement learning is similar to supervised learning in the sense that it is provided values for orientation. It answers questions of how a system can learn to interact with an environment in a desired way [SB05]. However, instead of labeling input data in a static context, reinforcement learning is performed in a dynamic environment in which the model learns through trial and error. The orientation values are given in the form of rewards and penalties based on the actions taken by the model. The goal is to derive a policy or recommendation that maximizes the cumulative reward over time. This makes it applicable in the context of controlling characters in video games, or autonomous systems. For example, a penalty may be given to an autonomous system for exerting too much stress on some of its components or moving parts too close to a human operator.

2.2.3. Transfer Learning

Aside from machine learning paradigms, transfer learning can be considered a methodology that complements paradigms that it is applied to. The idea of transfer learning is to utilize knowledge gained from a different but related task or domain [Zhu+19b]. When little data is available, this technique aims to achieve better performance than what training from scratch on an insufficient dataset would yield. To implement this, a pre-trained model is used, which is the saved state of an already trained neural network. The model can then be either used as-is from that save state, or modified to better adapt to the task at hand. Such a modification could be the exchange of the output layer to match the new task or domain. If the previous model used a fully connected layer with 100 outputs as a classification layer at the end, one could replace it with a fully connected layer with 10 outputs for use in a

classification task with 10 labels. This type of modification is called feature extraction and has the great benefit of not having to train the underlying base model. Another approach is fine-tuning, that works by unfreezing some base models' layers. This helps learn feature representations that are more common in the dataset of the new task. Transfer learning helps improve model generalization for the new task that it is applied to. It also speeds up the training process and helps in situations where labeled data is limited.

2.2.4. Classification

Classification is a task for a machine learning model with the goal of predicting to which class or category a certain input belongs [MG17]. This is commonly implemented as a supervised learning technique that learns from a set of labeled data. Each data unit is assigned a label, corresponding to one possible class. The model is implemented by using a fully connected layer as the output layer, with as many neurons as there are classes. After training, the model has generalized the learned relationships and patterns and can make predictions to which class new data belongs to. This implies that the model is able to learn what identifies and differentiate certain classes. Classification tasks exist in the form of binary classification, multi-class classification, and multi-label classification. Binary classification tries to classify data into two mutually exclusive classes. A possible example of this would be labeling whether an email should be considered to be spam. Multi-class classification similarly has multiple – possibly more than two – mutually exclusive classes from which one should be predicted per data unit. Examples include image classification e.g., images of birds, cats, and dogs into these three classes, or automatic recognition of handwritten letters and digits. Multi-label classification is somewhat different to the other classification types, since there can be multiple classes that apply to an input, removing mutual exclusion. This is useful for tagging all possible categories or classes that an input belongs to. In image recognition, this would induce that learned objects or even people are recognized. A common loss function to use for classification tasks is the Cross-Entropy Loss that computes the logarithmic loss between predicted probabilities and true labels.

2.2.5. Regression

Regression is a machine learning task that aims to predict one or more continuous numerical values [MG17]. Like classification, it also falls under in the category of supervised learning techniques, using labeled data for training and one neuron per value to predict. The training process also helps generalize learned relationships and patterns to make predictions of what values should be assigned to unseen data. There are multiple forms of regression available to find a function that maps input data accurately to output values. The simplest form of regression is linear regression, which tries to fit a linear equation to the numerical values. A polynomial regression goes a step further by using a polynomial equation that has a higher degree than in the linear case and is able to capture more complex relationships. In even more complex cases, it might be appropriate to use non-linear regression to fit a nonlinear function to the data. This makes use of more complex underlying structures like neural networks to better capture patterns and relationships from the data. A common loss function for regression tasks is the Mean Squared Error

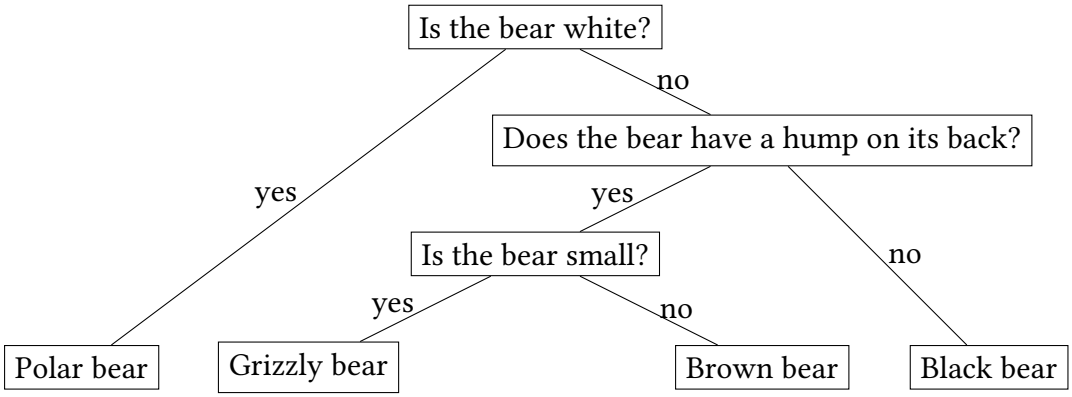


Figure 2.1.: Visualization showing the structure of a simple decision tree on the classification of bears based on a guide from the National Park Service, U.S. Department of the Interior [Inta].

(MSE) loss that computes the average squared differences of prediction and actual value, and thus penalizes larger deviations more severely.

2.2.6. Algorithms

There are multiple algorithms to choose from when implementing machine learning for a specific task. The selection of algorithms comes with a tradeoff between different aspects like flexibility, complexity, speed, dimensionality, and robustness.

Decision Trees Decision trees are non-parametric supervised learning algorithms that are applicable to both classification and regression tasks [MG17]. They are structured hierarchically like trees, as seen in Figure 2.1 for how to differentiate types of bears. The decision tree consists of nodes and branches that start at a root node and progress over multiple internal nodes towards leaf nodes. Root and intermediate nodes corresponds to decisions to be made, from which branches with different answers diverge. Leaf nodes are instead interpretable as outcomes of the decisions taken to get there, starting from the root node. As can be seen in the visualization, the final classification of the type of bear is located in the leaf nodes, while questions are represented by intermediate nodes. The root node is the initial question, from which the others diverge. Due to their boolean logic and corresponding visualization, decision trees can be easily interpreted compared to more complex algorithms like neural networks. Another advantage is the flexible handling of discrete and continuous data and missing values, making it a good option for data mining. However, decision trees easily overfit, are sensitive to data variance, and are more costly to train than other algorithms.

Random Forests A random forest is an ensemble learning method, using multiple decision trees that make decisions together through aggregation of the individual results [MG17]. The main difference to simple decision trees is feature randomness, which ensures that individual decision trees only consider a random subset of features instead of all

features, lowering correlation among them. Aggregation of results could for example be implemented as averaging for regression tasks or majority vote for classification tasks. The central advantages are a reduced risk of overfitting, along with being applicable to both regression and classification tasks. Since values have to be calculated for multiple decision trees, this algorithm is more computationally expensive than its singular tree counterpart.

Support Vector Machines SVMs aim to find the optimal hyperplane with maximized margins in an N-dimensional space, so that it splits data points into separate classes specified through supervised learning [MG17]. Maximizing the margins increases the distance of the hyperplane to the data points, future proofing it for the categorization of edge-cases in unseen data. Support vectors are the data points that are closest to the hyperplane and thus are most involved in constraining it. Accordingly, deleting support vectors entails a regeneration of the hyperplane. In cases of non-linear classification tasks, kernels are used that transform the data to a higher-dimensional space, in which the classes can be linearly separated by a hyperplane again. The main advantages of SVMs are their effectiveness in high dimensional spaces, their memory efficiency resulting from using a subset of data – the support vectors, and their ability to handle non-linearity. On the other hand, choosing the right kernel function might be crucial in avoiding overfitting when there are way fewer data points than features or when the data contains much noise. Furthermore, necessary computations are more expensive for larger datasets.

Neural Networks Neural networks try to mimic neurons and their interactions by organizing artificial neurons in layers and propagating information in one direction, while changes to neurons are propagated in the opposite direction [MG17]. They consist of layers of nodes with at least one input, hidden, and output layer as visualized in Figure 2.3. The first layer handles incoming information like text, speech, or imagery and is called the input-layer. The last layer returns results like classification or regression and is called the output-layer. Layers that lie in between these layers are not directly accessible and thus called hidden layers. Artificial neurons – or simply called neurons in the context of machine learning – are connected to other neurons of the directly neighboring layers and possess information about associated weights, a bias, and an activation function. A simple description of such a neuron, called a perceptron, is given by Rosenblatt [ROS58]. A neuron is presented in Figure 2.2 that show how input signals $x_1 \dots x_3$ are processed, resulting in the output value y. The received signals $x_1 \dots x_3$ are accumulated according to weights $w_1 \dots w_3$ and a bias b. The signal is then transformed using an activation function as seen in Figure 2.4. In many cases, a Sigmoid or ReLU function is used, introducing non-linearly into the network. This non-linearity, which heavily depends on the choice of activation function, is the main advantage over the perceptron model. After this so-called forward pass, accuracy of a neural network model is assessed through a loss function such as MSE or Cross-Entropy that should be minimized throughout the training process. To let the model learn from its mistakes, backpropagation is used, where the error is propagated backwards through the network. This allows to compute the gradients of weights and biases that are used by an optimization algorithm like steepest gradient descend (SGD),

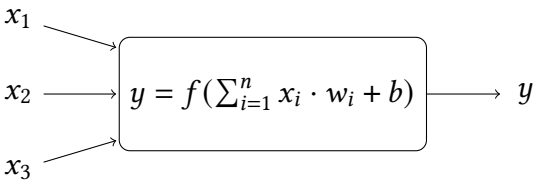


Figure 2.2.: Visualization of a single neuron

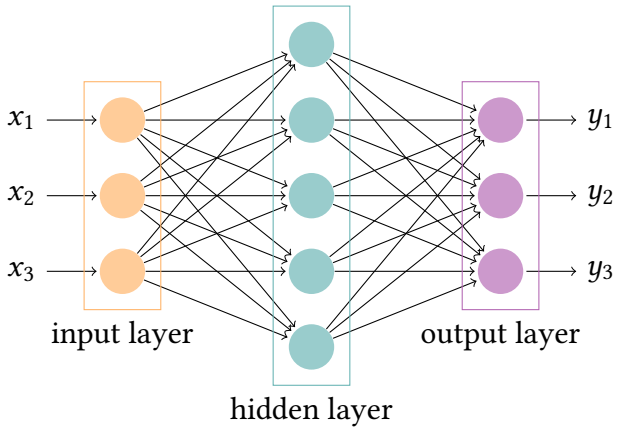


Figure 2.3.: A visualization of a simple neural network with three layers. There are three neurons in the input layer, five in the hidden layer, and three in the output layer. x_1, x_2, x_3 are input values and y_1, y_2, y_3 are output values.

which try to minimize the error by adjusting said weights and biases. One advantage of neural networks is their ability to learn non-linear and complex patterns and relationships from raw data without additional feature engineering requirements, making them a good choice for analyzing images, text, and speech data. They also have a high fault tolerance and are able to scale well, handling large amounts of data while further improving their accuracy. Using specialized hardware allows to efficiently parallelize processing for faster training. In comparison to clearly visualizable algorithms like decision trees, neural networks are difficult to understand because of their black-box nature. They are also less useful if not enough data is available that can be learned from. Deep learning is an application of neural networks that utilizes multiple hidden layers to learn more complex patterns associated with a given task. Fields with tasks of such high complexity include computer vision, speech recognition and natural language processing.

2.2.7. Activation Functions

As previously shown in paragraph 2.2.6, activation functions are at the core of introducing non-linearity into neural networks. If the selected activation function is linear, the machine learning model is not going to be able to solve non-linear problems [MG17]. Yet in reality, non-linear problems are all around us. Some exponential relationships are population growth, compound interest, and epidemic spread, making it an essential influence on the capabilities of a neural network. A subset of common activation functions is shown here

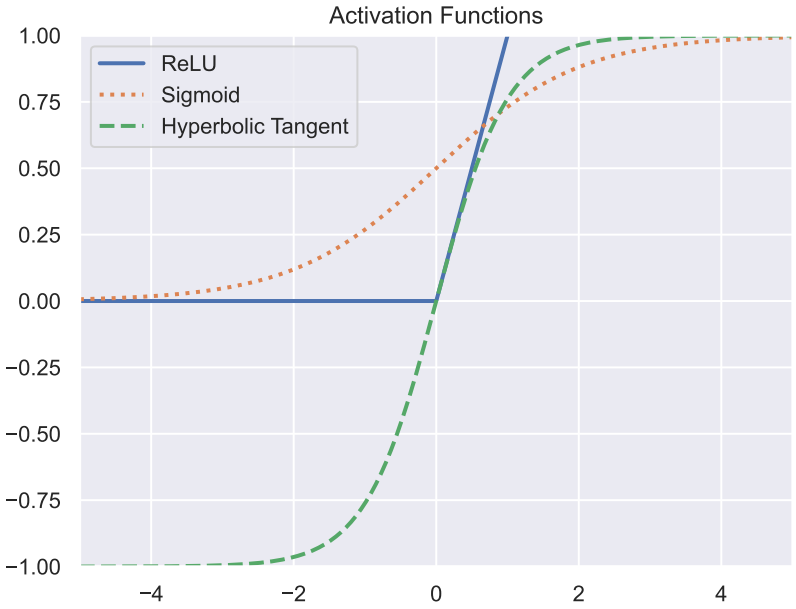


Figure 2.4.: A selection of various activation functions that can be applied in neural networks are ReLU, Sigmoid, and Hyperbolic Tangent.

with regard to deep learning applications and inter-function comparisons as explored by Nwankpa et al. [Nwa+18].

Sigmoid The sigmoid activation function restrains input values to the interval [0, 1]:

$$f(x) = \frac{1}{1 + e^{-x}}$$

It is commonly used for binary classification, since the output resembles a probability score. Additionally, the output is differentiable at any point and the function is also used for logistic regression. Drawbacks include slow convergence, vanishing gradients, gradient saturation, and issues with gradient updates due to not being centered around zero.

Hyperbolic Tangent The hyperbolic tangent activation function restrains input values to the interval [-1, 1]:

$$f(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$$

It is better than the sigmoid activation function in terms of training performance due to being zero centered. The function is still affected by vanishing gradients and can generate dead neurons during computation.

Softmax The softmax activation function produces values in the interval [0, 1] from a vector of real numbers:

$$f(x_i) = \frac{e^{x_i}}{\sum_j e^{x_j}}$$

It is often applied in multi-class classification problems to compute the probabilities of each possible class, ensuring that the sum of the probabilities is equal to one. The possibilities can then be compared to the target class for training.

ReLU The rectified linear unit function (ReLU) is an activation function that offers great performance and generalization capabilities for deep learning:

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

It is almost linear, upholding the potential for optimization of linear models using gradient-descent techniques and resulting in fast computation times. Setting inputs less than zero to zero solves the problem of vanishing gradients that other activation functions suffer from. A drawback of ReLU is that it can more easily overfit.

2.2.8. Convolutional Neural Networks

Convolutional Neural Networks (CNNs) are a class of deep learning models for processing and analysis of grid-like data like images, which is why they are often utilized in computer vision tasks. They excel at tasks that use complex input data such as images or audio, like speech. CNNs offer a scalable way of classifying images and recognizing objects in images. These capabilities come at the price of increased computational demand, and may require the use of Graphical Processing Units (GPUs) for the model training process. CNNs consist of three types of layers:

Convolutional Layer The fundamental building block of a CNN is the convolutional layer, which applies movable filter kernels on an image, resulting in a feature map. The filter kernel or filter is a matrix (or other-dimensional tensor), typically of size 3×3, that is applied to an area of the image of the same size with the goal of checking for the presence of a feature. The filter is applied to the image by computing the dot product between filter and image values at overlapping positions, also known as the receptive field. This concept can also be extended to an image with 3 color channels, as shown in Figure 2.5. The grid acts as an interpretation overlap between pixels of images and positions inside tensors. The filter kernel is chosen to be three-dimensional (3×3×3) in this case, in order to return a two-dimensional output. In a first step, the filter is applied to an equally sized section of the input image. The values are then combined with a dot-product to receive one value of the output tensor. Finally, the whole filter kernel is moved by some amount of grid-cell, also known as stride, on the input image to compute the next iteration, which continues until the output tensor is fully constructed, and the kernel has passed over the entire image. As the illustration suggests, through layers behind the currently computed output tensor, there may be multiple filters present to compute multiple outputs. The example only shows the computation using a single filter for a single output tensor, which is offset from the other output tensors. The size of the output depends on various factors such as the input size, filter size, stride. Other factors include padding and dilation. Padding adds values to all sides of the input to enable movement of the filter over the original image bounds. Dilation controls positional spacing between filter kernel applications. Equation 2.1 presents how

the output dimension out_i is calculated in the PyTorch framework. i refers to the dimension of interest like width, height, and depth. Each convolution operation is followed by the application of an activation function, introducing non-linearity.

$$\text{out}_i = \left\lfloor \frac{\text{in}_i + 2 \cdot \text{padding}_i - \text{dilation}_i \cdot (\text{kernel\_size}_i - 1) - 1}{\text{stride}_i} + 1 \right\rfloor \tag{2.1}$$

Pooling Layer After the creation of the feature map using convolutional layers, a pooling layer reduces the dimensionality. They act similar to convolutional layers in the sense that they use filter kernels that move over the entire input to generate the output. However, instead of applying the filter weights, the input values are aggregated instead. There are two major types of pooling operations:

- **Max pooling:** The maximum value inside the receptive field is sent to the output.
- **Average pooling:** The average value of the receptive field is sent to the output.

The downsampling applied by the pooling layer reduces complexity and improves efficiency of the model.

Fully-Connected Layer To complete the model that is created from convolutional layers and pooling layers, a fully connected layer at the end of the model is used for classification. The number of neurons used in this layer is equal to the number of classes that have to be determined from the input. Accordingly, this layer may be replaced by another fully-connected layer when using transfer learning to retrain the classification part of the model in a new domain without removing knowledge of patterns and textures that are learned in previous layers.

2.2.9. Data Mining

Data mining or knowledge discovery in data (KDD) is the methodology of discovering patterns, relationships, and other insights from data in order to gain a better understanding, make predictions or identify trends. Gaining descriptive or predictive insights through such a data analysis and visualizing them are important steps to guide and improve decision-making. Before delving into data mining, it is essential to clearly define the objectives that should be achieved through data analysis. When this decision is made, previously collected data is prepared for analysis. This includes cleaning the data by removing duplicates, missing values, as well as noise and outliers if possible and desired. In cases where the dimensionality of the data is very high, it might be appropriate to use dimensionality reduction techniques to promote a faster analysis. In a next step of the data mining process, algorithms are used to analyze the data. Depending on the specific requirements of the task at hand, a spatial or temporal examination of patterns and relationships may take place. Different aspects are interesting under different, previously set objectives. While the analysis of common, repetitive patterns and associations with related data might be of interest to better understand customers, outliers could be an indicator useful for fraud

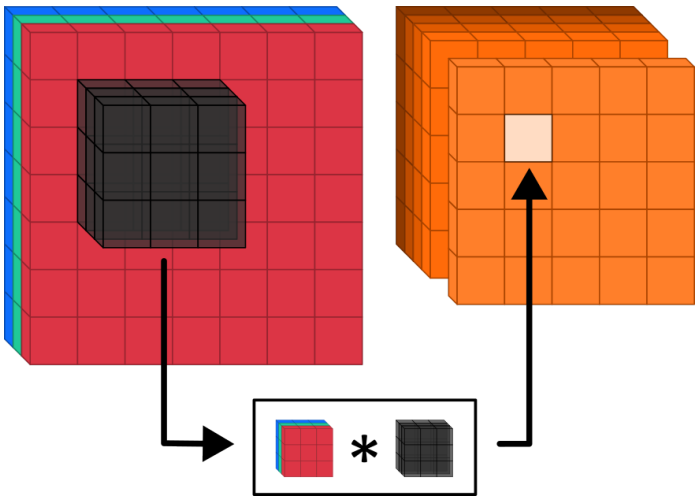


Figure 2.5.: Visualization of one step of the convolution computation in a convolutional layer for an image with 3 color channels (RGB) and a three-dimensional filter kernel (black). The output tensor is one of multiple tensors (orange) with individual filters.

detection. Algorithms for classification, clustering, or regression might be applied based on the underlying structure of the data like whether it is labelled. Finally, the results of the analysis are evaluated and interpreted with regard to the initially defined objective to plan the further course of action.

