Bachelor Project

Czech
Technical
University
in Prague

F3

Faculty of Electrical Engineering
Department of Computer science

Land-cover satellite imagery classification

Viktor Müller

Supervisor: DOC. ING. MIROSLAV BUREŠ, PH.D.
Field of study: Open Informatics
Subfield: Artificial Inteligence
May 2021

ZADÁNÍ DIPLOMOVÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení:	**Müller**	Jméno: **Viktor**	Osobní číslo: **466824**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra počítačů**

Studijní program: **Otevřená informatika**

Specializace: **Umělá inteligence**

II. ÚDAJE K DIPLOMOVÉ PRÁCI

Název diplomové práce:

Anotace mapového podkladu podle satelitních snímků terénu

Název diplomové práce anglicky:

Annotation of digital map data based on satellite photographs of terrain

Pokyny pro vypracování:

Navrhněte a vytvořte proof-of-concept systému pro anotaci mapových podkladů na základě leteckých nebo družicových snímků terénu.
Systém bude analyzovat družicové nebo letecké snímky z otevřeného zdroje dat a pomocí metody strojového učení bude ve snímcích rozpoznávat sadu definovaných objektů.
Informace o těchto objektech bude systém poté přidávat do zvoleného otevřeného mapového podkladu.
Funkčnost systému ověřte vhodnou sadou testů a přesnost jeho fungování na základě dodaných trénovacích dat.

Seznam doporučené literatury:

Russ, J.C., Neal, F.B. Image Processing Handbook. Taylor & Francis, 2017

Jméno a pracoviště vedoucí(ho) diplomové práce:

doc. Ing. Miroslav Bureš, Ph.D., laboratoř inteligentního testování systémů FEL

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) diplomové práce:

Datum zadání diplomové práce: **21.02.2021** Termín odevzdání diplomové práce: **21.05.2021**

Platnost zadání diplomové práce: **19.02.2023**

doc. Ing. Miroslav Bureš, Ph.D.		prof. Mgr. Petr Páta, Ph.D.
podpis vedoucí(ho) práce	podpis vedoucí(ho) ústavu/katedry	podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Diplomant bere na vědomí, že je povinen vypracovat diplomovou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací.
Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v diplomové práci.

.

Datum převzetí zadání Podpis studenta

Acknowledgements

I would like to thank my supervisor Doc. Ing. Miroslav Bureš, Ph.D. for his guidance during my work on this master thesis. I would also like to thank Ing. Ondřej Pešek for many pieces of valuable advice related to geospatial image classification.

Declaration

Abstract

The management of search-and-rescue operations is a difficult task. A proper assignment of resources for individual segments of an estimated area where a missing person has been reported requires lots of manual work. Automation of this process could reduce the time needed for the planning phase of these operations. The exact information available about the terrain type in hard-to-reach areas is often out of date by many years. This master thesis is concerned with the application of machine learning to the analysis of satellite imagery. Predicting the terrain type from a satellite image that is only a few days old can add valuable information to an existing system for planning search-and-rescue operations used by the rescue services of the Czech Republic. In this work, several classifiers were implemented and an accuracy assessment experiment was conducted.

Keywords: satellite imagery, Sentinel-2, Land-cover, machine learning, categorical classification, geo-spatial images, search-and-rescue operations

Supervisor: DOC. ING. MIROSLAV BUREŠ, PH.D.

Abstrakt

Management pátracích operací po pohřesovaných osobách je složitý úkol. Velké množství manuální práce je vyžadováno při rozvržení pátracích prostředů do jednolivých sektorů pátrací oblasti. Automatizace tohoto procesu může výrazně snížit čas potřebný pro plánování takové operace. Přesná data popisující typy terénu v odlehlých oblastech jsou často několik let stará. Tato diplomová práce se zabývá aplikací metod strojového učení pro analýzu satelitních snímků. Predikce typu terénu podle pouze několik dní starého satelitního snímku může poskytnout cenné informace existujícímu systému pro plánování pátracích operací používanému v České Republice. V této práci je implementováno několik klasifikátorů, u kterých je následně proveden experiment pro vyhodnocení jejich přesnosti.