2.2.10. F1-Score

A fundamental addition to the usage of accuracy for model optimization is the F1-Score. It is a concept associated with the predicted labels of a machine learning model and the actual labels of the underlying data [MG17]. Before explaining what F1-Score is, it is important to understand the concepts of **precision** and **recall**. Precision refers to the number of true positives in relation to all predicted positives. In the case of a multi label classification problem, precision can be observed on a per-label basis. For a given label **A** of **n** labels, the true positives refer to cases where the predicted label **A** and actual label match, while the remaining predicted positives – the false positives – refer to the cases where the prediction falsely classifies the data as being labeled **A**. Precision is calculated as:

$$\text{precision} = \frac{\text{true positives}}{\text{true positives} + \text{false positives}}$$

Recall, on the other hand, refers to the number of true positives in relation to all actual positives. For a multi label classification problem, recall can be observed on a per-label basis as well. For a given label **A** of **n** labels, the true positives refer to cases where the predicted label **A** and actual label match, while the remaining actual positives – the false negatives – refer to the cases where the prediction falsely classifies the data as not being

labeled **A**. Recall is calculated as:

$$\text{recall} = \frac{\text{true positives}}{\text{true positives} + \text{false negatives}}$$

Both precision and recall are conceptually restricted to the interval from 0 to 1. A perfect precision score of 1 corresponds to all data labeled **A** actually being **A**, but does not provide information on whether all **A** data is included in the results. A perfect recall score of 1 on the other hand corresponds to including all data that is actually **A**, but does not provide information on how much data that is not **A** is included as well. The F1-Score describes the harmonic mean of the precision and recall values, and thus facilitates the combination of both approaches. It is calculated as:

$$\text{F1-Score} = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$$

By definition, the F1-score is restricted to the interval from 0 to 1 as well. To get the F1-score for multiple labels, data is accumulated. For accumulation, techniques like micro averaging, macro averaging, and weighted averaging can be applied:

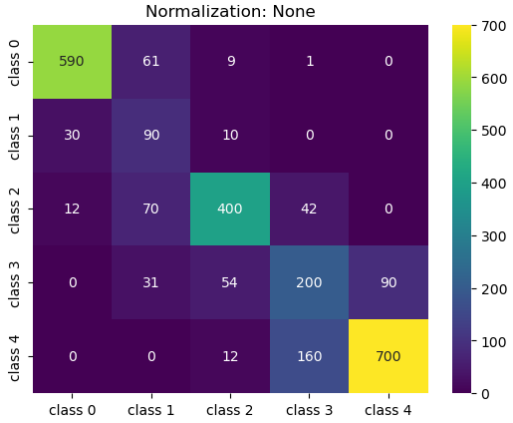
- Micro averaging: Compute global average F1-score from the sums of all True Positives, False Positives, and False Negatives
- Macro averaging: Compute the arithmetic mean of all per-label F1-scores
- Weighted averaging: Compute the mean of all per-label F1-scores, but weight them according to how much data is of that label

Micro averaging is equal to the concept of accuracy. A great benefit of using F1-scores with weighted averaging is that is still relevant when using an unbalanced dataset.

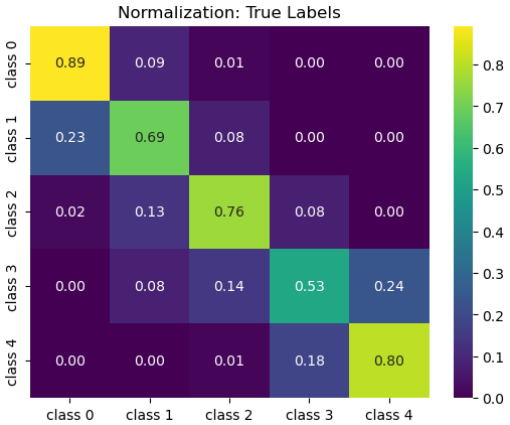
2.2.11. Confusion Matrix

The confusion matrix is another tool to evaluate the performance of a machine learning model, providing a visual component in the form of a table [MG17]. The matrix is organized in such a way so that each entry $C_{i,j}$ contains the value that corresponds to the amount of data that belongs to the label i is predicted to belong to the label j. So if the predictions (columns) are to align with the targets (rows), the matrix should have most entries on the diagonal. This approach makes results more easily comparable and drives development visually to get more entries on the diagonal and fewer entries in the upper and lower triangular matrices. The resulting table is then most commonly visualized with a heat map to highlight differences. Since the dataset might be imbalanced, it is advised to use some form of normalization:

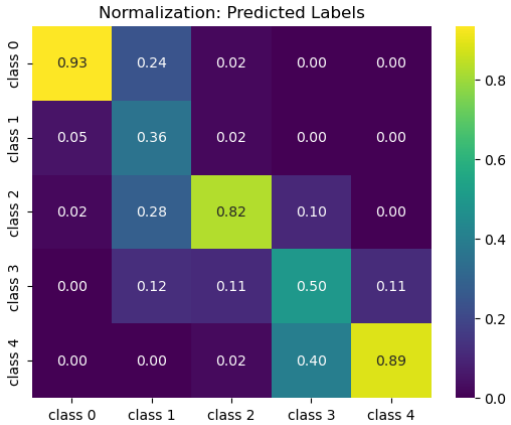
- true: the sum of each row is equal to 1 – What is data of label **A** predicted as?
- predicted: the sum of each column is equal to 1 – What actual label does the prediction result **A** correspond to?



(a) Confusion matrix without normalization.



(b) Confusion matrix normalized for actual / true labels.



(c) Confusion matrix normalized for predicted labels.

Figure 2.6.: Different normalization of confusion matrices for the same data.

- all: the sum of all entries is equal to 1 – The relative values from dividing by the dataset size

Examples of the first two normalization techniques are seen in Figure 2.6, that uses the same data for all matrices. Without any normalization (a) entries of classes with more data (class 0, class 2, class 4) are more prominent, despite predictions of class 1 and class 3 also performing well for the majority of cases. Normalizing for targets (true) (b) provides a solid basis to evaluate how well targets are predicted. Normalizing for predictions (c) instead allows identifying the spread of predictions to different targets. Since imbalanced datasets have less or more cases per target label (row), only the true-label normalization offers performance comparability between classes. Without explicit notice, confusion matrices in this thesis will use the **true** normalization as a basis for analysis and model improvement, as it highlights the misclassifications per label and input class imbalances are addressed directly.

3. Related Work

As the climate is a chaotic system that makes predictions and simulations difficult, automatic and efficient detection of cloud coverage in satellite images and the creation of detailed cloud masks is still a hot topic. Whether the reason lies in the observation of cloud formation and development or in examining the ground, cities, and infrastructure, the detection of clouds is the first required step to proceed, facilitating the foundation of further processing. Since the approach of this thesis uses satellite imagery and local weather observations, related work is explored in both fields in the next sections. Previous work is put into perspective by including information on how it relates to the approach proposed by this thesis.

3.1. Cloud Detection

This section focuses on cloud detection in satellite imagery, acknowledging that there are different approaches through Lidars for example, as seen by Yorks et al. [Yor+21]. The most common form of cloud detection models works on a per-pixel basis, creating a cloud mask as a result. There are two general categories of cloud masking algorithms. On one hand, there are single-image algorithms that observe a single image and build the corresponding cloud mask from that image. On the other hand, multi-temporal algorithms that facilitate cloud mask generation through observation of the same area of a certain period of time to better separate moving clouds from the earth's surface. Single-image algorithms are straightforward. Their simplicity reduces implementation overhead and makes them more easily applicable to new, unseen images without the need for additional data, making them faster than multi-temporal algorithms. However, having only the information that is present in this single image, they lack the ability to capture contextual and temporal information, such as similarities in the neighborhood of a pixel on a cloud over multiple images with potential cloud movement. They struggle with distinguishing clouds from bright surfaces, snow, and ice, as they rely on spectral band information and spatial patterns. Additional difficulties can arise from overlapping clouds and complex cloud formations. This inherent difference reduces the quality of the resulting prediction. Multi-image algorithms are more accurate and robust, providing better results. Due to the higher computational complexity and need for additional data from time-series, these algorithms are not as fast as single-image algorithms. Another drawback is the essential necessity to have such a time-series available, making it not applicable to unseen regions. The model developed throughout this thesis belongs to the category of single-image algorithms, additionally integrating local point-based measurements to refine the result instead of time-series information. In the scope of this master's thesis, the model is restricted to classifying an image as a whole instead of producing a cloud mask. However,

the underlying image preprocessing and dataset creation techniques may be applied to image segmentation as well.

Single-image algorithms An early cloud detection algorithm is ACCA by Hollingsworth et al. [Hol+96] that uses thresholds derived from physical rules to detect clouds in satellite imagery. It builds upon the algorithm developed by the Earth Resources Observation and Science Data Center [Intb] and uses images from the Landsat satellites. The algorithm uses a statistical approach with n-dimensional histograms to identify cloudy pixels. The Fmask algorithm by Zhu et al. [ZW12] produces a mask for clouds, cloud shadows, and potentially snow in Landsat L1T imagery from physical properties based on the spectral bands for TOA and Brightness Temperature (BT). Potential cloud and cloud shadow pixels are computed using thresholds and with NDSI and Normalised Difference Vegetation Index (NDVI) values in mind for separation of thin clouds from snow, ice, and vegetation. Additionally, the resulting potential cloud and cloud shadow layers are matched using geometric relationships. Building on this base, Zhu et al. [ZWW15] also adapts the application to Sentinel-2 imagery and highlights the advantage of the included spectral band (band 10) for the detection of cirrus clouds in comparison to the usage of the thermal band that is provided by Landsat imagery of earlier missions. Fmask version 4 by Qiu et al. [QZH19] reaches an overall accuracy of 94.59 % for Landsat 8 imagery and 94.30 % for Sentinel-2 imagery. ATCOR by Richter et al. [Ric04] first pre-classifies pixels into categories based on spectral information from VNIR and SWIR bands from the Sentinel-2 mission like ACCA by Hollingsworth et al. [Hol+96] and Fmask by Zhu et al. [ZW12] [ZWW15] which uses Landsat data instead. The categories are land, water, snow and ice, cirrus clouds, non-cirrus clouds, haze, and shadow. ATCOR then applies atmospheric correction, which uses the radiative transfer equation, to remove atmospheric scattering and absorption effects, followed by topographic correction that accounts for differences in elevation and slopes of the observed terrain. The influence of topography on remotely sensed data is explored by Proy et al. [PTD89] and by Smith et al. [SLR80] as early as 1980. The Sen2Cor algorithm by ESA [Mai+17][ESAe] also focuses on the removal of atmospheric correction effects and is employed to correct Sentinel-2 Level-1C images (TOA) to Level-2A images (Bottom of Atmosphere (BOA)). Like the other algorithms, Sen2Cor encompasses the classification of pixel values to cloud and cloud shadow, among other categories. A comparison by Zekoll et al. [Zek+21] shows performance differences between masking models, where the overall accuracies are 89 % for Fmask, 91 % for ATCOR and 92 % for Sen2Cor. The data is taken from Sentinel-2 Level-1C at 20 globally distributed testing sites, resampling all spectral bands to 20 meters per pixel. To account for differences in classification behavior, the outputs are mapped to eight classes – including background – that focus on clear sky, (semitransparent) cloud, cloud- and topographic shadow, water, as well as snow and ice. Similar to other algorithms, the model of this thesis uses data from the Sentinel-2 Level-1C, which does not account for atmospheric correction to capture the extent of clouds in satellite images. The model uses the VNIR spectral bands for red, green, and blue light.

Multi-temporal algorithms MACCS-ATCOR joint algorithm (MAJA) by Lonjou et al. [Lon+16] builds upon the techniques from both MACCS developed by CNES and CESBIO and ATCOR. MACCS uses a multi-temporal approach itself, while ATCOR is a single-image algorithm. Similar to other approaches, MACCS handles clouds, cloud shadows, water, and snow before applying atmospheric correction to the images. MAJA is more than the sum of its parts, as new methods are introduced as well. For example, MAJA includes correction methods for haze and cirrus clouds. It also introduces a way to mix image sets of different missions, with the goal of combining data from the Sentinel-2 and Landsat8 satellites. The Multi-Temporal Cloud Detection (MTCD) model developed by Hagolle et al. [Hag+10] uses satellite images at constant viewing angles to better differentiate between cloudy and clear sky conditions per pixel. It uses imagery from the FORMOSAT-2 and LANDSAT missions and focuses on abrupt changes in reflectance values between pixels, with additional information on linear correlation from neighboring pixels. To handle scenarios of landscape changes like melting fields of snow, the algorithms require an adequate timescale that captures these gradual changes. The necessary timescale varies based on individual conditions like observing a seasonal or a short term change in landscape. If these changes are not accurately captured, the temporal consistency degrades, leading to incorrect interpretations of the algorithms due to the difference in observed spectral and spatial patterns.

Machine learning algorithms Machine learning is a field of research that complements both single-image and multi-temporal approaches. It provides the potential to improve cloud classification accuracy by learning complex, hidden patterns and relationships. Machine learning algorithms enable advancements in cloud detections through integration with other techniques for the identification of cloud cover. This allows the algorithms to augment the technique they learn from in order to reach better performance and differentiate between clouds and anomalies that are detected by the original algorithm. Entering the realm of machine learning and CNN, KappaMask by Domnich et al. [Dom+21] collected satellite images from the Sentinel-2 mission over the land of Northern Europe. It utilizes a U-Net architecture with image segmentation to generate masks for clear sky, cloud, semi-transparent cloud, and cloud shadow. Another mask is reserved for invalid values. In the context of the models' architecture, dice coefficient loss produced better masks, even though categorical cross entropy loss had lower validation loss. The dice coefficient, as a statistical measure for evaluating the similarity between datasets and – contrary to accuracy – focusing on the identification of specific regions, is located at 59 % for Sen2Cor and at 61 % for Fmask. KappaMask highlights the potential of deep learning approaches by achieving a dice coefficient of 76 % for Sentinel-2 L1C and 80 % for Sentinel-2 L2A imagery. The S2coudless algorithm performs slightly better than Fmask at 63 %, but worse than KappaMask, even though it uses machine learning as well. Another approach by Bai et al. [Bai+16] uses SVMs for machine learning and suggests the usage of multi-feature fusion, which aggregates information provided by features. The feature selection regards spectral bands, texture features like arithmetic mean, variance, and contrast. NDVI resides as a separate, but included feature category. The data is collected from the Gao Fen-1/2 satellite missions. While the model does not account for differences

in seasons, it does not rely on a thermal band or time series. A different idea is presented by Yu et al. [YL21] that uses unsupervised classification results to train an ensemble model, which decision tree learners then identify the clouds. It is applied on a per-pixel basis on Landsat8 images using 10 spectral bands and achieves an accuracy improvement of around 10% compared to Fmask 4.0 by Qiu et al. [QZH19]. The RS-Net model by Jeppesen et al. [Jep+19] uses a CNN with a U-net architecture and data from the Landsat8 Biome and SPARCS datasets with corresponding cloud masks. The model focuses on the spectral bands for red, green, blue, and infrared light and achieves good results from the addition of spatial pattern recognition that such deep-learning models excel at. The model that is developed by Shi et al. [Shi+16] uses a CNN and focuses on reducing the error in cloud detection for complex scenarios and thin clouds. It uses the simple linear iterative cluster (SLIC) method to cluster the image into superpixels to reduce noise and reduce processing time, from which a cloud probability map is then generated and refined for the final result. UATNet by Wang et al. [Wan+22] uses a transformer-based approach instead to overcome the smaller receptive field of CNN models, while still opting for a U-net architecture. This thesis' model uses both a CNN and transformer-based model for state-of-the-art image classification, called EfficientNetV2 and Swin Transformer V2, respectively. They act as an interchangeable basis of the model, where transfer learning is applied to leverage previous knowledge from the ImageNet dataset for classification of clouds. EfficientNetV2 by Tan et al. [TL21] is the second version of the model developed by Tan et al. [TL19] and Swin Transformer V2 by Liu et al. [Liu+21a] is the second version of the model developed by Liu et al. [Liu+21b].

3.2. Local Weather Observations

There are multiple forms of observations that are available from weather stations around the world. The WMO and the ICAO work on global standards regarding weather reports and their application in the aviation sector, making them a good data source for weather observations around the world. Information about the different formats is made available by the WMO [WMO22] and by the ICAO [Int21]. Aviation weather reports that are valid for the immediate time of observation include METARs and SPECIs. METARs encompasses routine weather events and is issued at intervals of one hour or 30 minutes. SPECIs focuses on special weather events, but uses the same encoding as METARs. They are issued, when certain significant weather criteria are met. If METARs are issued every 30 minutes, an additional SPECI report is not necessary. Both types of reports include a section for trends and remarks, but their main purpose remains centered around local observations at some point in time. TAFs on the other hand, provide forecasts of meteorological conditions over a given period of time. While METARs and SPECIs are often issued automatically, TAFs are manually created by professionals, following international practices and guidelines. Their forecast periods lie between 6 and 30 hours. Reports that are created routinely and are valid for less than 12 hours are to be issued every 3 hours, while reports for a period of over 12 hours are expected to be issued every 6 hours instead. This thesis uses METARs due to their high availability of common weather phenomena. A specified time-limit restricts METARs to ensure the drift between local and satellite-based observation is small. This thesis fills

the gap in research by using automated local weather observations as ground truth values to train a machine learning model on the task of cloud detection in satellite images, without precomputed or manually created cloud masks. Aviolat et al. [ACC98] shows how the creation of METARs can be improved by using artificial intelligence for cloud observation, by using information available from a pyrgeometer for cloud amount estimation and a laser ceilometer to get the clouds base heights. In contrast, the model developed in this thesis aims at detecting clouds in images from satellite observations, comparing it to METARs that are used as ground truth for learning. Novotny et al. [Nov+21] assess the formal quality of the different meteorological reports issued from inside the Czech Republic and accumulated over several years. The approach is guided by the ICAO regulation Annex 3 [Int18]. This includes checking the syntax and date of the reports for validity. The individual problems are categorized based on severity into either errors (high severity) or warnings (low severity). Errors make it impossible to interpret the report completely and without ambiguities, excluding the corresponding report from further processing. Warnings mostly encompass false categorizations of change groups that would allow the creation of such a report, which is not enough to exclude the report from following steps. METARs achieve an accuracy of around 98.68% (N = 113'989) and the accuracy of SPECIs is around 97.66% (N = 10'184). TAFs that are created by human professionals show higher error rates. Of the around 12'800 assessed TAFs, only around 10% of reports are without issues, while around 10% to 21% contain at least one error. Most of the reports – around 80% – contain at least one warning but no error. In a next step, METARs are used to assess the accuracy of TAFs, resulting in a success rate between 73.1% and 79.6%. Since there is no way to encode changes to the cloud amount, differences between expected and actual values negatively impact the success rate. In this thesis, the METAR report type is used for assessment of actual cloud coverage, since it is issued regularly and in shorter time intervals than other report types. Furthermore, the assessment by Novotny et al. [Nov+21] shows problems in the success rate of cloud amount prediction of TAFs, where METARs provide these values through direct measurements around the desired point in time. The work by Nastos et al. [NBM17] explores the influence of Föhn winds in the region of Crete Island (Greece) on human heat stress perception. It further shows the significance and information quality of METAR-based weather information, as METARs are used as the data source for half-hourly information on air temperature, relative humidity, wind speed, and cloud coverage. Gornyy et al. [Gor+22] uses both satellite data and METARs to explore an automatic assessment of the minimum mortality temperature as a result of urban overheating. The infrared satellite imagery is collected from the Landsat7 and Landsat8 missions. It focuses on METARs from two stations in Helsinki (Finland), where one station is exclusively used for analysis of the results. This concept is applied on a larger scale in this thesis, where METARs are gathered from significantly more weather stations and processed automatically as a source for the developed machine learning model.

4. Methodology

4.1. Data Collection

Before machine learning can be applied, the necessary data has to be collected, preprocessed, and combined from various sources. This chapter focuses on how the data is collected, as well as preprocessing steps that are common regardless of application, like filtering and formatting. To improve reuse, the code employed for data collection follows a microservice architecture that enables developers and researchers to add or replace different services of the whole without having to restructure a rigid, monolithic application and thus reducing the scope of necessary changes. The microservices are independently and fully automatically deployable using containers provided by Docker [Doc] and focus on different capabilities. Communication is facilitated through HTTP-REST requests. Following the leading machine learning frameworks PyTorch [Fou] and TensorFlow [Goo], the microservice API is written in Python. As for this thesis, two services are provided that are also written in Python. One service is responsible for collecting and preprocessing satellite images from the Sentinel-2 mission, while the other service facilitates access to METARs from globally distributed weather stations. In addition to accessing the functionality of the services, the API also offers functionality to automatically parse python data classes for METARs after loading the corresponding data into a table using the Pandas library [Num]. Using data classes helps with the processing of METAR entries like sky conditions, which contain information that can not be described by a single value.

4.1.1. Sentinel-2 Satellite Imagery

The service encapsulates access to data from the Copernicus Open Access Hub [ESAb] with login management that allows multiple people to download data through individual accounts. The quota for the download request is taken from the Copernicus user account that initiated the request. However, since login information is not stored locally as a security measure, the system is not able to continue pending requests after restarting. Because of this, the service will instead continue the request when a user requests the same satellite image again, using their quota for the request instead. A typical workflow first requests metadata for a given region and time frame, followed by the initiation of data requests for products (satellite imagery) and finally the preprocessing of that product, returning an archive file of cut-out regions of the whole image around named geospatial positions with a given radius in meters. The extracted images are squares, where the side length is equal to the diameter computed from the radius and divided by the resolution in meters per pixel. The archive contains JPEG2000 images of all queried spectral bands for the named geospatial positions. In case of the product being not currently available

but retrievable from the Long Term Archive (LTA), a corresponding request is initiated, and a scheduler queries the data in an interval of 30 minutes. Since not all data might be available at all times, requests for data always return the corresponding state so that the users of the API can choose how to react to it. The following states are defined:

- New – The request is new and has not been processed yet
- Invalid – The identifier of the satellite image is not valid
- Pending – The satellite image data is not locally available but online and a request to retrieve it from the LTA has been made
- Incomplete – The satellite image data is partially downloaded (if the download has previously been interrupted, it will now be continued)
- Available – The data is available locally and can be further processed to extract regions from it
- Unavailable – The data is not available locally and no longer available from the Copernicus Open Access Hub
- Processed – The data has been processed and can be removed from storage

Finally, the service offers the functionality to query all grid tiles that cover parts of a multi-polygon input geometry in geospatial coordinates, which enables partitioning in subsequent tasks.

4.1.2. METARs and Weather Station Metadata

The second service is responsible for gathering METARs from weather stations using the information provided by the University of Iowa [IOW]. It enables the user to query both metadata and METARs for stations, by either specifying the names of the stations or requesting all stations contained in a user-defined polygon. Following a typical workflow, the metadata for weather stations for a given region and time frame is retrieved first. Since the location of weather stations provided by the University of Iowa is inaccurate in a few cases, the service automatically updates the position in correspondence to geometric information on country borders around the world, which will help with future processing of weather station metadata. The METARs are requested next, by selecting from available properties and units of measurement (e.g., temperature in degrees Celsius, wind speed in meters per second), which are automatically unfolded from the compressed METAR format. METARs are stored in their original, abbreviated form in a PostgreSQL database to facilitate partial unfolding on request, while retaining all information that they originally contained. The underlying database technology is utilized in through a library with object-relational mappings to eliminate dependencies to a single database technology. The service efficiently handles download requests for METARs of various stations for a given date and time range. Since the service provided by the University of Iowa restricts requests to whole days, the dates are derived from the date and time range first. Through a local database

access, information is collected about what METARs are already present for which station in the specified date range. This implicitly provides information about for what dates METARs are missing, which is utilized to exclude stations from the download, whose date range is completely covered by locally available data. With the goal to minimize the load on the service provided by the University of Iowa, both the number of requests and the number of stations per request are considered. To minimize the number of requests, the METARs are collected for all station where some data is missing, as only a single date range can be given per request. The missing dates of these stations are then combined to allow an algorithm to divide it into contiguous date chunks, removing dates for which all stations already have information locally available. Finally, the METARs are queries per date chunk for all weather stations in batches of at most 100 stations. The retrieved data is then stored into the database, making sure that no duplicate entries are created through masking with the previously queried availability of METARs per station in the selected date range. After completion, all METARs are queried for the specific range of date and time. The METARs are unfolded next, revealing their contents in an automatically processable format defined by the API. The unfolded METARs are then returned to the user.

4.1.3. Collecting Data from the Services

Before collecting data, the date time range and regions of interest are specified. If not specified otherwise, the data is collected for the past six months from the date of the request. They are passed onto later metadata-queries of satellite imagery and weather stations. Initially, metadata of the satellite imagery is collected that contains the identifiers of each product among other information like cloud coverage in percent and the identifier of the grid tile. Since the objective of this thesis is to inspect the enrichment potential of satellite imagery with ground based weather data for machine learning purposes, metadata about the weather stations is retrieved next. The geospatial information of both metadata datasets is then combined, resulting in weather stations clustered by grid tiles. A visualization of all weather station positions that are available from the service provided by the University of Iowa can be seen in Figure 4.1. Each weather station is shown as a single dot on the map, where the color represents the elevation of the area around the weather station using the magma color map. Violet represents values near the sea level, while yellow stands for high elevation of mountainous areas. In total, there are around 82'000 unequally distributed weather stations available. With usability and quality related pre-filtering of stations in mind, a higher density of weather stations that provide more data points is desirable. While the density on the level of magnification of Figure 4.1 looks similar for the USA, Mexico, and Europe, a higher magnification reveals larger differences as can be seen in Figure 4.2. The globally highest densities of weather station that provide METARs are available in the USA, which is why the data of this thesis is limited to a selection of states of the USA. Another reason for choosing an area of high station density is that downloading large amounts of data to cut out small portions around a single weather station is not efficient. Since Sentinel-2 grid tiles cover around 10,000 square kilometers, and the area for which predictions of weather stations are issued lies around a 16 kilometers radius, this would mean that only 2.56% of the data is used, while the rest would simply get

discarded after processing is complete. Even though the whole grid tile is downloaded, wastefulness can be reduced to a minimum by selecting areas of higher density. For each of the satellite products, the METARs of the related grid tile cluster of weather stations are retrieved for a given time window of +/- 30 minutes, as seen in algorithm 1. Afterward, the METARs are analyzed and filtered to ensure the information is available that is later used for machine learning, meaning that in the case of this thesis the cloud coverage must be specified using one of the codes explained in Table 2.2. These codes will also be called cloud cover classes from this point, since they act as categories for the ranking of cloudiness. Since cloud coverage is seen in satellite images from a top-down perspective, the overall cloudiness is assessed in contrast to the per layer cloudiness that might be given by METARs. This maximum cloud coverage is computed by ranking the METAR cloud coverage from lowest to highest cloudiness according to the classes in Table 2.2. This does not account for the fact, that cloud covers might overlap in such a way that the cloud cover class is actually increased. An example of overlapping clouds at different heights is seen in Figure 4.3, which is taken from California's coastline. The clouds in this image can be differentiated fairly well by taking note of the shadows cast by higher clouds. The remaining METARs are grouped by satellite products to avoid the unnecessary download of satellite products for which no METARs are available and contain cloud coverage information. Then, the satellite products that do have METARs that satisfy the requirement of containing sky condition information are downloaded. While the handling of LTA requests is implemented, it will not be fully utilized for the collection of data due to the implied significant time overhead and usage quota. According to the Copernicus user guide [ESAc], satellite products are restored within a few hours and kept available for download for at least three days. However, there is a quota that is adapted to the access patterns of the user, which can not be queried itself. Then, the archive files are constructed, where each named geospatial position corresponds to the name and position of a weather station inside the grid tile of the given satellite product. With improved reusability of the collected data in mind, metadata of Sentinel-2 satellite products and weather stations is provided alongside the satellite imagery and METARs.

4.1.4. Selected Regions

As described in subsection 4.1.3, satellite imagery and local observations are collected from the United States of America. Data is collected over time for individually selected states according to their different climatic conditions and terrains that could affect the detection, availability, and types of clouds. The states are chosen according to the goals of having a high diversity, while maintaining a reasonable dataset size. Aside from flat topographies, regions with mountain ranges are included for the representation of orographic clouds. A basis for the selection is given by the Köppen-Geiger climate classification map, created by [Bec+18] and visualized in Figure 4.4. The map is developed for the time period between 1980 and 2016 and uses the same classification criteria as by Peel et al. [PFM07], which are a slight variation from the original criteria by Köppen et al. [Köp36]. Additional climatic information is taken from up-to-date climate maps provided by NOAA [ONc], which are accumulated from around 1981 until around 2010. Severe weather events include intense thunderstorms and tornados. While this does not factor in the impacts of climate change

Algorithm 1: METARs querying algorithm

Data: products, stationMetadata, radius, datetimeRange, metarProperties, timeWindow
Result: metars
metars ← DataFrame of METAR
stations ← List of STATION
for station ∈ station_metadata **do**
    tileFootprint ← products[station.product].footprint
    if station.geometry.sphere(radius).boundingBox().within(tileFootprint) **then**
        stations.append(station)
metars ← queryMetars(stations, datetimeRange, metarProperties, timeWindow)
metars ← metars.filter(*lambda* metar: metar.skyConditions not empty)
metars ← metars.filter(ARESKYCONDITIONSVALID)
return metars

Function ARESKYCONDITIONSVALID(metar)
    maxCloudCover ← metar.skyConditions.getMaxCloudCover()
    return maxCloudCover ∈ {CLR, FEW, SCT, BKN, OVC}

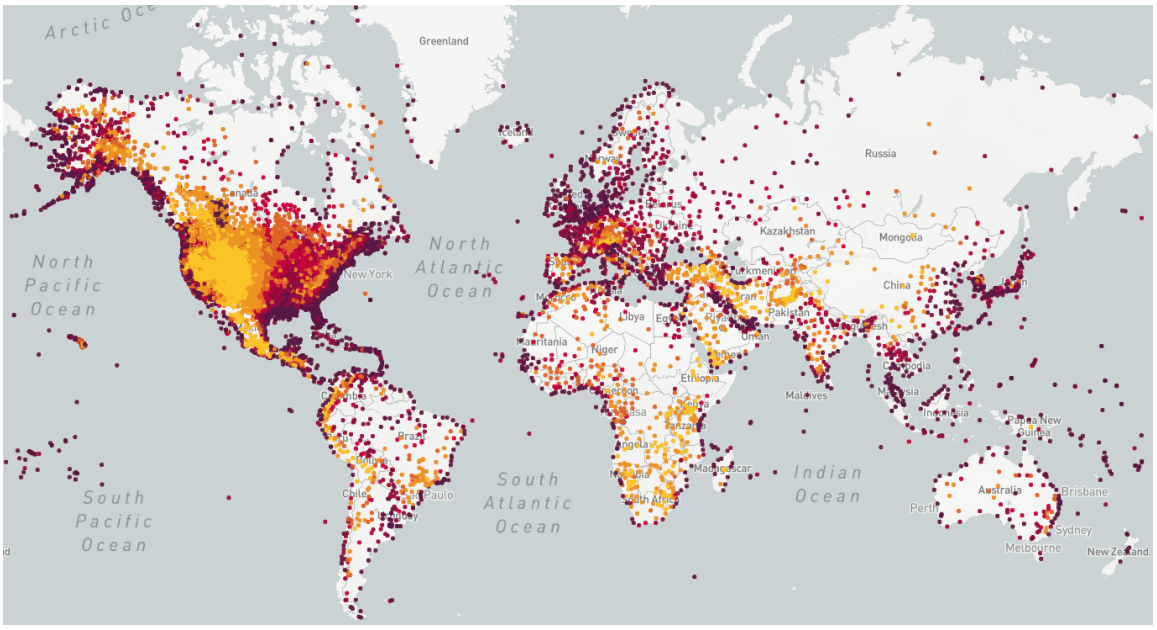
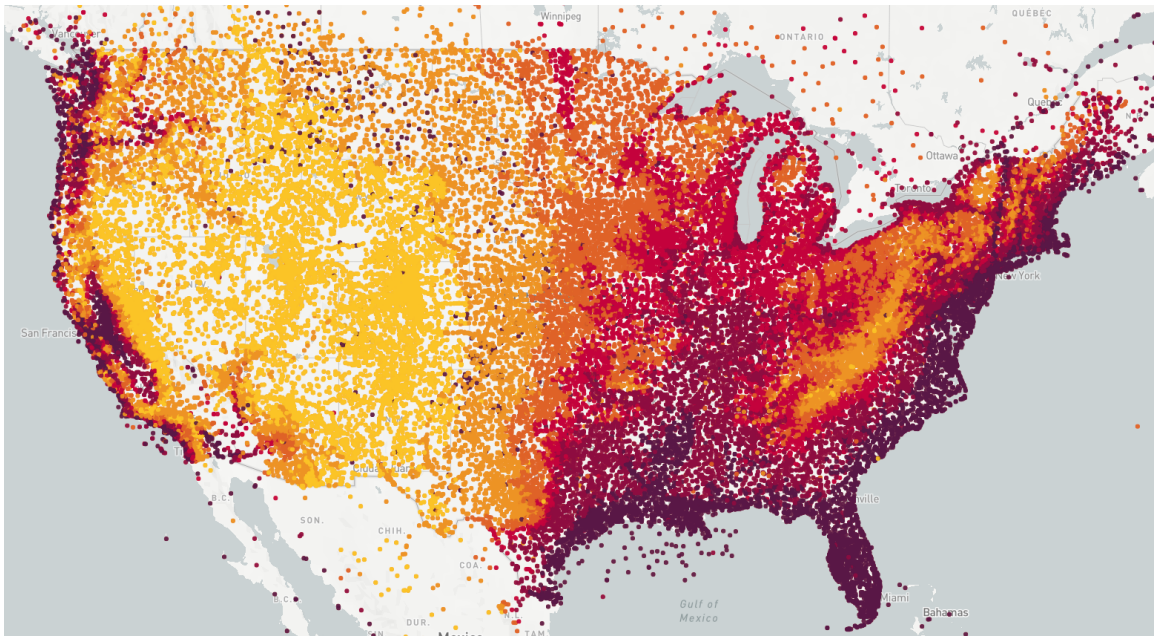
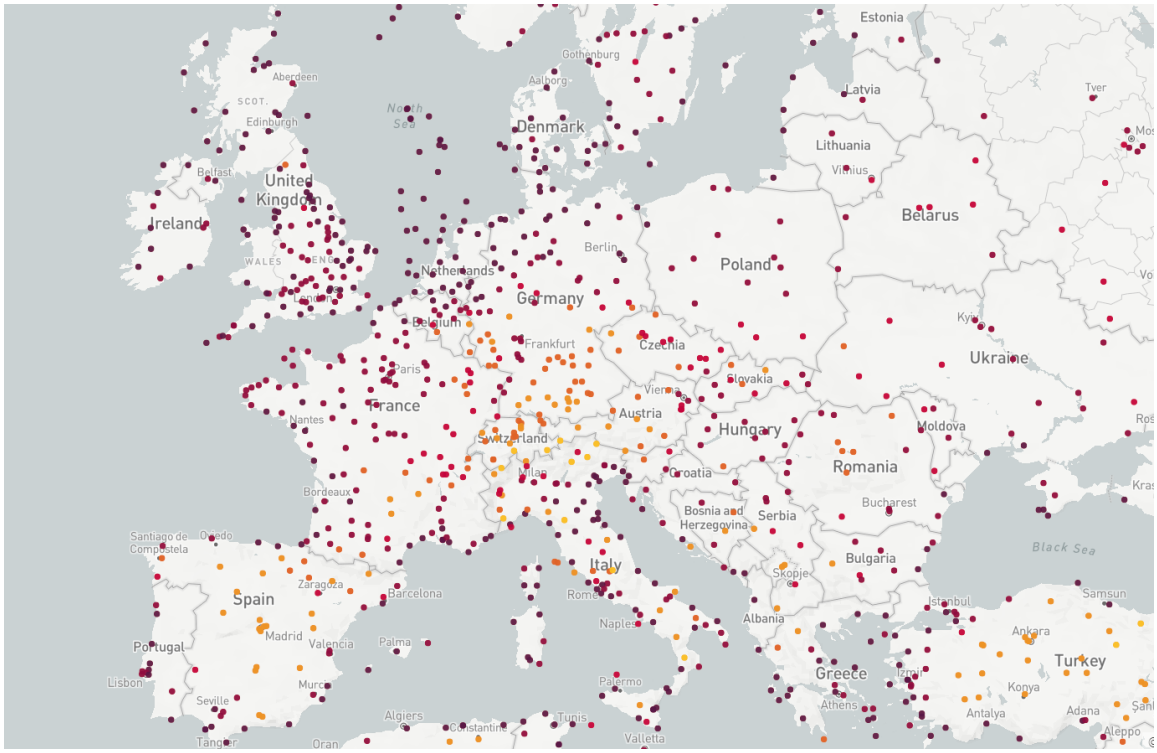


Figure 4.1.: Visualization of all METAR providing weather stations that are available from the IEM service by the University of Iowa.



(a)



(b)

Figure 4.2.: Visualization of METAR providing weather stations that are available from the IEM service by the University of Iowa. The stations are shown on the same level of magnification for (a) the USA and (b) Europe.

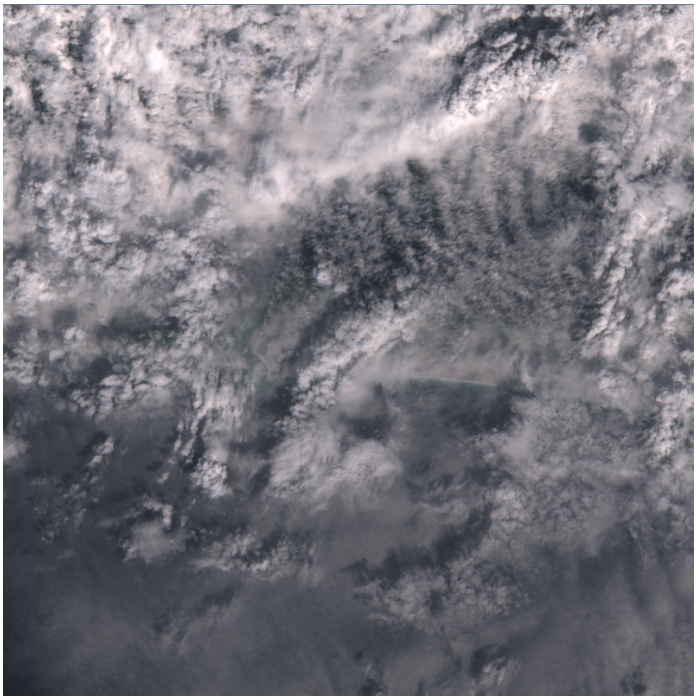


Figure 4.3.: A satellite image over a beach in California that shows the effect of overlapping clouds on total cloudiness.

in recent years, it provides a solid basis for comparisons between climatic conditions of different US states regardless. Not all criteria will be listed here, instead an interpretation is given for the criteria of selected individual states. The full table of codes and meanings is presented in the appendix in Table A.1.

Florida The climatic region of Florida is interesting because of the tropical climatic conditions at the most southern part of the peninsula that are not found anywhere else in the United States of America. The largest part of Florida has a temperate climate, experiences hot summers and no dry seasons. The winters in Florida are generally warm. Accordingly, it sees large amounts of precipitation throughout the year, with especially high concentrations from June to September. Historically, there is a moderate probability (up to 4%) for severe weather events in summer, mostly in the north. The terrain of Florida is relatively flat at a low elevation and contains many bodies of water. Furthermore, Florida is characterized by its sandy beaches and coastal marshes. It is affected by the Gulf of Mexico in its west and the Atlantic Ocean with the Gulf Stream on its eastern side, establishing warm water temperatures.

California There are various climatic regions present in California, making it a great addition to create a compact dataset. The south-east is mostly dry desert, with both hotter and colder temperatures. The west coast of California is temperate with dry, and warm to hot summers. In contrast, there is usually more precipitation present in the time from November to March. The area around the center also includes some steppe, while the

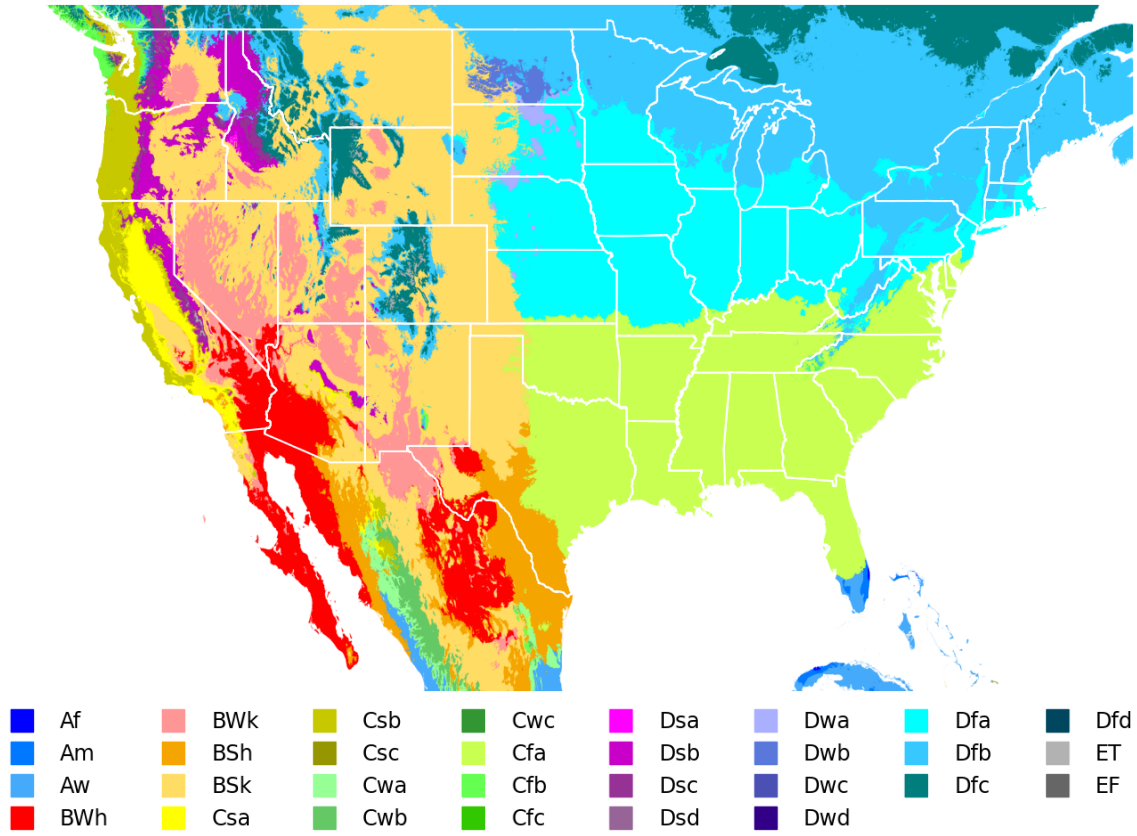


Figure 4.4.: Köppen-Geiger classification for selected regions of the U.S.A. with high data density.

north-east regions are cold and experience dry summers. While the historical occurrence probability of severe weather events is very low, the region often faces wildfires in the summer. California's topography is interesting, since changes in landscape are relatively close. On one hand, there is the Sierra Nevada mountain range, with its highest point at 4,421 meters elevation. On the other hand, there is the shoreline that exhibits both rocky areas and cliffs, as well as sandy beaches. The region is affected by the Pacific Ocean and the cold California Current, flowing southward.