Klíčová slova: satelitní snímky, Sentinel-2, anotace mapového podkladu, strojové učení, kategorická klasifikace, geoprostorové snímky, pátrací operace po pohřešovaných osobách

Překlad názvu: Anotace mapového podkladu podle satelitních snímků terénu

Contents

Figures

Tables

Chapter 1

Introduction

Segmenting the surface of the Earth by distinct terrain types and providing machine-readable labels for such segments - a problem also known by the name Land-cover - can be used in a variety of geographic information system applications.

The focus of this work is to classify terrain in hard-to-reach regions, such as mountains, wast forests, and highlands. Proper labeling of such areas can then be used by the search-and-rescue services of the Czech Republic as an input for their rescue-operation planner system called *Pátrač*. With accurate enough information about the area of a particular operation, the planner can provide more relevant recommendations for the organization of the rescue operation and ultimately help to locate the missing person faster.

For this purpose, the terrain data must be as recent as possible. Therefore, using precise map structures is not desirable because this data is typically a few years out of date. Aerial surface imaging provides more up-to-date data, however, the delay can still be unacceptable for some regions.

Orbital satellite missions like Sentinel-2 photograph every region of the whole planet every week. This data is also available for any use and free of charge, which makes it the data-source of choice for this project.

In this thesis, several machine learning methods used for satellite imagery classification are discussed and validated on several possible rescue operation locations.

1.1 Task-definition

The task is defined in the following way: given a satellite image of rescue operation's area, produce labeled image for the *Pátrač* planner system. The goal of this thesis is to:

1. inspect the geo-spatial images available from the Sentinel-2 mission,
2. explore existing methodology to design a method for satellite image classification,
3. analyze Sentinel-2 data to further understand the data-source and it's limitations and possibilities,
4. follow selected classification methods and implement a classifier for Sentinel-2 imagery,
5. evaluate the classifier on real examples and provide relevant accuracy assessment.

Chapters of this thesis are structured based on the steps of the mentioned goal in the following manner:

1. Chapter one describes the current state of search-and-rescue operation methodology in the Czech Republic and then covers some background knowledge about geographic information systems and remote-sensing imagery using orbital Satellites.
2. Second chapter explores general methods of obtaining labels from satellite imagery and existing approaches used for solving this particular task before.
3. Chapter three then states algorithms and structures of selected methodology used in our implementation.
4. Fourth chapter is the implementation part, where we go over all of the steps necessary to predict labels for a satellite image.
5. Fifth chapter defines an experiment used for evaluating our method and reports results of conducted experiments.
6. And in the last chapter we discuss the prerequisits and accuracy of our classification method in a brief summary.

Chapter 2

Domain introduction

This chapter briefly addresses the essentials regarding the domain of this classification task. That includes the methodology of the Czech Republic's rescue operations, remote sensing systems - such as the Sentinel-2 - and a brief introduction to geographic information systems.

2.1 Search-and-rescue missions

If a missing person's assumed location is in the open-terrain (e.g. mountains, high-lands, fields, etc.), a search-and-rescue operation is initiated by the Czech Republic's Police. During this operation, human and material resources from the Police as well as resources from other structures of the Integrated Rescue System of the Czech Republic (like medics and firemen) are assigned for the search-and-rescue operation.

The problem of allocating specific resources for specific regions of the searched area as efficiently as possible arises. The goal of the *Vyhledávání pohřešovaných osob* (The Search of Missing People) methodology [2] is to use all available resources most efficiently, with respect to search time and economy. It achieves this by assigning proper resources to regions with their most search-effective terrain type.

The duration of the search-and-rescue operation can also be reduced by minimizing the time spent during decision processes. [2] This is achieved by:

- expert-training of the commanders of the operation and rescue teams,
- using IT systems for efficient task assignment among rescue teams,
- having state-of-the-art precise topography maps,
- using electronic navigation and communication devices.