Montana Large regions of Montana are covered by cold, arid steppe. However, in the west there are cold regions without dry seasons. In general, Montana sees relatively little precipitation over the year. Severe weather events happened with a relatively low probability (mostly 1% to 2%) The topography of Montana is coined by its mountain ranges, the Rocky Mountains, and a higher overall elevation in the Eastern Plains with gently rolling terrain. In comparison to the other selected states, Montana does not have any coastal areas.

Washington The region east of Washington's center is cold desert, which is surrounded by cold steppe. Around this area, there are cold areas that undergo dry summers. The west coast of Washington is mainly covered by temperate climate with dry, and warm to hot summers, but also contains areas of temperate climate with warm summers and no dry seasons. Its coast experiences most of its precipitation in the time from October until April. While the coast is mostly flat and lies at a low elevation, the amount of mountainous formations and high elevation areas increase eastward. While the maritime influence of the Pacific Ocean contributes to the formation of clouds, the Rain Shadow Effect, occurring in the mountains, leads to increased precipitation in the western parts of Washington.

Texas The climatic conditions of Texas can mostly be split into three zones. The east is temperate with no dry season and experiences hot summers. The west consists of mostly arid steppe, with colder temperatures in the north and warmer temperatures in the south. Additionally, there are a few desert regions in the west to south-west of Texas with mostly hot temperatures. Texas went through historical severe weather events with a relatively high probability (up to 7%) mostly focused on the north. This is also reflected in Texas being part of Tornado Alley, which is also an area of high frequency of severe weather and supercell thunderstorms, aside from the eponymous tornados. The elevation distribution of Texas starts lower in the south-east and increases towards the north-west. In comparison to California or Washington that also contain both low and high elevations, Texas exhibits more level ground with a small inclination that gradually accumulates to the north-west. Similarly to Montana, the state covers a portion of the Great Planes that has grasslands and a gently rolling terrain. The south of Texas possesses sandy beaches connected to the Gulf of Mexico.

4.1.5. Future Compatibility

During the writing of this thesis, the Copernicus Data Hub announced that the system is going to be replaced by the new Copernicus Data Space Ecosystem [ESAa] as of November 2023. Until the end of June, provision of data is continued normally, while it is reduced from July until September. API access is first made available in the month of April and the full extent of functionality and data is available in July, which is why the services developed for this thesis still uses the API that is associated with the Copernicus Data Hub. Due to the microservice architecture, this poses only a minor inconvenience, as the communication to the new API of the Copernicus Data Space Ecosystem can be developed without affecting other parts of the system. The service itself is also structured in a way that separates scheduling, communication, and processing to ensure a smooth migration to the new API.

4.2. Dataset Preparation

Following the data collection workflow of section 4.1, individual preprocessed data is retrieved for multiple areas of interest with different geological and weather conditions. Before generating a dataset, the local availability of satellite imagery needs to be examined. Even though it is feasible to restrict the data and metadata to local availability before beginning preprocessing, this would exclude the option to add data later on using LTA based retrieval. Since LTA retrieval is included by design, both data and metadata are filtered to only include local Sentinel-2 products after loading, introducing a small overhead. To improve the accuracy of the METAR observations, the included time delta in minutes in relation to the time of observation for the satellite image is used to find the smallest absolute deviation between observation times. In preparation for machine learning, overlapping image bounds of different weather stations need special care, because they have partially identical geography that might cause the machine learning program to learn those patterns and focus on the ground rather than the sky. The significance of this problem is imminent if the learned ground based patterns from the training dataset would help the model achieve better results during the validation phase, but would not generalize well to other geographical regions. An approach to alleviate this problem is to group overlapping regions together into clusters and enforcing the restriction that data associated with each of these clusters can only be distributed to the same dataset. The clustered data is then distributed to the datasets with the goal to achieve a certain 3-way split (e.g., 80-10-10). As a next step, the satellite imagery is preprocessed using band-wise mathematical expressions to fill the outgoing 3-band (RGB) image files, combining and compressing available information. Finally, the labels and additional metadata that are relevant to machine learning are stored together with the preprocessed images in a new directory.

4.2.1. Cluster Analysis for Weather Stations

A common approach to clustering data points that comes to mind is the Density-based spatial clustering of applications with noise (DBSCAN) algorithm by Ester et al. [Est+96]. It uses the two parameters ϵ and **minPts** to add points to a cluster, if they are close to

many other points of that cluster. ϵ refers to the maximum distance between two points that would classify them as neighbors, while **minPts** defines the minimum amount of points required for a cluster to form. DBSCAN then further distinguishes between **core** points, **(directly) reachable** points and **outliers**. Considering a point **A** and a sphere of radius ϵ around it, the point is a

- **core** point, if there are at least **minPts** points in the sphere (including point **A**)
- **directly reachable** point from a **Core** point **B**, if that point lies in the sphere of point **A**
- **reachable** point from a **Core** point **B**, if there is a path $P_0, \ldots, P_N$ with $P_0 = \mathbf{B}$ and $P_N = \mathbf{A}$, where each point P_{i+1} is **directly reachable** from point P_i, implying that all points of the path except **A** have to be **core** points
- **outlier**, if no **core** point lies in the sphere of point **A**

While this density based technique provides a good foundation for clustering that is applicable for data mining in many scenarios with a runtime complexity of just $O(n \log n)$ and a space requirement of $O(n)$, its introduction would come at the cost of accuracy. Since the most important factor is the overlap of square image bounds, setting the search radius ϵ too small would not detect all intersections. On the other hand, if the radius is too large, weather stations whose image bounds do not intersect would still be clustered together. An example of such a scenario is shown in Figure 4.5. Various radii ϵ interact differently with neighboring weather stations **A**, **B** and **C**. The rectangles around the weather stations represent their image bounds. All radii originate from weather station **A**, as it is chosen as the center in this example. The image bounds overlap only for stations **A** and **B**. The illustration shows in (a) that when the radius is too small, the overlap between stations **A** and **B** is not recognized and clustering does not occur. In contrast, when the radius is too large, as is the case in (b), station **C** is falsely clustered together with **A** even though no overlap is present. (c) depicts a thought experiment: Even considering an extension of the DBSCAN algorithm that facilitates intersecting the radius with the image bounds to recognize overlapping regions, it would not be possible to only detect the overlap of stations **A** and **B** without also falsely detecting overlap between **A** and **C**, due to the rectangular shape of the image bounds and the specific station positions in this example. The loss in accuracy results from falsely clustering together stations that have no overlap and can be seen as bleeding to nearby not overlapping stations.

To counter this accuracy problem, a different shape based approach is used. Since metadata of the weather stations is stored after data collection, the identifiers and geospatial positions can be retrieved and utilized for further processing. A key approach is to generate the shapes corresponding to the image bounds of each weather station. With these geospatial shapes in tabular data and the weather station identifier as keys, the intersections are calculated using a table join. When only looking at the remaining primary and foreign keys, this is interpretable as an adjacency list. Viewing this adjacency list in the form of a graph yields the desired clustering, where nodes represent weather

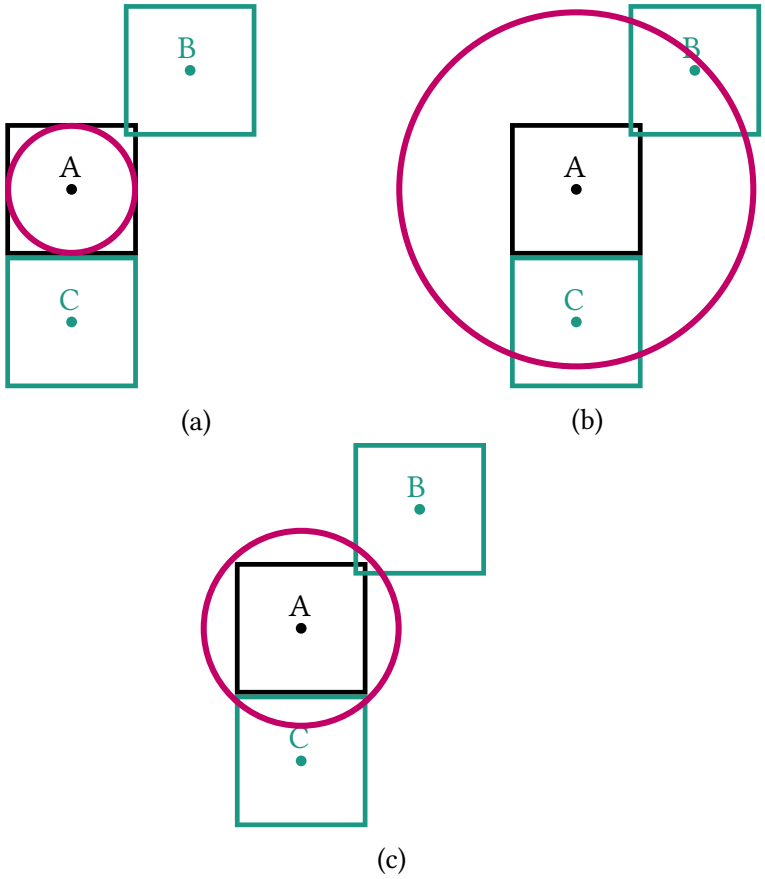


Figure 4.5.: Visualization of the implications of using different radii ϵ with image bounds around the positions of weather station **A**, **B** and **C**.

stations and edges represent intersections of image bounds. A visualization of such a graph can be seen in Figure 4.6 for the state of California. Square borders represent the image boundaries around weather stations, and squares of the same color signify a single cluster (gray squares excluded). The center points of overlapping image bounds are connected by edges to form a network. The network is represented by the black lines that connect squares of the same cluster. All nodes that are reachable from a starting node – including itself – are contained in a single cluster. This provides an accurate result without bleeding, compared to DBSCAN.

After deriving all clusters, weights are assigned based on the accumulated amount of METARs provided by each weather station that is assigned to that cluster. The amount of METARs is restricted to only account for one METAR per weather station and satellite image, reflecting the desire to only use the METAR that has the smallest difference in time of observation between METAR and satellite image. These weights build the foundation for distributing the data to the datasets.

4.2.2. Dataset Creation

To build a dataset with a desired split (e.g., 80-10-10) for training, validation, and testing, the weighted clusters are distributed to corresponding datasets. This equates to solving a partition problem, for which a variant of greedy number partitioning is used. Instead of putting each cluster into the dataset whose current quantity is the smallest, the dataset with the largest missing quantity is chosen in algorithm 2. The new datasets are created with a target quantity equal to their split percentage of the total available data quantity. Each cluster is sequentially added to the dataset with the largest missing quantity. Sorting the clusters according to their weight first, increases the runtime from $O(n)$ to $O(n \log n)$. The approximation improves as well, since clusters with larger weights are distributed earlier, leaving the small weights for the final adjustment.

Since there is more data available from regions with more weather stations and a higher observational frequency than others, distributing data independently of area all at once might cause sparse data to fall into a single dataset. And if sparse data with special regional properties like ice is only present in the test dataset, the machine learning model will not be able to accurately learn these patterns. To ensure that the created datasets all contain information about the different underlying geospatial regions with their individual properties in similar amounts, the data distribution is applied to all areas of interest on their own. Finally, the resulting per region datasets are joined, resulting in the final training, validation and test datasets as shown in Figure 4.7. By utilizing the greedy algorithm algorithm 2 for a given dataset of 7783 labeled images, the target split of 80-10-10 is approximated as seen in Table 4.1.

To reduce the size of the final dataset, the spectral band measurements from the Sentinel-2 satellites residing in individual are combined and stored in a single outgoing file. All images are rescaled to a customizable but per dataset fixed resolution to enable using them

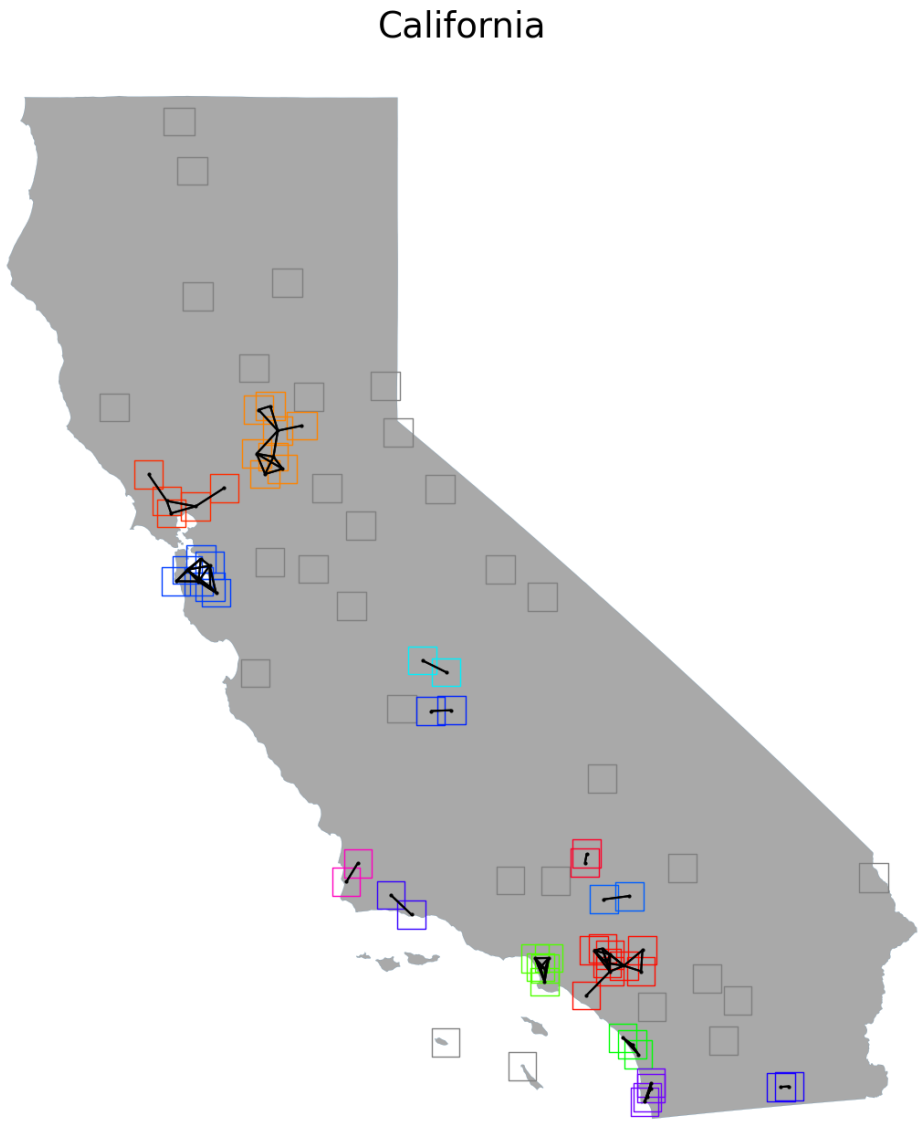


Figure 4.6.: A visualization of the network that is build from the tabular join adjacency list for the example of California.

Algorithm 2: Greedy Number Partitioning – Largest Missing Quantity

```
Data: split, data
Result: datasets
datasets ← List of Dataset
for percentage ∈ split do
    datasets.append(Dataset(percentage · data.quantity))
for cluster ∈ data do
    dataset ← getDatasetWithMostFreeSpace(datasets)
    dataset.add(cluster)
return datasets
```

Function getDatasetWithMostFreeSpace(datasets)

```
maxQuantity ← 0
maxDataset ← None
for dataset ∈ datasets do
    if dataset.quantity > maxQuantity then
        maxQuantity ← dataset.quantity
        maxDataset ← dataset
return maxDataset
```

Function getMissingQuantity(dataset)

```
return dataset.targetQuantity − dataset.currentQuantity
```

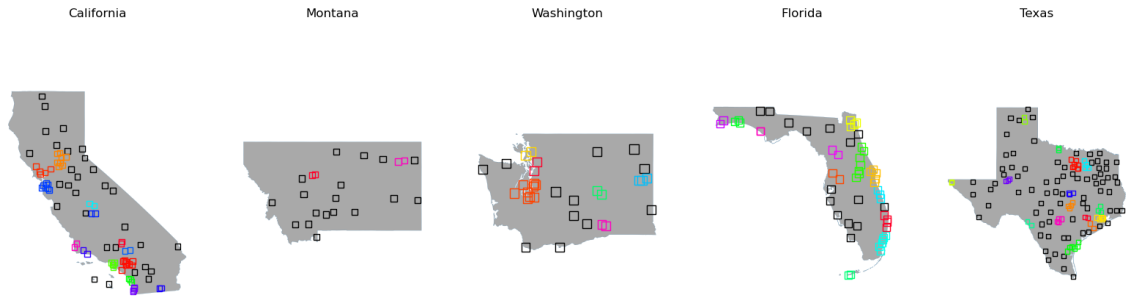


Figure 4.7.: The result of distributing data using the greedy algorithm algorithm 2 and handling different regions independently.

	Target distribution		Actual distribution	
	Percentage	Quantity	Percentage	Quantity
Training	80.00	6225	79.97	6224
Validation	10.00	779	10.06	783
Test	10.00	779	9.97	776

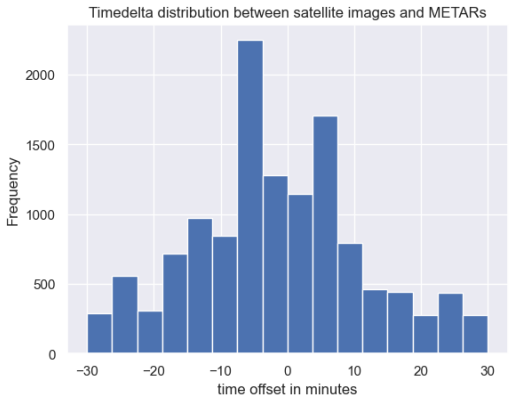
Table 4.1.: Target and actual distribution for the distribution of data into three datasets using the greedy algorithm algorithm 2. There are 7783 data units distributed in total.

in the PyTorch machine learning framework. For future compatibility and visualization purposes, geospatial data is copied as well. The bands are combined in such a way that each band acts as a matrix on which mathematical expressions can be evaluated. An intuitive approach is to start with just bands B4, B3, and B2 that represent the red, green, and blue values of a True Color Image (TCI). All bands are scaled according to the specification of Sentinel-2 Level-1C images, by dividing them by their quantification value that is set to 10,000 by default and accordingly converted to floating point values. Finally, the resulting combined images, and labels extracted from the METARs are stored in the dataset directory.

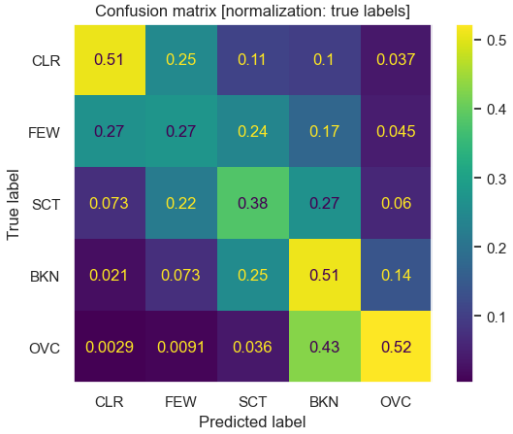
4.2.3. Final Dataset

For the dataset used in the machine learning process, a split of 80-10-10 for the training, validation, and test datasets is chosen to retain most of the data for training and still have an useable amount of data for validation and testing from the total of around 12800 images. The label distributions of the dataset split are compared to ensure similarity. A problem of the dataset is its imbalance, as shown in Figure 4.9c for the METAR cloud coverage classes shown in Table 2.2. There are significantly more labels equal to CLR than any other class on its own. While there are around 5,300 labels in the CLR class, the FEW class consists of just around 920 labels. It is necessary to address this imbalance in the dataset to ensure that the model does not simply try to predict the majority classes. Figure 4.8a shows the distribution of time-differences between satellite images and METARs in minutes. The anchor point in time is chosen to be the satellite image, since there are more METARs available for a single satellite image and because the satellite image acts as the ground truth of cloud coverage, which even the METAR labels might deviate from. The majority of METARs are given in a time-window of around +/- 15 minutes from the observation of the satellite image. Due to the limitations imposed on data collection from the Copernicus Open Access Hub, the data is collected from January 2023 until June 2023.

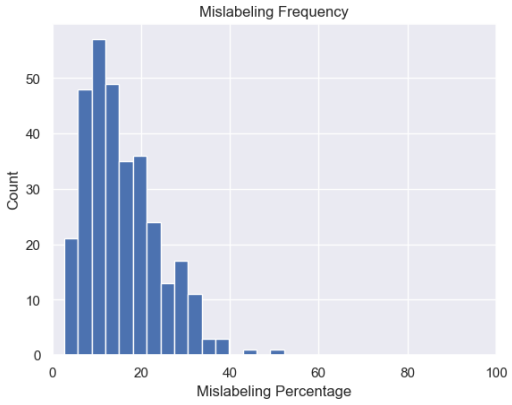
To provide an estimate of the data quality as well as a way to improve testing of developed models, the full dataset of around 12800 images is also labeled manually. While this is done with best practices and conventions regarding the observation of cloud coverage and the METAR format in mind, making mistakes can not be avoided completely. Nevertheless, these mistakes should almost always be constraint to the directly neighboring classes, establishing clarity for cases in which the METARs provide information that does not match the satellite images. Aside from incorrect data sent by the weather stations, the occurrence of mismatches between METARs and satellite images is possible due to the time window of +/- 30 minutes, in which METARs are considered to be in range of the satellite image. The selection of the time-window is a tuneable parameter that also impacts the amount of data that is available for a selected region, whose negative impacts in the form of mismatches are reduced by only selecting the most recent METAR to add to the dataset. The confusion matrix in Figure 4.8b indicates how well the METAR classes fit to the satellite images. Here, the true labels are equivalent to the METAR cloud coverage information, whereas the predicted labels represent the manually labeled satellite images. Even considering that a certain mismatch occurred between neighboring classes, the matrix shows that there are some edge cases like OVC being labeled CLR or vice versa. Nonetheless, the labels are in



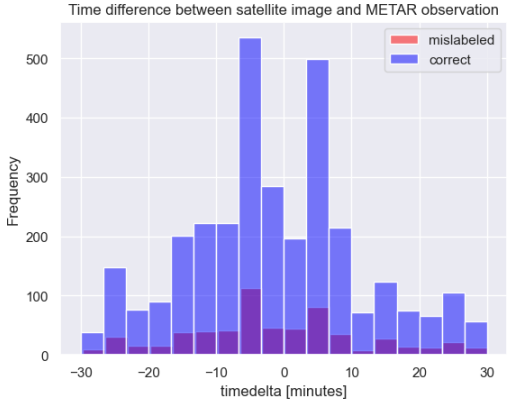
(a) The distribution of time-differences between the satellite images and the METARs of the final dataset.



(b) Confusion matrix of METAR classes as true labels (Y-axis) put against manually predicted labels (X-axis).

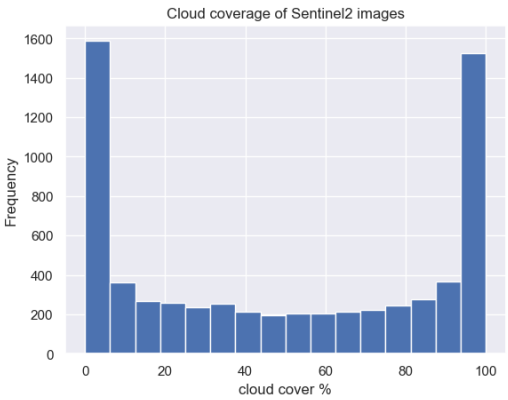


(c) Frequency of mismatches between METARs and satellite images on a per weather station basis.

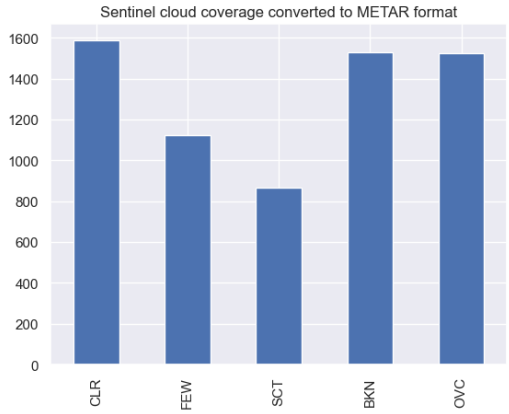


(d) Overlapping histograms of both correct labels and mismatches between METARs and satellite images.

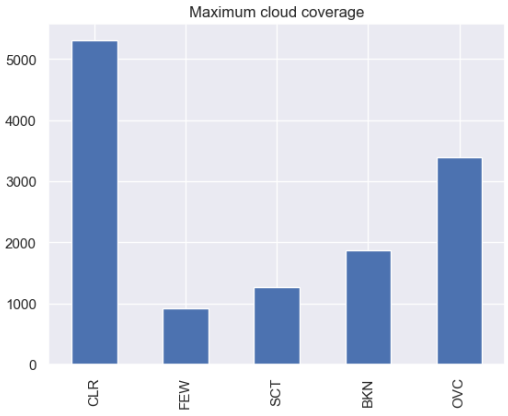
Figure 4.8.: General statistic information regarding the final dataset.



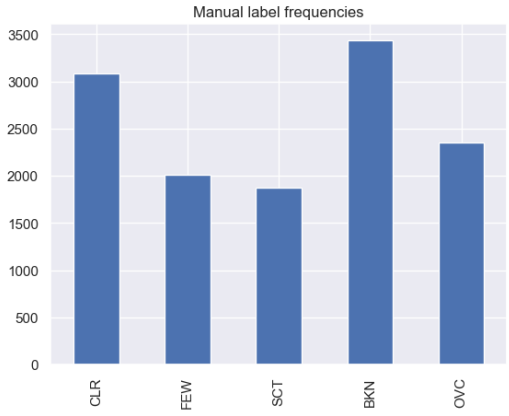
(a) The distribution of cloud coverage (in percent) of the complete satellite images of which only parts are used.



(b) The distribution of cloud coverage (as METAR classes) of the complete satellite images of which only parts are used.



(c) The maximum cloud coverage distribution of the final dataset.



(d) The manually labeled cloud coverage distribution of the final dataset.

Figure 4.9.: Cloud coverage distributions of the final dataset and its data sources.

similar classes overall, making model training feasible, as patterns and relationships can still possibly be derived. When only looking for mismatches that are further away than the directly neighboring classes, the average mismatch frequency per station is around 15.9%, as exhibited by Figure 4.8c. Coming back to the previously shown time delta distribution, Figure 4.8d presents a histogram of mismatches and correct labels in relation to each other in the form of a histogram over the selected time window of +/- 30 minutes, showing that most of the labels either match directly or fall into a neighboring class. Furthermore, the distribution highlights that the amount of mismatches are independent of the selected time window. Figure 4.9a illustrates the distribution of cloud coverage in percent for the full-resolution source images, from which only parts are extracted for the dataset. Figure 4.9b builds up on this information by selecting what percentages fall into which METAR class. This grants a new perspective on the distinction between the distribution of automatic METAR labels (Figure 4.9c) and manual labels (Figure 4.9d). The distribution of manual labels fits the expected distribution derived from the satellite images significantly better than the automatic labels of the METARs, which on one hand signifies the accuracy of manual labeling and on the other hand shows imperfections in automatic labeling. To better reflect the capabilities of the model developed in this thesis, confusion matrices are provided for both METAR labels and manual labels as true labels.

4.3. Machine Learning

The machine learning model developed in this thesis aims to predict the cloud coverage categories according to the specifications of the METAR format that is defined by the WMO [WMO22] and the ICAO [Int21], and implemented by the United States NOAA [ONa]. It strives to correctly classify cloud coverage into the categories show in Table 2.2. The challenge of choosing an initial machine learning model is coined by the requirements to work on both smaller and larger datasets effectively and efficiently, while also providing state-of-the-art results for image classification tasks. Furthermore, the training process should be efficient, having only a limited number of parameters to train in a shorter amount of time. After considerations of other models like the Swin TransformerV2 by [Liu+21b] [Liu+21a], EfficientNetV2 by Tan et al. [TL19] [TL21] is chosen for meeting these requirements as its training is faster compared to other state-of-the-art models with the addition of being up to 6.8 times smaller than these models. The model is characterized by its fast training speed and low parameter count. During the initial phase, computation is done on the bwUniCluster 2.0 using a JupyterLab based working environment. The model and data is then moved to the HDF-ML cluster after initial experimentation, where Simple Linux Utility for Resource Management (SLURM) jobs run in the background to train the model.

4.3.1. Transfer Learning

There are multiple reasons why transfer learning is an applicable technique for cloud coverage classification, such as only leveraging existing knowledge. While there are approximately 12800 labeled satellite images that cover the aerodrome of the weather

stations available in the dataset, pre-trained models can still provide additional insights. They are usually trained on more diverse visual patterns compared to the satellite images of the dataset used in this thesis, which are also taken from the same visual distance, further reducing the variety of patterns. Since classification of cloud coverage is a task that involves complex visual patterns and textures, making it challenging to capture the desired characteristics. Examples of these aspects are varying cloud shapes that are most prominent between different types of clouds, like fluffy cumulus clouds and thin, stretched cirrus clouds. These variations make it difficult to precisely define boundaries and learn consistent cloud patterns. Likewise, clouds also tend to overlap, creating even more complex visual structures in the process. This is where transfer learning is able to support the model by using the pre-trained extraction capabilities for high-level features, in addition to the lower layers that are already capable of extracting general information on a smaller scale. On the other hand, feature extraction can still be fine-tuned by unfreezing and training layers of the pre-trained model before the last classification layer. Aside from image complexity itself, transfer learning also addresses generalization and robustness. When trained on an appropriate dataset, the pre-trained model may help stabilize the training process with regard to changes in time – relating to shadows and lighting – and locations with different topography. Finally, training a model that is based on a pre-trained model using transfer learning allows to rapidly develop new machine learning models, as computation times are significantly reduced due to the basic knowledge that is already available from a different domain, also improving convergence. The chosen base model EfficientNetV2 by Tan et al. [TL21] is trained to detect 1000 classes using the Imagenet-1k dataset, incorporating a total of around 1.43 million images. This provides a solid foundation for transfer learning that helps accelerate model development and achieve better results. The model also shows great results for transfer learning in other scenarios, achieving a better accuracy with an average parameter reduction factor of 4.7 in comparison to other models such as ResNet by He et al. [He+15] and DenseNet by Huang et al. [HLW16].

4.3.2. Model

EfficientNetV2 uses a convolutional neural network (see subsection 2.2.8) structure. One of the central aspects that are already introduced in EfficientNet by Tan et al. [TL19] is compound scaling, which facilitates uniform scaling of network width, depth, and height. This scaling builds on the intuition of needing more layers to capture finer patterns in a larger image. Since the pretrained parameters of the EfficentNetV2 model only account for RGB images, the images of the dataset are restricted to using exactly three outgoing bands as well. The initial spectral band to image channel mapping is set to R = B4, G = B3, and B = B2, representing TCIs that are also used in the ImageNet training processes of the pretrained parameters. If not specified otherwise, it can be assumed that model performance is evaluated on images of size 300×300 pixels, to still capture relevant details while reducing the size of the dataset. To let the model learn patterns and relationships from the satellite images that it does not already know of, model parameters are unfrozen. To measure model performance on this imbalanced dataset, the F1-score is used, as explained in subsection 2.2.10. Since the dataset is imbalanced, confusion matrices that are shown in

the following all use the true-label normalization that is explained in subsection 2.2.11. Confusion matrices are always created from previously unseen data of the training dataset.

4.3.3. Direct Neighbor Accumulation

The technique refers to the summation of neighboring performance values of a confusion matrix. The reason for the introduction of this method is the potential for mismatches of manual labels to one of the neighboring classes, due to human error. By allowing a mismatch to the neighboring classes, the results can be aggregated. This presents a clearer view on information regarding mismatches that are very likely not due to human error, because their difference is greater than one class. For each row of the true-normalized confusion matrix (see subsection 2.2.11) the neighboring values are summed as shown in Equation 4.1, where DNA_i represents the direct neighbor accumulation value for the i-th row, and $CM_{i,k}$ symbolizes the entry in the i-th row and k-th column of the confusion matrix CM. The column index k is adjusted over the iteration process to include the entries left and right of the primary diagonal. Edge cases are treated by constraining the column index k to the dimensionality of the confusion matrix with n classes.

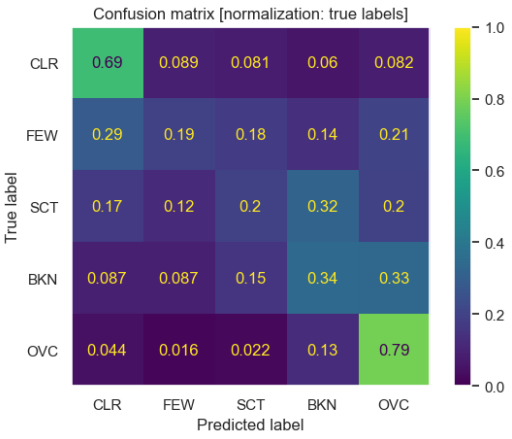
$$DNA_i = \sum_{k=\max(i-1,0)}^{\min(i+1,n-1)} CM_{i,k} \tag{4.1}$$

4.3.4. Classification and Regression

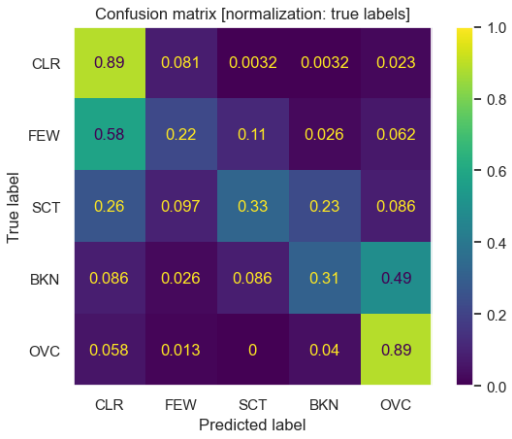
An important decision for the following model architecture is the choice between classification and regression techniques. The techniques are compared here to find out which one delivers the best performance. While classification techniques regard different classes as exclusive and unrelated, regression techniques focus on the presence of a relationship in the data (e.g., CLR is less cloudy than OVC). Even though multi-class classification is often used when comparing images of different classes, regression can potentially better understand the ordinal relationship between the classes that is expressed as being least cloudy to most cloudy: CLR, FEW, SCT, BKN, OVC.

Classification The multi-class classification implementation uses a fully connected layer with five neurons, representing the five METAR classes, as its output or classification layer. It is trained using a Cross-Entropy loss function, and the model prediction is determined by the maximum value of the output layer's tensor. The resulting confusion matrix is shown in Figure 4.10. The model performs good for the most distinct edge-classes CLR and OVC, but fails to achieve an acceptable performance on the inner classes FEW, SCT, and possibly BKN. A potential pitfall of this approach is treating classes independently of each other and not learning their ordinal relationship, resulting in less accurate predictions for intermediate classes and deviation from the expected class by more than one class-rank.

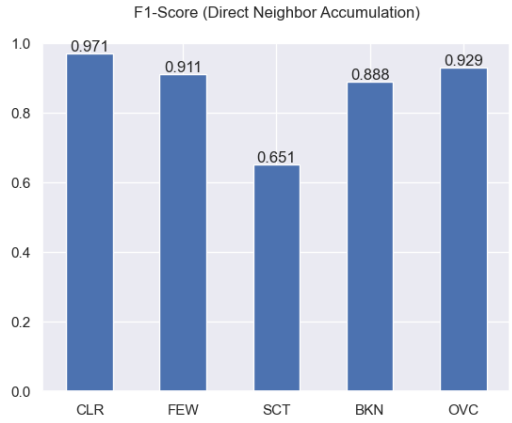
One-Neuron Regression One way to implement regression is to reduce the number of neurons in the fully connected output layer to just one and constraining its predictions



(a) Confusion matrix using METARs as true labels.

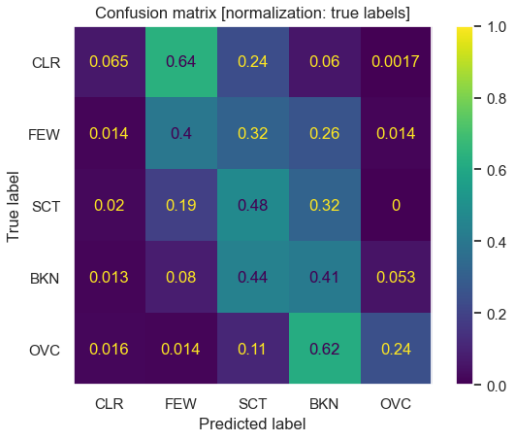


(b) Confusion matrix using manual labels as true labels.

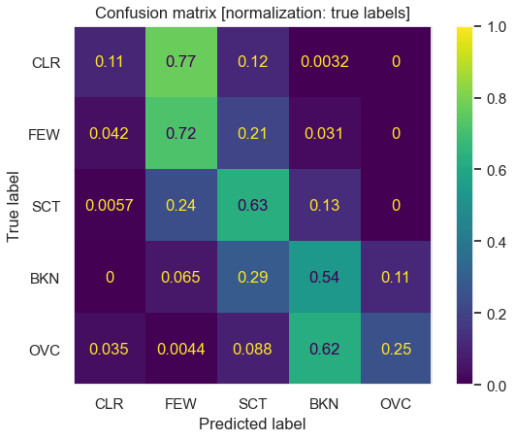


(c) Direct neigbor accumulation for confusion matrix using manual labels as true labels.

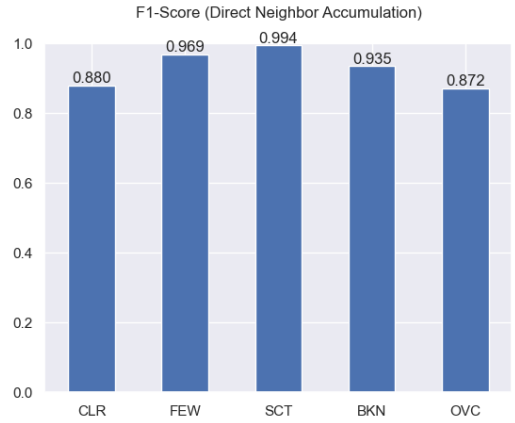
Figure 4.10.: Evaluation of the classification model experiment.



(a) Confusion matrix using METARs as true labels.



(b) Confusion matrix using manual labels as true labels.



(c) Direct neigbor accumulation for confusion matrix using manual labels as true labels.

Figure 4.11.: Evaluation of the one-neuron regression model experiment.

to the range from 0 to 1, using a sigmoid function at the end. Training is done using a Mean-Squared-Error loss function, while the predicted class is derived from distributing thresholds in between the interval [0, 1] of the model output, so that 0 corresponds to CLR and 1 stands for OVC. The corresponding confusion matrix is shown in Figure 4.11. While the performance of the model is good for intermediate classes like FEW, SCT, and BKN, the identification performance of the CLR and OVC classes is not acceptable, as the images are almost always misclassified as their corresponding neighbor class. This disadvantage possibly arises from the use of the sigmoid function, requiring progressively absolute-larger input values to get closer to the edges of the interval [0, 1], reducing the probability of results falling into the edge classes CLR and OVC.