The steps of the pipeline for this methodology are as follows:

1. estimating an area where the missing person is probably still located,
2. splitting this area into search regions (sectors),
3. estimate needed resources for an effective search of all sectors,
4. managing provided resources (allocate rescue teams to specific sectors) and initiate the physical search,
5. analyzing and validate the work of rescue teams,
6. creating documentation of the work that has been done,
7. evaluating search and rescue operation and store all statistics obtained during the operation.

The main goal of this thesis is to provide relevant information about the current terrain type within each sector. Given a segmentation into sectors, this information can be used to allocate search-and-rescue resources to sectors more efficiently (the third step of the pipeline) based on recent data. The system could then either automatically allocate resources or mark areas with inconsistent terrain types for further inspection by a human operator. Inconsistent terrain type occurs in such sectors where the terrain type defined by an outdated map data source does not match the recent terrain type predicted by our method.

2.1.1 Search sectors

The division of the target area into distinct sectors is crucial for the effectiveness of the search-and-rescue operation.

Each sector must be assignable to a specific search-and-rescue resource. In practice, this means that one sector should consist of a single terrain type. Table 2.1 shows the effectiveness of search and rescue resources for various terrain types.

Terrain type **Resource type**	Dog	Human swarm	Horse rider	Quad-bike / Motorbike	Helicopter / Drone	Thermal imaging	Boat	Diver
Open field	E	E	E	P	E	E	N	N
Farming field	E	P	P	N	P	P	N	N
Forest	E	E	E	P	P	P	N	N
Thicket	E	N	N	N	P	P	N	N
Steep slope	E	P	N	N	E	E	N	N
Rocks	E	N	N	N	E	E	N	N
Road/Path	E	E	E	E	E	E	N	N
Trench	E	E	P	P	E	E	N	N
Water area	N	N	N	N	P	N	E	E
Bank/Coast	E	P	P	N	E	P	E	E

Table 2.1: Effectiveness of search and rescue resources (E - effective, P - partially effective, N - not effective) [2].

In an ideal case, the classifier built in this thesis would be able to distinguish all terrain types listed in table 2.1. However, that is certainly not possible using only satellite imagery. To achieve proper separation of all mentioned terrain types, a combination of satellite imagery, elevation maps and infrastructure maps needs to be used by the search-and-rescue operation planner.

The list of terrain types in table 2.1 also does not include all possible terrain types. Additional types such as snow, ground-soil, and man-made objects need to be considered when classifying a satellite image, even though it does not correspond to any search sector of the search-and-rescue mission.

The size and shape of a sector are also important. The optimal size of a sector was set to 20 ha and minimal size to 10 ha (from previous experience with search-and-rescue operations) [2]. As for the shape, an optimal one would be a square, but that is usually not feasible because of the landscape. The shape should be as simple as possible and the edges of the shape should utilize natural landmarks, such as rivers, forest edges, or man-made objects like roads and paths. Following natural landmarks helps the rescue teams to successfully search the entire sector even where the GPS is not available [2].

2.2 Remote sensing systems

The following section is based mostly on the Remote Sensing book [3] by Andrew Skidmore.

Remote sensing can be defined as the acquisition of physical data of an object with a sensor that has no direct contact with the object. Photography of the surface of the Earth from above started with balloon flights, then proceeded with aerial images taken from planes, and then finally in the 1960s, the first meteorological satellite was launched into orbit [4]. The first repetitive and systematic observations were acquired by the Landsat 1 mission.

In 1980 NASA developed the first high spectral resolution instruments, which were able to cover not only the visible light waves but also shortwave infrared portions of the electromagnetic spectrum. At almost the same time, the first microwave remote sensing instrument was invented. Mounting these instruments on orbiting satellites provided ready access data of the Earth's surface on a global scale.