Ordinal Regression Since sigmoid functions perform well for binary classification tasks, a new approach is used with this aspect in mind. It builds on five neurons (equal to the

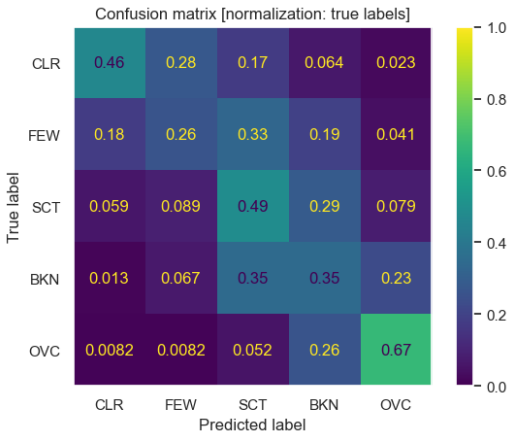
number of classes) in the fully connected output layer, similar to multi-class classification, but uses a sigmoid function for the outputs, similar to one-neuron regression. The sigmoid function restrains the outputs to the range of 0 to 1. The Mean-Square-Error loss function is applied in this context as well, while the threshold approach is replaced with another technique developed by Cheng et al. [CWP08]. Given the output tensor in Equation 4.2 for the classes CLR, FEW, SCT, BKN, and OVC in this order from left to right, a new vector is derived based on the inner values. Initially, it is checked, whether the values are greater than 0.5. If that is the case, they are set to one, and if not, they are set to zero. In the given example, this results in the tensor in Equation 4.3. By calculating the cumulative product, all ones after the first zero are also set to zero. Finally, summing this tensor yields an index of the corresponding predicted class. This guarantees that the Mean-Squared-Error difference between classes like CLR and OVC, that are far apart, is higher (4 class-ranks) than the difference of classes like FEW and SCT (1 class-rank), which are closer. The resulting confusion matrix is visualized in Figure 4.12. The matrix shows mediocre performance for the FEW and BKN classes, while delivering good results for the CLR, SCT, and OVC classes. Furthermore, the model shows almost no deviation from the diagonal, further highlighting its capabilities for this task, which is why it is used as the reference model in the following experiments.

$$[0.7,\ 0.6,\ 0.4,\ 0.8,\ 0.3] \tag{4.2}$$

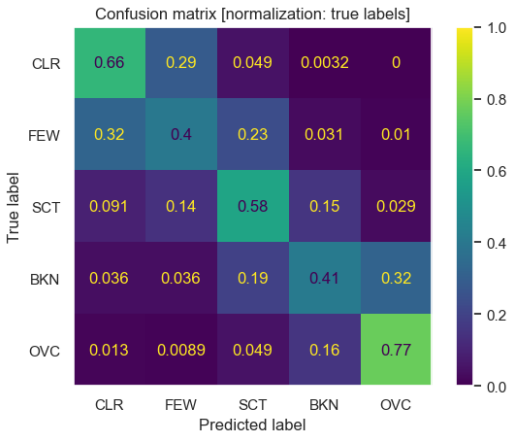
$$[1,\ 1,\ 0,\ 1,\ 0] \tag{4.3}$$

4.3.5. Manual labels for training

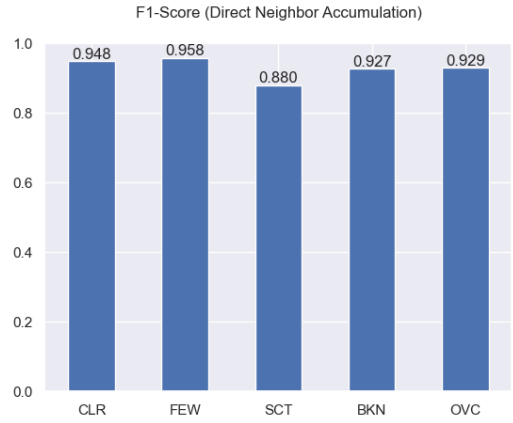
The results of the experiments show the significance of having accurate labels for performance comparisons. The mismatches between METARs and satellite images is especially apparent when comparing the confusion matrices of the ordinal regression model experiment in Figure 4.12. This raises the question of how good the model would perform, if given the manual labels instead of the METAR labels. The results of this training for the ordinal regression model is shown in Figure 4.13. Although Figure 4.13 (a) featuring METAR labels as true labels is similar to Figure 4.11 (a), showing a higher density of predictions around the primary diagonal of the matrix, Figure 4.13 (b) illustrates how accurate the model is able to predict most classes when given the cloud coverage information that is derived directly from the satellite images. The mediocre performance regarding the CLR class can be partially attributed to lack of distinction capabilities between clouds, snow, and ice in mountainous regions. Additionally, small bright details like buildings in the image might get classified as clouds, ranking it as FEW instead but almost never above that class. Moreover, the direct neighbor accumulation performance shown in Figure 4.13 (c) improves significantly as well, with most values deviating only by one class from the primary diagonal.



(a) Confusion matrix using METARs as true labels.

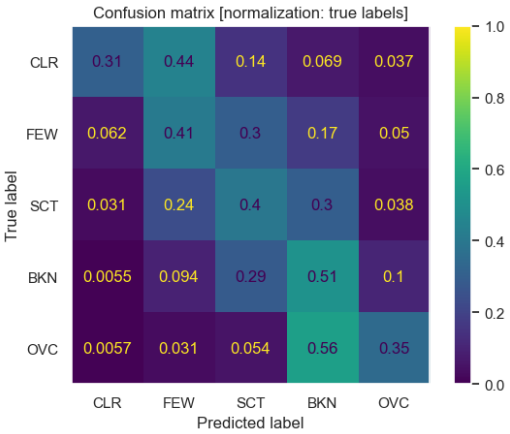


(b) Confusion matrix using manual labels as true labels.

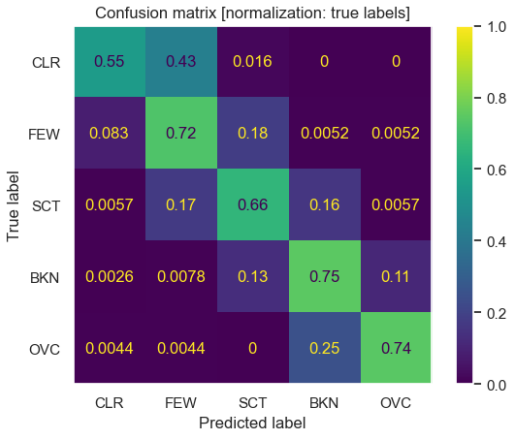


(c) Direct neigbor accumulation for confusion matrix using manual labels as true labels.

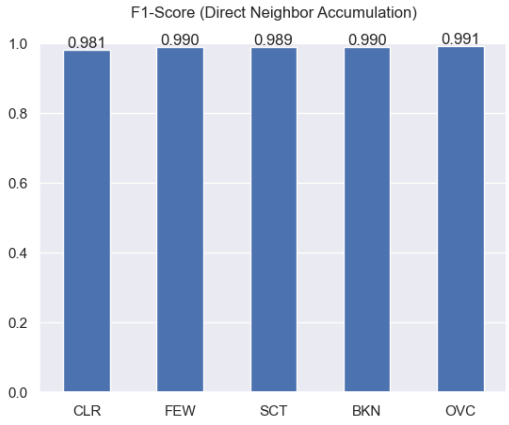
Figure 4.12.: Evaluation of the ordinal regression model experiment.



(a) Confusion matrix using METARs as true labels.



(b) Confusion matrix using manual labels as true labels.



(c) Direct neigbor accumulation for confusion matrix using manual labels as true labels.

Figure 4.13.: Evaluation of ordinal regression model performance when trained on manual labels.

4.3.6. Image Size

A notable parameter of model training is the size of the images, with a focus on width and height, since the number of color channels is locked by the pre-trained model. Since the original images are taken at a resolution of 10 meters per pixel for a radius of 16 kilometers, they are around 3200×3200 pixels large. Since the previous results are created using images that are downscaled to 300×300 pixels, the relation to image size is tested not with the full resolution but with a higher scale of 800×800 pixels. Figure 4.14 presents a relatively similar result compared to evaluation of the model on images of size 300×300 pixels in Figure 4.12. Improvements can be seen for classes CLR and FEW, while the performance regarding SCT, BKN, and OVC has declined. There are other factors of influence that need to be addressed regarding larger image sizes. With this increase in image size, the dataset grows from around 12.8 Gigabytes to around 91.2 Gigabytes, which is around 7.1 times the size of the original dataset. Moreover, the training time increases from around 1.25 hours to around 9 hours – taking around 7.2 times as long. Since the larger images take up more memory of the GPU, the batch-size is reduced from 48 to 6. To ensure that this does not affect the performance of the model, gradient accumulation is used. Gradient accumulation enables training on larger batch sizes than a system would be able to fit into memory. This is done by accumulating gradients over several batches – here 8 batches, since $6 \cdot 8 = 48$ - and then performing the gradient step after the desired number of batches have been processed. In the context of the evaluation results in Figure 4.14 and the discussed downsides that go hand in hand with increasing the image resolution, it is deemed appropriate to remain with using the lower image resolution of 300×300 pixels.

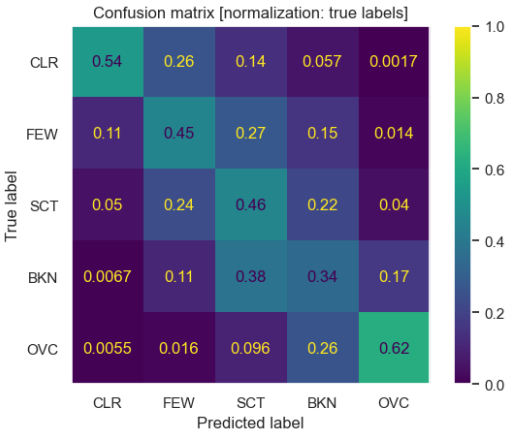
4.3.7. Separating Clouds from Snow

Differentiating clouds from snow is crucial for their detection in satellite images and is examined by many existing cloud cover detection algorithms [Hol+96; CB04; ZW12; Mai+17; Jep+19]. The differentiation is especially important, because clouds as well as snow and ice share similar reflectances in the visible spectrum of light. To counter this problem and separate clouds from snow and ice, the NDSI is used. The value is computed per pixel according to Equation 4.4, where $B3$ represents green light in the visible spectrum at a wavelength of around 560 nm and $B11$ is SWIR light at a wavelength of around 1610 nm as shown in Table 2.1. Then, the complement of the NDSI is computed and multiplied with the red, green, and blue color channels (RGB) like in Equation 4.5. This ensures that the image still resembles colors similar to the original RGB image, which is helpful for knowledge transfer, since the EfficientNetV2 base model is trained on RGB images as well.

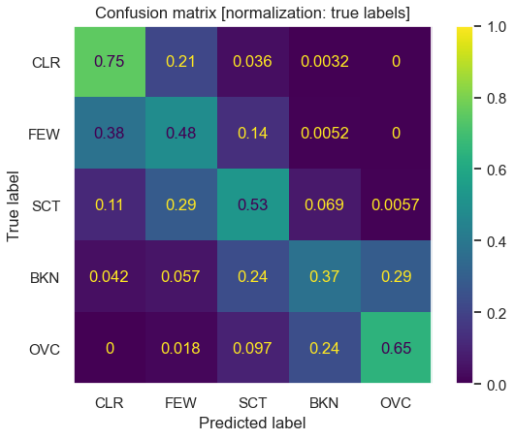
$$\text{NDSI} = \frac{B3 - B11}{B3 + B11} \tag{4.4}$$

$$\text{img} = [R, G, B] \cdot (1 - \text{NDSI}) \tag{4.5}$$

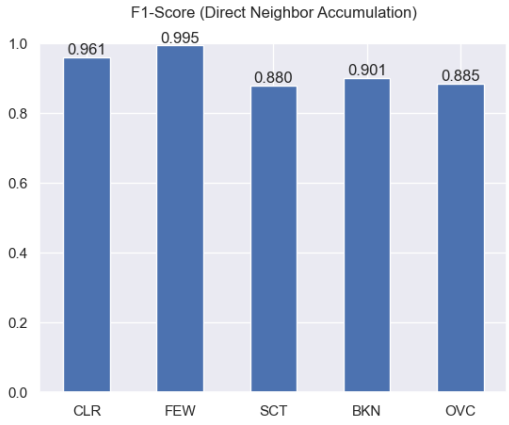
Figure 4.15 presents an RGB image, the NDSI values using an additional infrared band (B11), and the combination of RGB image with the NDSI values according to Equation 4.5. In comparison to the original RGB image, the image using both RGB and NDSI values



(a) Confusion matrix using METARs as true labels.

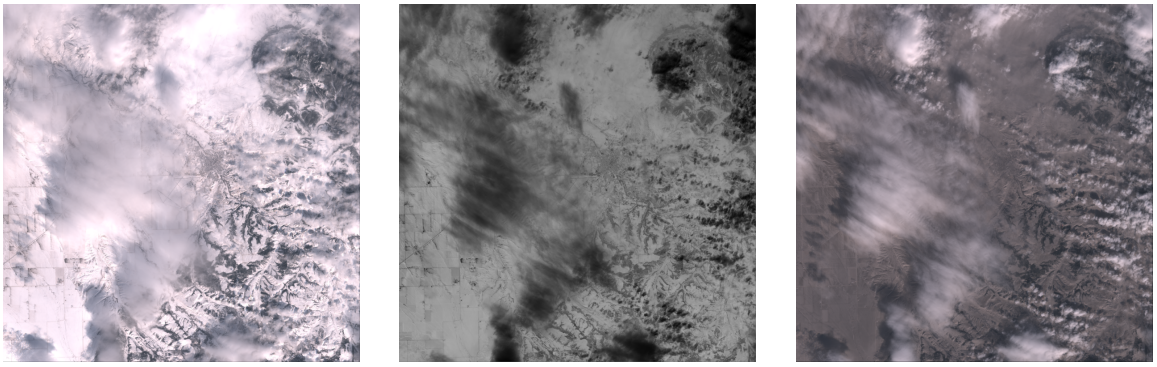


(b) Confusion matrix using manual labels as true labels.



(c) Direct neigbor accumulation for confusion matrix using manual labels as true labels.

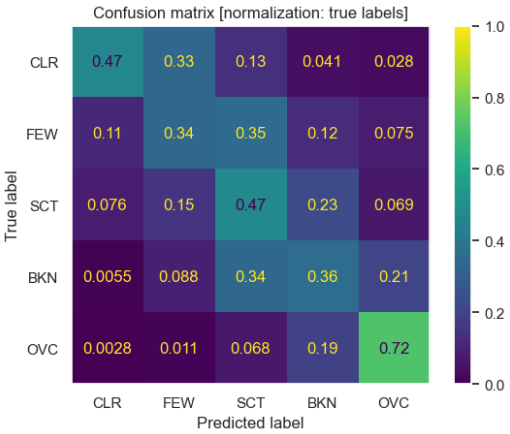
Figure 4.14.: Evaluation of ordinal regression model performance when trained on images of size 800×800 pixels.



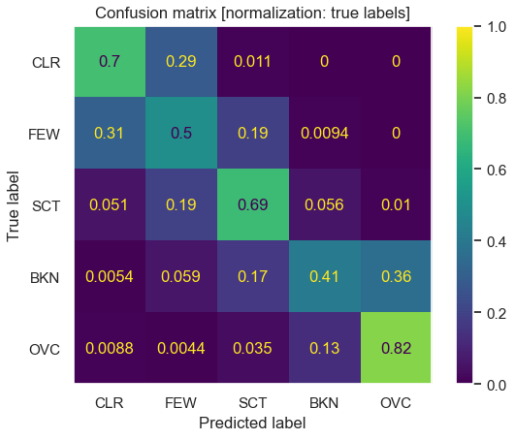
(a) (b) (c)

Figure 4.15.: Visualization of (a) a simple RGB image, (b) the NDSI value of the image, and (c) the RGB image multiplied by the complement of the NDSI

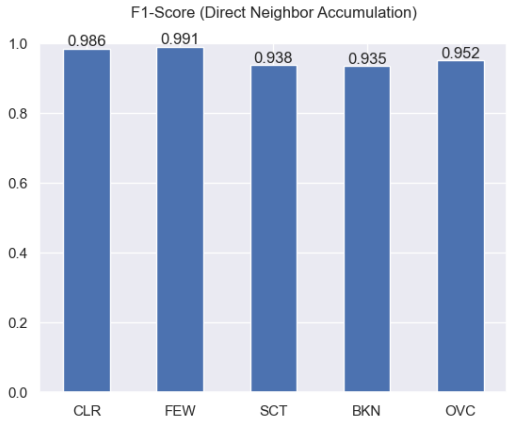
improves the contrast between clouds and snow. The original dataset of images of size 300×300 pixels is recreated with the application of Equation 4.4 and Equation 4.5. Further assessments regarding the separation of clouds from snow and ice and effects on terrain are presented in the appendix in section A.2. Training the machine learning model on this new dataset shows overall improvements in detection accuracy, as illustrated in Figure 4.16 for training on METAR labels. In comparison to the performance on the original RGB image dataset Figure 4.12, the model improves its accuracy on the primary diagonal for most classes by 4% to 11%, while not negatively affecting the performance of any other class. The original dataset restrained the model performance with direct neighbor accumulation to the range of 88.0% to 95.8%, while the new dataset with NDSI values improves to the range of 93.5% to 99.1%.



(a) Confusion matrix using METARs as true labels.



(b) Confusion matrix using manual labels as true labels.



(c) Direct neigbor accumulation for confusion matrix using manual labels as true labels.

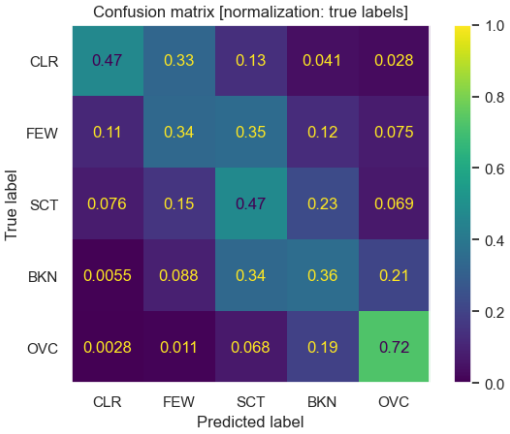
Figure 4.16.: Evaluation of ordinal regression model performance when trained on METAR labels on the NDSI supported dataset.

5. Evaluation

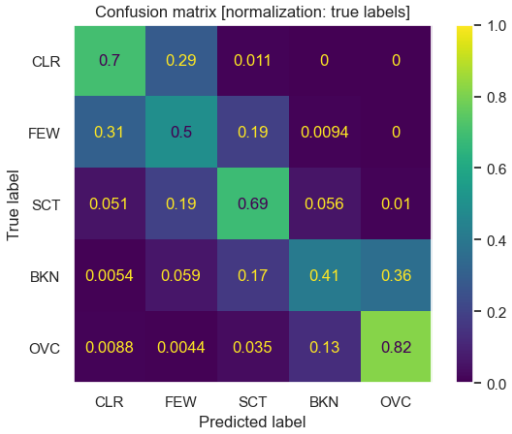
Following the results of the experimentation process, the best results are achieved by the ordinal-regression model on the dataset that utilizes NDSI values for the separation of clouds from snow and ice. The base model used for transfer learning is the small variant of the EfficientNetV2 family, having just 22 million parameters. The MSE loss function is used for training of the model. Both steepest gradient descend (SGD) and the Adam algorithm by Kingma et al. [KB14] are tested for optimization. Adam shows better results and is used with a learning rate of 10^{-3} and a momentum of 0.9. The image size remains at 300×300 pixels, as increases in image size showed little improvement in performance, while increasing both the dataset size and the runtime to a factor of around 7. Hyperparameter optimization is beyond the scope of this thesis, and the focus is primarily on evaluating the performance of the selected model architecture and asserting the effectiveness of satellite sensing in combination with local data. A batch size of 48 images with the support of a weighted sampler are found to provide good results. The application of a weighted loss function is also investigated. However, using a weighted sampler yields better results. Weights are computed as the inverse of the fraction of METARs of the target class in relation to the number of METARs of all classes. Dropout or weight decay do not show improvements in model performance. Furthermore, manual labels have been assessed to overall overlap more closely with expected cloud cover distributions. This is supported by the model's ability to predict labels more closely when testing on manually labeled images, as shown by both Figure 5.1 and Figure 5.2 when comparing confusion matrices using METAR labels (a) against manual labels (b).

METAR Labels The best performance for training on METAR-based labels is presented in Figure 5.1, highlighting that the model is still able to generalize well from partially inaccurate data provided by METARs. While the prediction accuracies range from 41% to 82%, the most wrong predictions are just one cloud coverage class away from the expected result. When considering the directly neighboring classes as accurate enough too, the accuracy rises to the range of 93.5% to 99.1%. The F1-Score (d) is higher for validation than for training, indicating a problem with the model or the data. Data leakage is avoided through an accurate partitioning of the available stations at dataset creation. The quality of METARs is another possible source of this problem that fits previous observations.

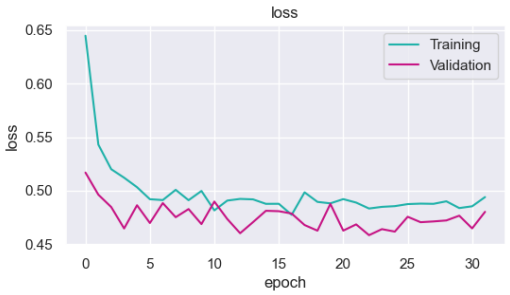
Manual Labels Training on manual labels further improves the performance of the model, as illustrated in Figure 5.2. The inaccuracy of METARs reflects in the predictions of the model, as seen in the confusion matrices using METAR labels (a) and manual labels (b), as well as between models in relation to Figure 5.1. Using manual labels as training basis, the same ordinal-regression model achieves prediction accuracies in the range of 62% to



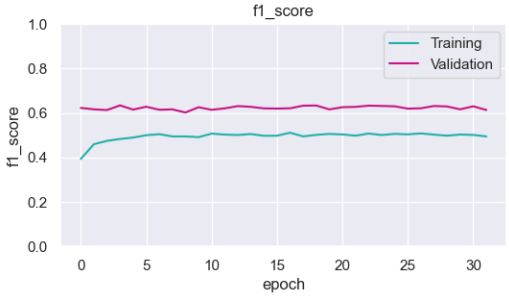
(a) Confusion matrix using METARs as true labels.



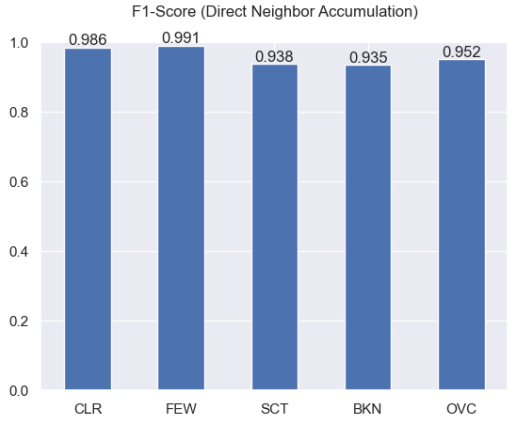
(b) Confusion matrix using manual labels as true labels.



(c) MSE loss function evaluation over time.



(d) F1-Score evaluation over time.



(e) Direct neigbor accumulation for confusion matrix using manual labels as true labels.

Figure 5.1.: Evaluation of ordinal regression model performance when trained on METAR labels on the NDSI supported dataset.

83%, with the largest performance gain being 34% for the BKN cloud coverage class. Also considering the neighboring classes for accuracy assessment, increases performance to the range of 98.4% to 100%, since most wrong predictions are still made inside the direct class neighborhood. Comparing the overall performance values between the usage of METAR labels and manual labels for training, shows that training on METAR labels generally leads to more deviation from the expected class and in most cases remaining in its direct vicinity. The F1-Score (d) is similar for both validation and training, which helps with addressing issues with the F1-Score in Figure 5.1 (d). Since the only change occurred with regard to the labels, data quality is the likely reason for the phenomena observed in the F1-Score for training on METAR labels.

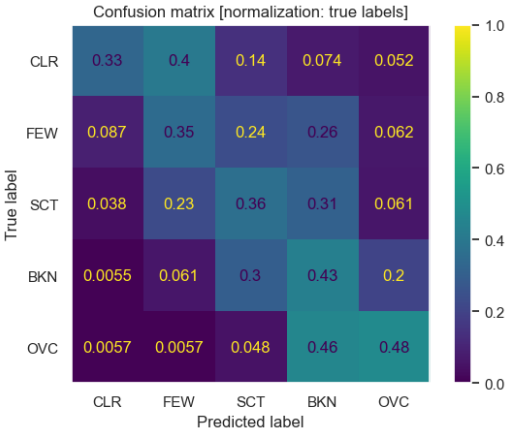
Model Evaluation Time Evaluating either model for images of size 300×300 pixels with 3 channels results in around 5.36 GMAC (Giga multiply-accumulates (MACs)) operations or around 10.72 GFLOPS (Giga floating point operations per second (FLOPS)) with approximately 20.18 million parameters, using the ptflops tool developed by Sovrasov [Sov]. Processing on an Nvidia RTX 3070 GPU with 8 GB of video memory, supported by an AMD Ryzen 7 7700X 8-Core Processor, shows that the model is able to predict the cloud coverage of 1286 images in around 16.6 seconds on average, relating to a processing speed of around 77.47 images per second. In comparison, the smallest variant of the Transformer-based UATNet uses 97 million parameters and encompasses 222.1 GFLOPS for processing images of size 512×512 pixels with 14 channels [Wan+22]. Other CNN-based models like KappaMask and RS-Net only provide run-time information on a specific machine, which impairs with comparability [Dom+21; Jep+19]. KappaMask uses TCI images of size 512×512 pixels with 3 channels for prediction on an Nvidia GTX 1070 GPU with 8 GB of video memory, supported by an Intel Core i7-8700K CPU. Its runtime on the GPU is around 237 seconds per image.

5.1. Hardware

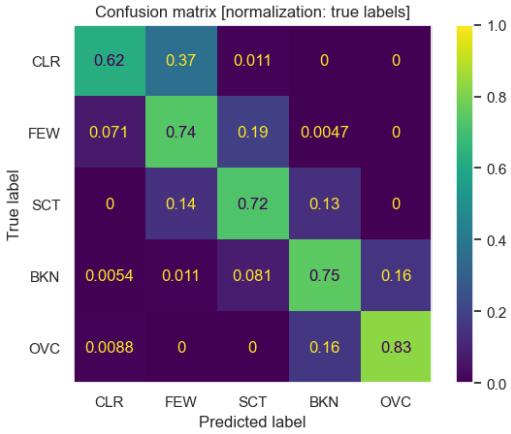
The model is tested on different systems. The bwUniCluster 2.0 system is used throughout early development, while the HDF-ML system is used for mid to late stages of development.

5.1.1. bwUniCluster 2.0

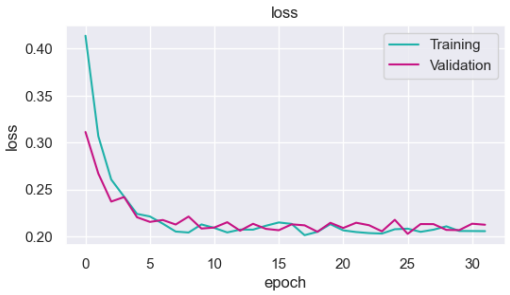
The bwUniCluster 2.0 [bwH] is a high-performance system that has been developed by universities of the German state Baden-Württemberg and universities of Applied Sciences. It is located at the Steinbuch Centre for Computing (SCC) at Karlsruhe Institute of Technology (KIT). The authors acknowledge support by the state of Baden-Württemberg through bwHPC. Multiple GPUs are used for training processes of the machine learning model. The bwUniCluster 2.0 provides Tesla V100, A100 and H100 cards from Nvidia for such tasks.



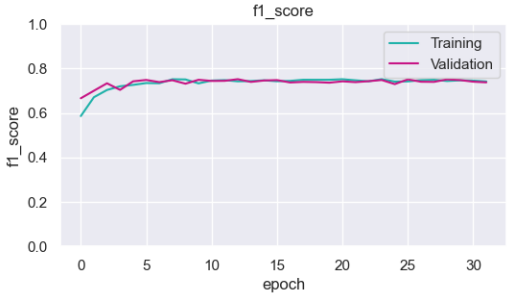
(a) Confusion matrix using METARs as true labels.



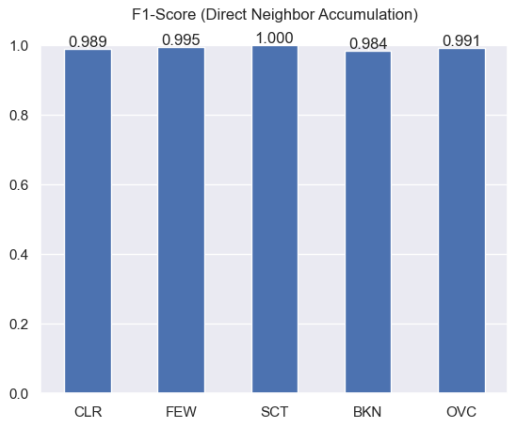
(b) Confusion matrix using manual labels as true labels.



(c) MSE loss function evaluation over time.



(d) F1-Score evaluation over time.



(e) Direct neigbor accumulation for confusion matrix using manual labels as true labels.

Figure 5.2.: Evaluation of ordinal regression model performance when trained on manual labels on the NDSI supported dataset.

5.1.2. HDF-ML

The HDF-ML supercomputer [Adv] provided by the Forschungszentrum Jülich (Juelich Research Center) and managed by the Institute for Advanced Simulation (IAS) is used for the training of the machine learning model. It runs on 48 Intel Xeon Gold 6126 CPUs at 2.60GHz and provides Nvidia V100 GPUs.

5.2. Software

The machine learning program itself is written in Python using the PyTorch framework [Fou]. The microservice-based backend for acquiring and preprocessing both satellite imagery and METARs is also written in Python. Communication between the machine learning program and the backend is facilitated through HTTP-REST communication that is wrapped by clients in the provided Python API for easier access. Python notebooks handle tasks regarding data queries, dataset creation, data analysis, model training and model evaluation. The code is made available together with this thesis for reproducibility.

6. Conclusion

The research objective of this thesis is evaluating the potential of using satellite observations to supplement ground based cloud cover data for deriving local cloud cover conditions at data-sparse regions without local weather stations. The thesis focuses to provide accurate but less complex, faster per-image cloud cover classification, adhering to the internationally standardized METAR format for the classes. Machine learning provides a great way to automatically learn hidden patterns and relationships from labeled data, which is why a deep-learning image classification algorithm is chosen. Model training simulates available ground based cloud cover information, while validation evaluates the prediction performance, using the provided cloud cover information only as comparison. Transfer learning is used to leverage general knowledge about patterns and relationships in images, improve the model, and reduce training time.

It is found that clearly representing the rank that is associated with the different cloud cover classes in the learning process provides the best results. This is done by using an ordinal-regression model. Further findings conclude that an image size of 300×300 pixels is adequate for the model, and that larger image sizes don't improve the performance. Additionally, the utilization of NDSI values provides good results for visually separating clouds from snow and ice. To provide some form of comparison for assessment of the quality of METAR labels, the 12800 images are also labeled manually. The results show that METARs provide partially inaccurate cloud cover labels, leading to worse performance in the model compared to manual labels. However, the model is still able to generalize from the partially inaccurate METARs, mostly deviating from the expected label by just one class-rank. It also performs well in terms of computational requirements and the usage of COTS hardware, enabling fast on-demand processing for decision-making, as well as tasks regarding continuous monitoring and near-real-time applications.

As the quality of local cloud cover measurements through METARs is partially inaccurate even for smaller deviations between the observations of satellite and weather station, impacting the performance of the machine learning model, further quality improvement techniques offer potential for future work. The data provided by METARs allows for the expansion of the model to assess learning capabilities regarding other measurements like wind speed and direction. Adding data sources for local measurements that include information about cloud composition holds potential to improve understanding of both internal and external cloud dynamics, enabling the assessment of cloud types. On the topic of machine learning, the exploration of different base models for transfer learning still holds potential. While the model benefits from the high density of data that is available in

the United States of America, regional expansion has the potential to further improve its generalization capabilities.

Bibliography

[ACC98] F Aviolat, T Cornu, and D Cattani. "Automatic clouds observation improved by an artificial neural network". In: *JOURNAL OF ATMOSPHERIC AND OCEANIC TECHNOLOGY* 15.1, 1 (Feb. 1998). 9th Symposium on Meteorological Observations and Instrumentation, CHARLOTTE, NC, MAR 27-31, 1995, pp. 114–126. ISSN: 0739-0572. DOI: 10.1175/1520-0426(1998)015<0114:ACOIBA>2.0.CO;2.

[Adv] Forschungszentrum Jülich - Institute for Advanced Simulation (IAS). *Jülich Supercomputing Centre (JSC)*. URL: https://www.fz-juelich.de/en/ias/jsc (visited on 06/26/2023).

[AN] National Aeronautics and Space Administration - NASA. *The Atmosphere: Getting a Handle on Carbon Dioxide*. URL: https://climate.nasa.gov/news/2915/the-atmosphere-getting-a-handle-on-carbon-dioxide/ (visited on 06/27/2023).

[Ari+21] P.A. Arias et al. "Technical Summary". In: *Climate Change 2021: The Physical Science Basis. Contribution of Working Group I to the Sixth Assessment Report of the Intergovernmental Panel on Climate Change*. Ed. by V. Masson-Delmotte et al. Cambridge, United Kingdom and New York, NY, USA: Cambridge University Press, 2021, pp. 33–144. DOI: 10.1017/9781009157896.002.

[Atz13] Clement Atzberger. "Advances in Remote Sensing of Agriculture: Context Description, Existing Operational Monitoring Systems and Major Information Needs". In: *Remote Sensing* 5.2 (2013), pp. 949–981. ISSN: 2072-4292. DOI: 10.3390/rs5020949. URL: https://www.mdpi.com/2072-4292/5/2/949.

[Bai+16] Ting Bai et al. "Cloud Detection for High-Resolution Satellite Imagery Using Machine Learning and Multi-Feature Fusion". In: *REMOTE SENSING* 8.9 (Sept. 2016). DOI: 10.3390/rs8090715.

[Bec+18] Hylke E. Beck et al. "Present and future Köppen-Geiger climate classification maps at 1-km resolution". In: *Scientific Data* 5.1 (Oct. 2018), p. 180214. ISSN: 2052-4463. DOI: 10.1038/sdata.2018.214. URL: https://doi.org/10.1038/sdata.2018.214.

[Bos16] Marina Bosch. "Principles Of Planetary Climate". In: 2016.

[bwH] bwHPC. *BwUniCluster2.0*. URL: https://wiki.bwhpc.de/e/BwUniCluster2.0 (visited on 04/15/2023).

[Car+22] M.A. Caretta et al. "Water". In: *Climate Change 2022: Impacts, Adaptation and Vulnerability. Contribution of Working Group II to the Sixth Assessment Report of the Intergovernmental Panel on Climate Change*. Ed. by H. O. Pörtner et al. Cambridge, UK and New York, USA: Cambridge University Press, 2022, pp. 551–712. ISBN: 9781009325844. DOI: doi:10.1017/9781009325844.006.

[CB04] Hyeungu Choi and Robert Bindschadler. "Cloud detection in Landsat imagery of ice sheets using shadow matching technique and automatic normalized difference snow index threshold value decision". In: *Remote Sensing of Environment* 91.2 (2004), pp. 237–242. ISSN: 0034-4257. DOI: https://doi.org/10.1016/j.rse.2004.03.007. URL: https://www.sciencedirect.com/science/article/pii/S0034425704000951.

[CWP08] Jianlin Cheng, Zheng Wang, and Gianluca Pollastri. "A neural network approach to ordinal regression". In: *2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence)*. 2008, pp. 1279–1284. DOI: 10.1109/IJCNN.2008.4633963.

[Doc] Docker. *Docker*. URL: https://www.docker.com/ (visited on 06/24/2023).

[Dom+21] Marharyta Domnich et al. "KappaMask: AI-Based Cloudmask Processor for Sentinel-2". In: *Remote Sensing* 13.20 (2021). ISSN: 2072-4292. DOI: 10.3390/rs13204100. URL: https://www.mdpi.com/2072-4292/13/20/4100.

[Dub+21] Oleg Dubovik et al. "Grand Challenges in Satellite Remote Sensing". In: *Frontiers in Remote Sensing* 2 (2021). ISSN: 2673-6187. DOI: 10.3389/frsen.2021.619818. URL: https://www.frontiersin.org/articles/10.3389/frsen.2021.619818.

[ESAa] ESA. *Copernicus Data Space Ecosystem*. URL: https://dataspace.copernicus.eu/ (visited on 04/25/2023).

[ESAb] ESA. *Copernicus Open Access Hub*. URL: https://scihub.copernicus.eu/ (visited on 04/25/2023).

[ESAc] ESA. *Copernicus Open Access Hub – User guide*. URL: https://scihub.copernicus.eu/userguide/ (visited on 06/24/2023).

[ESAd] ESA. *ESA Copernicus program*. URL: https://www.copernicus.eu/ (visited on 04/06/2023).

[ESAe] ESA. *ESA Copernicus program*. URL: https://step.esa.int/main/snap-supported-plugins/sen2cor/sen2cor-v2-11/ (visited on 04/13/2023).

[ESAf] ESA. *ESA ICEYE Constellation*. URL: https://www.eoportal.org/satellite-missions/iceye-constellation#performance-specifications (visited on 07/10/2023).

[ESAg] ESA. *ESA Sentinel-2 mission*. URL: https://sentinel.esa.int/web/sentinel/missions/sentinel-2 (visited on 05/05/2023).

[Est+96] Martin Ester et al. "A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise". In: *Knowledge Discovery and Data Mining*. 1996.

[Fou] The Linux Foundation. *PyTorch*. URL: https://pytorch.org/ (visited on 04/15/2023).

[Fu+19] Lee-Lueng Fu et al. "50 Years of Satellite Remote Sensing of the Ocean". In: *Meteorological Monographs* 59 (2019), pp. 5.1–5.46. DOI: https://doi.org/10.1175/AMSMONOGRAPHS-D-18-0010.1. URL: https://journals.ametsoc.org/view/journals/amsm/59/1/amsmonographs-d-18-0010.1.xml.

[Goo] Google. *TensorFlow*. URL: https://www.tensorflow.org/ (visited on 05/05/2023).

[Gor+22] V. I. Gornyy et al. "Satellite Mapping of Urban Air Overheating Risk (Case Study of Helsinki, Finland)". In: *COSMIC RESEARCH* 60.SUPPL 1, 1 (Dec. 2022), S38–S45. ISSN: 0010-9525. DOI: 10.1134/S0010952522700058.

[Hag+10] O. Hagolle et al. "A multi-temporal method for cloud detection, applied to FORMOSAT-2, VEN mu S, LANDSAT and SENTINEL-2 images". In: *REMOTE SENSING OF ENVIRONMENT* 114.8 (Aug. 2010), pp. 1747–1755. ISSN: 0034-4257. DOI: 10.1016/j.rse.2010.03.002.

[He+15] Kaiming He et al. "Deep Residual Learning for Image Recognition". In: *CoRR* abs/1512.03385 (2015). arXiv: 1512.03385. URL: http://arxiv.org/abs/1512.03385.

[Hea16] John Hearty. *Advanced Machine Learning with Python*. Packt Publishing, 2016. ISBN: 9781784398637.

[HLW16] Gao Huang, Zhuang Liu, and Kilian Q. Weinberger. "Densely Connected Convolutional Networks". In: *CoRR* abs/1608.06993 (2016). arXiv: 1608.06993. URL: http://arxiv.org/abs/1608.06993.

[Hol+96] Ben V. Hollingsworth et al. "Automated cloud cover assessment for Landsat TM images". In: *Imaging Spectrometry II*. Ed. by Michael R. Descour and Jonathan Martin Mooney. Vol. 2819. International Society for Optics and Photonics. SPIE, 1996, pp. 170–179. DOI: 10.1117/12.258064. URL: https://doi.org/10.1117/12.258064.

[Inta] National Park Service - U.S. Department of the Interior. *Bear Identification*. URL: https://www.nps.gov/articles/bear-identification.htm (visited on 06/26/2023).

[Intb] U.S. Department of the Interior. *Earth Resources Observation and Science – Data Center*. URL: https://eros.usgs.gov/eros-data-center-services/ (visited on 06/13/2023).

[Int18] International Civil Aviation Organization. *Annex 3 - Meteorological Service for International Air Navigation*. 20th Edition. ANN-00003-020-01. Montréal, Canada: International Civil Aviation Organization, 2018.

[Int21] International Civil Aviation Organization. *Manual of Aeronautical Meteorological Practice*. 13th Edition. DOC-08896-013-01. Manual. Doc 8896. Montréal, Canada: International Civil Aviation Organization, 2021.

[IOW] IOWA-IEM. *Iowa State University - Iowa Environmental Mesonet - ASOS-AWOS-METAR Data*. URL: https://mesonet.agron.iastate.edu/request/download.phtml (visited on 04/14/2023).

[Jep+19] Jacob Høxbroe Jeppesen et al. "A cloud detection algorithm for satellite imagery based on deep learning". In: *Remote Sensing of Environment* 229 (2019), pp. 247–259. ISSN: 0034-4257. DOI: https://doi.org/10.1016/j.rse.2019.03.039. URL: https://www.sciencedirect.com/science/article/pii/S0034425719301294.

[KB14] Diederik P. Kingma and Jimmy Ba. "Adam: A Method for Stochastic Optimization". In: *CoRR* abs/1412.6980 (2014).

[KMB16] Nada Kadhim, Monjur Mourshed, and Michaela Bray. "Advances in remote sensing applications for urban sustainability". In: *Euro-Mediterranean Journal for Environmental Integration* 1.1 (Oct. 2016), p. 7. ISSN: 2365-7448. DOI: 10.1007/s41207-016-0007-4. URL: https://doi.org/10.1007/s41207-016-0007-4.

[Köp36] Wladimir Peter Köppen. "Das geographische System der Klimate". In: 1936.

[Liu+21a] Ze Liu et al. "Swin Transformer V2: Scaling Up Capacity and Resolution". In: *CoRR* abs/2111.09883 (2021). arXiv: 2111.09883. URL: https://arxiv.org/abs/2111.09883.

[Liu+21b] Ze Liu et al. "Swin Transformer: Hierarchical Vision Transformer using Shifted Windows". In: *CoRR* abs/2103.14030 (2021). arXiv: 2103.14030. URL: https://arxiv.org/abs/2103.14030.

[Lon+16] Vincent Lonjou et al. "MACCS-ATCOR joint algorithm (MAJA)". In: *Remote Sensing of Clouds and the Atmosphere XXI*. Ed. by Adolfo Comerón, Evgueni I. Kassianov, and Klaus Schäfer. Vol. 10001. International Society for Optics and Photonics. SPIE, 2016, p. 1000107. DOI: 10.1117/12.2240935. URL: https://doi.org/10.1117/12.2240935.