2.2.1 Low-resolution satellite systems

With a spatial resolution higher than 100 m, these satellites trade reduced spatial resolution for the high frequency of visits. They are typically placed in the geostationary orbit, providing a continuous point of view over a certain area. Frequent revisits are useful for meteorological applications, as changes in weather can be noticed quickly by these satellites. Examples of low-resolution satellite systems are Meteosat and NOAA.

Meteosat. Meteosat 1 was the first European meteorological geostationary satellite. Nowadays, Meteosat satellites have a spectral resolution of 2.5 km, providing images every 30 minutes.

NOAA. The NOAA satellite program is designed primarily for meteorological applications. The system has evolved over several generations of satellite instruments. NOAA operates in a two-satellite system. Both satellites are in a sun-synchronous orbit. Every spot on the surface of the Earth is captured at least twice each day. The spatial resolution of these satellites ranges from 1 km to 6 km, depending on wavelength bands. [5]

2.2.2 Medium-resolution satellite systems

These systems have medium area coverage, medium spatial-resolution, as well as a moderate revisit capacity. The width of the swath captured by the satellite is between 50 and 200 km, the spatial resolution is typically 10 - 100 m, and a revisit time of the same area on the surface of the Earth is more than three days.

The scale of images taken by medium-resolution satellite systems makes them especially suited for land management and land-use planning areas such as regions, countries, and possibly continents.

Landsat. Landsat mission, developed by NASA and the US Geological Survey, is the longest-running operation of this kind (operational since 1972). The most recent version of Landsat satellite systems is the Landsat 8. The orbits of Landsat 7 and 8 are set up for a revisit time of 8 days. The spatial resolution is 30 m.

Sentinel. Sentinel-2 mission, developed by the European Space Agency, is Europe's version of the Landsat operation. More details about this mission can be found in section 2.3.

2.3 Sentinel-2 mission

The Sentinel-2 is a European high-resolution, multi-spectral imaging mission. The mission is characterized by twin satellites flying in the same orbit but phased at 180°. This allows for a high revisit frequency of 5 days at the Equator. [1]

The objective of the Sentinel-2 mission is to provide frequent data for the study climate change, land monitoring, emergency management and security. Particular examples of its usage are land-cover maps, land-change detection maps and geophysical variables.

This makes the Sentinel-2 mission a useful data source for achieving the goal of this project. The only downside of the mission is the 10-meter resolution, which does not provide an ideal amount of details. However, only commercial or military satellites that are not publicly available offer satellite imagery with higher resolution, and for this reason, the Sentinel-2 mission was chosen as the main dataset for this thesis.

2.3.1 Sensing

The two satellites scan the Earth's surface in a passive way. The Multi-Spectral Instrument (MSI) uses a push-broom concept. A push-broom sensor works by collecting rows of image data across the orbital swath and utilizes the forward motion of the spacecraft along the path of the orbit to provide new rows for acquisition [1].

The instrument measures the radiance reflected by Earth's surface in 13 spectral bands according to their wavelengths. Table 2.2 lists all of the important information about supported MSI bands relevant for this project.

Band number	Central wavelength (nm)	Spatial resolution (m)	Description
1	443	60	coastal aerosol
2	492	10	blue visible light
3	560	10	green visible light
4	665	10	red visible light
5	704	20	vegetation red edge
6	740	20	vegetation red edge
7	782	20	vegetation red edge
8	833	10	near infra-red (NIR)
8a	864	20	narrow near infra-red
9	945	60	water vapour
10	1374	60	SWIR - cirrus
11	1614	20	SWIR
12	2202	20	SWIR

Table 2.2: Sentinel-2 satellite bands [1].

2.3.2 Provided data

There are two output products available for the Sentinel-2 mission.

The Level-1C product contains the top-of-atmosphere reflectance mapped to a cartographic geometry. This product is suitable for detecting clouds in the atmosphere and even comes with a rough pre-computed cloud mask [1].

The second product is the Level-2A atmospherically corrected surface reflectance mapped to a cartographic geometry. This product is considered to be analysis-ready data, as it can be directly used for analysis without further pre-processing [1].