[Mai+17] Magdalena Main-Knorn et al. "Sen2Cor for Sentinel-2". In: *Remote Sensing*. 2017. DOI: 10.1117/12.2278218.

[Man+18] Michael E. Mann et al. "Projected changes in persistent extreme summer weather events: The role of quasi-resonant amplification". In: *Science Advances* 4.10 (2018), eaat3272. DOI: 10.1126/sciadv.aat3272. eprint: https://www.science.org/doi/pdf/10.1126/sciadv.aat3272. URL: https://www.science.org/doi/abs/10.1126/sciadv.aat3272.

[MG17] Andreas C. Müller and Sarah Guido. *Introduction to Machine Learning with Python: A Guide for Data Scientists*. O'Reilly Media, Incorporated, 2017.

[Mit23] Ashim Kumar Mitra. "Use of Remote Sensing in Weather and Climate Forecasts". In: *Social and Economic Impact of Earth Sciences*. Ed. by Vineet K. Gahalaut and M. Rajeevan. Singapore: Springer Nature Singapore, 2023, pp. 77–96. ISBN: 978-981-19-6929-4. DOI: 10.1007/978-981-19-6929-4_5. URL: https://doi.org/10.1007/978-981-19-6929-4_5.

[NAS] NASA. *NASA Landsat program*. URL: https://landsat.gsfc.nasa.gov/data/ (visited on 04/06/2023).

[NBM17] P. T. Nastos, A. G. Bleta, and I. T. Matsangouras. "Human thermal perception related to Fohn winds due to Saharan dust outbreaks in Crete Island, Greece". In: *THEORETICAL AND APPLIED CLIMATOLOGY* 128.3-4 (May 2017), pp. 635–647. ISSN: 0177-798X. DOI: 10.1007/s00704-015-1724-3.

[Nor+16] Joel R. Norris et al. "Evidence for climate change in the satellite cloud record". In: *Nature* 536.7614 (Aug. 2016), pp. 72–75. ISSN: 1476-4687. DOI: 10.1038/nature18273. URL: https://doi.org/10.1038/nature18273.

[Nov+21] Josef Novotny et al. "Assessment of TAF, METAR, and SPECI Reports Based on ICAO ANNEX 3 Regulation". In: *ATMOSPHERE* 12.2 (Feb. 2021). DOI: 10.3390/atmos12020138.

[Num] NumFOCUS. *Pandas – Python Data Analysis Library*. URL: https://pandas.pydata.org/ (visited on 05/09/2023).

[Nwa+18] Chigozie Nwankpa et al. "Activation Functions: Comparison of trends in Practice and Research for Deep Learning". In: *CoRR* abs/1811.03378 (2018). arXiv: 1811.03378. URL: http://arxiv.org/abs/1811.03378.

[ONa] National Oceanic and Atmospheric Administration - NOAA. *FEDERAL METEOROLOGICAL HANDBOOK No. 1 - Surface Weather Observations and Reports*. URL: https://www.icams-portal.gov/resources/ofcm/fmh/FMH1/fmh1_2019.pdf (visited on 06/29/2023).

[ONb] National Oceanic and Atmospheric Administration - NOAA. *What are El Niño and La Niña?* URL: https://oceanservice.noaa.gov/facts/ninonina.html (visited on 06/26/2023).

[ONc] National Oceanic and Atmospheric Administration – NOAA. *Data Snapshots – An up-to-date archive of freely available climate maps*. URL: https://www.climate.gov/maps-data/data-snapshots (visited on 07/01/2023).

[ONe+22] B. O'Neill et al. "Key Risks Across Sectors and Regions". In: *Climate Change 2022: Impacts, Adaptation and Vulnerability. Contribution of Working Group II to the Sixth Assessment Report of the Intergovernmental Panel on Climate Change*. Ed. by H. O. Pörtner et al. Cambridge, UK and New York, USA: Cambridge University Press, 2022, pp. 2411–2538. ISBN: 9781009325844. DOI: 10.1017/9781009325844.025.2412.

[PFM07] M. C. Peel, B. L. Finlayson, and T. A. McMahon. "Updated world map of the Köppen-Geiger climate classification". In: *Hydrology and Earth System Sciences* 11.5 (2007), pp. 1633–1644. DOI: 10.5194/hess-11-1633-2007. URL: https://hess.copernicus.org/articles/11/1633/2007/.

[PTD89] Catherine Proy, Didier Tanré, and Pierre-Yves Deschamps. "Evaluation of topographic effects in remotely sensed data". In: *Remote Sensing of Environment* 30 (1989), pp. 21–32.

[QZH19] Shi Qiu, Zhe Zhu, and Binbin He. "Fmask 4.0: Improved cloud and cloud shadow detection in Landsats 4-8 and Sentinel-2 imagery". In: *REMOTE SENSING OF ENVIRONMENT* 231 (Sept. 2019). ISSN: 0034-4257. DOI: 10.1016/j.rse.2019.05.024.

[Raw+22] R. Rawshan Ara Begum et al. "Point of Departure and Key Concepts". In: *Climate Change 2022: Impacts, Adaptation and Vulnerability. Contribution of Working Group II to the Sixth Assessment Report of the Intergovernmental Panel on Climate Change*. Ed. by H. O. Pörtner et al. Cambridge, UK and New York, USA: Cambridge University Press, 2022, pp. 121–196. ISBN: 9781009325844. DOI: 10.1017/9781009325844.003.

[Ric04] Rudolf Richter. "ATCOR: Atmospheric and Topographic Correction". In: 2004.

[ROS58] F ROSENBLATT. "THE PERCEPTRON - A PROBABILISTIC MODEL FOR INFORMATION-STORAGE AND ORGANIZATION IN THE BRAIN". In: *PSYCHOLOGICAL REVIEW* 65.6 (1958), pp. 386–408. ISSN: 0033-295X. DOI: 10.1037/h0042519.

[Rum+15] Emal Rumi et al. "Field trial of an automated ground-based infrared cloud classification system". In: *METEOROLOGICAL APPLICATIONS* 22.4 (Oct. 2015), pp. 779–788. ISSN: 1350-4827. DOI: 10.1002/met.1523.

[SB05] Richard S. Sutton and Andrew G. Barto. "Reinforcement Learning: An Introduction". In: *IEEE Transactions on Neural Networks* 16 (2005), pp. 285–286.

[Shi+16] Mengyun Shi et al. "Cloud detection of remote sensing images by deep learning". In: *2016 IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*. 2016, pp. 701–704. DOI: 10.1109/IGARSS.2016.7729176.

[SLR80] J. Smith, Tzeu Lin, and K.J. Ranson. "The Lambertian Assumption and Landsat Data". In: *Photogrammetric Engineering and Remote Sensing* 46 (Sept. 1980).

[Sov] Vladislav Sovrasov. *ptflops: a flops counting tool for neural networks in pytorch framework*. URL: https://github.com/sovrasov/flops-counter.pytorch (visited on 07/10/2023).

[TL19] Mingxing Tan and Quoc V. Le. "EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks". In: *CoRR* abs/1905.11946 (2019). arXiv: 1905.11946. URL: http://arxiv.org/abs/1905.11946.

[TL21] Mingxing Tan and Quoc V. Le. "EfficientNetV2: Smaller Models and Faster Training". In: *CoRR* abs/2104.00298 (2021). arXiv: 2104.00298. URL: https://arxiv.org/abs/2104.00298.

[Wan+22] Zhanjie Wang et al. "UATNet: U-Shape Attention-Based Transformer Net for Meteorological Satellite Cloud Recognition". In: *Remote Sensing* 14.1 (2022). ISSN: 2072-4292. DOI: 10.3390/rs14010104. URL: https://www.mdpi.com/2072-4292/14/1/104.

[WMO22] World Meteorological Organization - WMO. *Aerodrome reports and forecasts: A Users' Handbook to the Codes*. 7 bis, avenue de la Paix – P.O. Box 2300 – CH-1211 Geneva 2, Switzerland: World Meteorological Organization, 2022.

[WWU23] Sophie-Berenice Wilmes, Sophie Ward, and Katsuto Uehara. "Chapter 9 - Present day: Tides in a changing climate". In: *A Journey Through Tides*. Ed. by Mattias Green and João C. Duarte. Elsevier, 2023, pp. 185–229. ISBN: 978-0-323-90851-1. DOI: https://doi.org/10.1016/B978-0-323-90851-1.00009-1. URL: https://www.sciencedirect.com/science/article/pii/B9780323908511000091.

[YL21] Xiaohe Yu and David J. Lary. "Cloud Detection Using an Ensemble of Pixel-Based Machine Learning Models Incorporating Unsupervised Classification". In: *REMOTE SENSING* 13.16 (Aug. 2021). DOI: 10.3390/rs13163289.

[Yor+21] John E. Yorks et al. "Aerosol and Cloud Detection Using Machine Learning Algorithms and Space-Based Lidar Data". In: *ATMOSPHERE* 12.5 (May 2021). DOI: 10.3390/atmos12050606.

[Zek+21] Viktoria Zekoll et al. "Comparison of Masking Algorithms for Sentinel-2 Imagery". In: *REMOTE SENSING* 13.1 (Jan. 2021). DOI: 10.3390/rs13010137.

[Zhu+19a] Zhe Zhu et al. "Benefits of the free and open Landsat data policy". In: *REMOTE SENSING OF ENVIRONMENT* 224 (Apr. 2019), pp. 382–385. ISSN: 0034-4257. DOI: 10.1016/j.rse.2019.02.016.

[Zhu+19b] Fuzhen Zhuang et al. "A Comprehensive Survey on Transfer Learning". In: *CoRR* abs/1911.02685 (2019). arXiv: 1911.02685. URL: http://arxiv.org/abs/1911.02685.

[Zup] Anze Zupanc. *Improving Cloud Detection with Machine Learning*. URL: https://medium.com/sentinel-hub/improving-cloud-detection-with-machine-learning-c09dc5d7cf13 (visited on 04/14/2023).

[ZW12] Zhe Zhu and Curtis E. Woodcock. "Object-based cloud and cloud shadow detection in Landsat imagery". In: *REMOTE SENSING OF ENVIRONMENT* 118 (Mar. 2012), pp. 83–94. ISSN: 0034-4257. DOI: 10.1016/j.rse.2011.10.028.

[ZWW15] Zhe Zhu, Shixiong Wang, and Curtis E. Woodcock. "Improvement and expansion of the Fmask algorithm: cloud, cloud shadow, and snow detection for Landsats 4-7, 8, and Sentinel 2 images". In: *REMOTE SENSING OF ENVIRONMENT* 159 (Mar. 2015), pp. 269–277. ISSN: 0034-4257. DOI: 10.1016/j.rse.2014.12.014.

A. Appendix

A.1. Köppen-Geiger Classification Criteria

The Köppen-Geiger classification criteria is decoded in Table A.1. The definition is split into three letters. The first letter is associated with the climate group that the region belongs to. The second letter represents seasonal precipitation and is not present for polar climate. The third letter relates to the seasonal temperature experienced by that region and is not present for tropical climate.

A.2. Image comparisons with and without NDSI support

This section contains comparisons between images using only the visible light (RGB) and those using the NDSI-based technique described in subsection 4.3.7. In addition to the images themselves, the METAR cloud cover assessment is provided as "max cloud cover", while "true cloud cover" represents the manually determined cloud coverage. The "timedelta [minutes]" provides information of the difference between the time of satellite observation and the time at which the weather station measurement is taken, in minutes. The difference is signed, meaning that a local station measurement that is provided earlier than the satellite image is annotated with a sign of negativity or minus "-". Potential NaN values of the NDSI computation are addressed by setting the problematic value to zero. Figure A.1 shows how using the NDSI approach effectively eliminates snow cover from the image by reducing its brightness. Figure A.2 highlights how the application of the NDSI complement affects bodies of water, reducing its brightness. This also leads to better separation of cloud streaks and the effects of water dynamics. Figure A.3 illustrates that there are reflectance variations with regard to cloud composition that reduce their brightness. However, clouds generally remain brighter than snow and ice after processing. Figure A.4 and Figure A.5 show more examples of separating clouds from snow and ice using the NDSI based approach to illustrate its general applicability.

Code	Specification
Af	Tropical, rainforest
Am	Tropical, monsoon
Aw	Tropical, savanna
BWh	Arid, desert, hot
BWk	Arid, desert, cold
BSh	Arid, steppe, hot
BSk	Arid, steppe, cold
Csa	Temperate, dry summer, hot summer
Csb	Temperate, dry summer, warm summer
Csc	Temperate, dry summer, cold summer
Cwa	Temperate, dry winter, hot summer
Cwb	Temperate, dry winter, warm summer
Cwc	Temperate, dry winter, cold summer
Cfa	Temperate, no dry season, hot summer
Cfb	Temperate, no dry season, warm
Cfc	Temperate, no dry season, cold
Dsa	Cold, dry summer, hot summer
Dsb	Cold, dry summer, warm summer
Dsc	Cold, dry summer, cold summer
Dsd	Cold, dry summer, very cold winter
Dwa	Cold, dry winter, hot summer
Dwb	Cold, dry winter, warm summer
Dwc	Cold, dry winter, cold summer
Dwd	Cold, dry winter, very cold winter
Dfa	Cold, no dry season, hot summer
Dfb	Cold, no dry season, warm summer
Dfc	Cold, no dry season, cold summer
Dfd	Cold, no dry season, very cold
ET	Polar, tundra
EF	Polar, frost

Table A.1.: Köppen-Geiger classification criteria decoded according to [Bec+18]

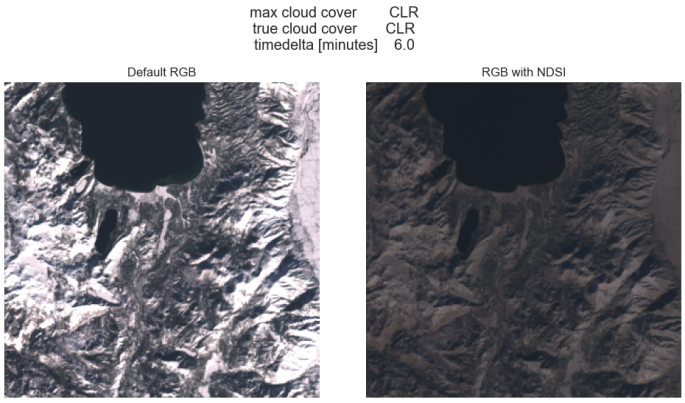


Figure A.1.: Snowy mountains with lake

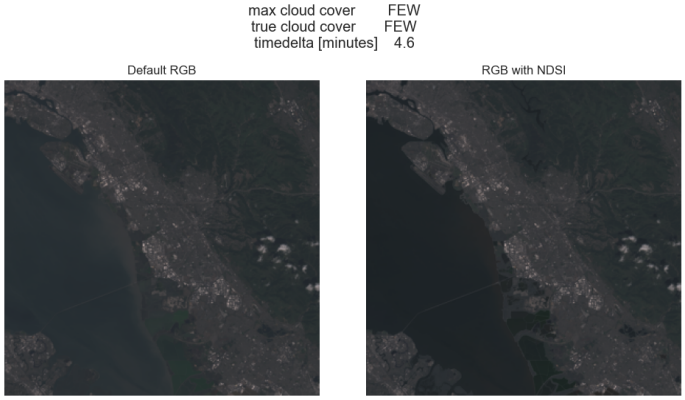


Figure A.2.: Coastline with few clouds

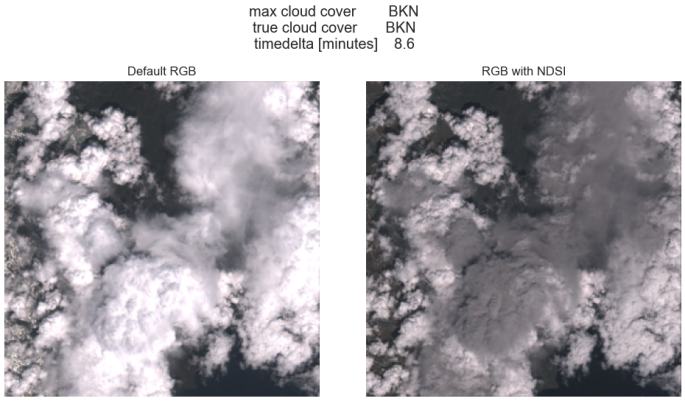
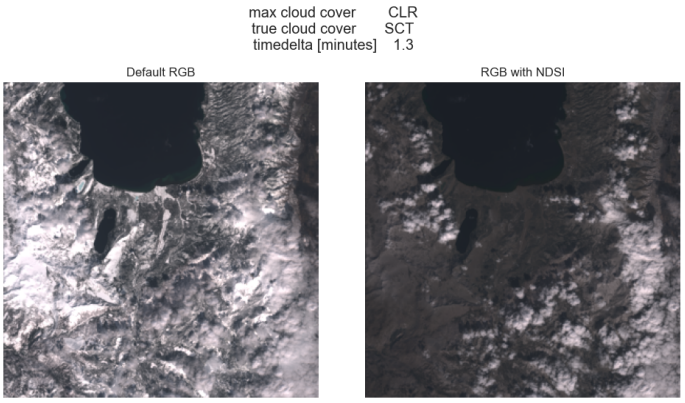
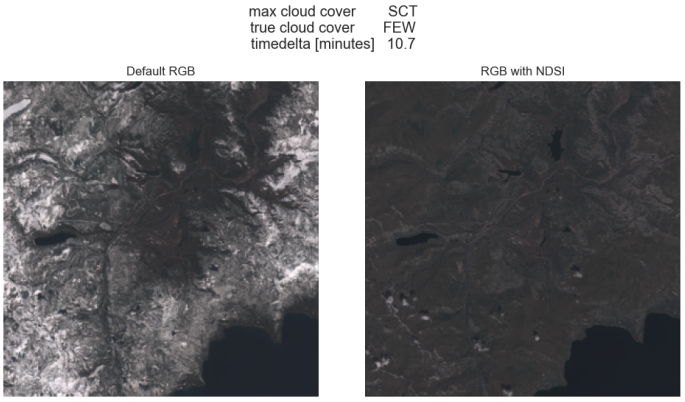


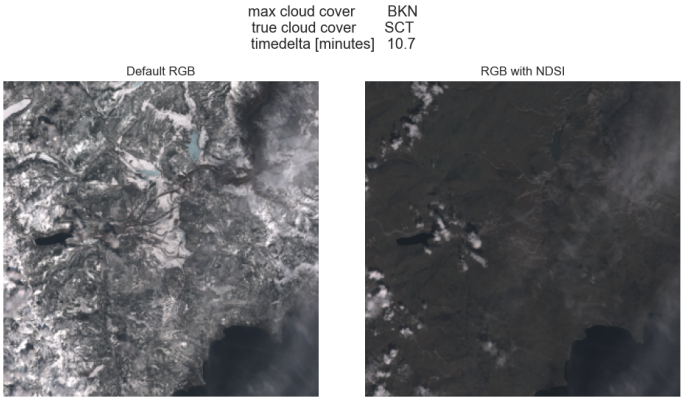
Figure A.3.: Occurrence of discolored clouds



(a)

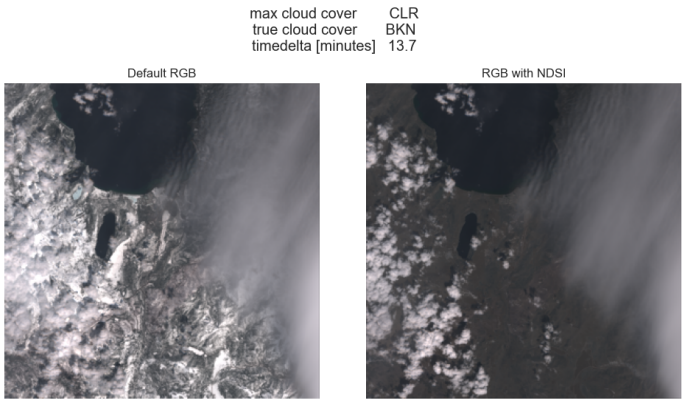


(b)

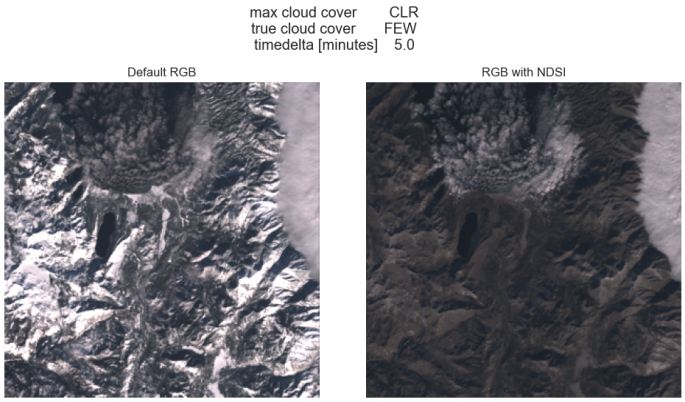


(c)

Figure A.4.: Examples of cloud coverage #1



(a)



(b)

Figure A.5.: Examples of cloud coverage #2