2.4 Geo-spatial data

In this section, several topics regarding the geo-spatial datasource are explored.

2.4.1 Geographic coordinate systems

GCS is a coordinate system associated with positions on a planet. The position can be given by various representations. The most common and relevant for the next chapters are the spherical coordinate system and coordinates projected onto a plane.

Spherical coordinate system. A 3-tuple: latitude longitude and elevation. With the origin in the center of a spherical planet, these two angles and one distance number uniquely define a position (often on the surface) with respect to the reference ellipsoid of the planet. Omitting the elevation number, latitude and longitude are enough to determine a point on the surface of the Earth.

Map projection. A systematic representation of all or part of the surface of a round body on a plane [6]. A distortion of some kind must always be present in the projection because a spherical object can not be mapped onto a plane without one. Depending on the use case, a map projection that does not obscure the result too much is chosen.

There are more than 20 map projections described in detail in [6]. In this thesis, only the Universal Transverse Mercator projection used by Sentinel-2 satellite imagery [1] will be discussed further.

Transverse Mercator projection. This projection method, first presented by Johann Heinrich Lambert in 1772, is a transverse-cylindrical projection. The central meridian, the equator and each meridian 90° from the central meridian are straight lines. This map projection has little error close to the Equator. The main usage of this projection is for large-scale maps (like an entire map of the United States of America at a scale of 1:250,000) [6]. The Universal Transverse Mercator projection was adopted by the US Army for designing rectangular coordinates on large-scale military maps of the entire world. It contains additional parameters, such as central meridians. This creates a world-wide grid that segments the planet into quadrangles. Splitting the world into zones minimizes the amount of distortion of the Transverse Mercator projection [7].

2.4.2 Projected coordinate systems

A two-dimensional coordinate reference system is commonly defined by two axes. At right angles to each other, they form the X and Y plane. Additionally, an optional Z-axis perpendicular to both X and Y axes can be added for elevation.

2.4.3 Geo-spatial image

A geo-spatial image is a raster image, where each pixel can be mapped to a specific location on Earth's surface. This mapping is possible with additional metadata (apart from standard image resolution) associated with the image file:

- dataset bounds,

- transform matrix,
- coordinate-reference system (CRS).

Firstly, the data-set bounds determine the bounding box of the image with respect to the projected coordinate system.

Secondly, the transform matrix is an affine transformation matrix that maps pixel positions in (row, column) coordinates to (x, y) spatial positions of the projected coordinate system.

And finally, the CRS contains the map projection associated with the geo-spatial image. A position of a pixel in the projected coordinate system can be transformed to a location in the geographic coordinate system, thus resulting in a specific location on the surface of the Earth.

The geo-spatial image usually contains only one data channel, which is often a specific wavelength band reading. However, more channels can be present in the image, for example red, green, and blue channels for a true-color image composite. In fact, the geo-spatial image formats (like *GeoTIFF* or *netCDF*) support unlimitted number of data bands.

Chapter 3

Research of existing methodology and solutions

This chapter firstly states several approaches to the satellite imagery classification problem, including methods used directly for the Sentinel-2 images. Then, a few existing solutions to this problem are discussed in this chapter.

3.1 Existing satellite imagery classification methods

In this section, the theoretical approaches to the satellite image classification problem are described in more detail. After that, a few existing solutions to image classification are discussed further.

3.1.1 Manual labeling

The most straightforward approach to satellite imagery classification is to use humans to label the images. This approach is very robust and therefore it is often used for obtaining ground-truth labels. These labels can then be used to evaluate automatic methods. Manual labeling, however, has a few problems.

Firstly, it is very time-consuming. Precise labeling of a megapixel image can take a significant amount of time. This makes the manual approach not suitable for labeling high-resolution images of larger areas (like countries and continents).

Another problem is the requirement of an analyst who is familiar with the area. The accuracy of the classification does depend on the analyst's knowledge and experience.

An example of a data-set of manually collected labels for most European countries is the CORINE Land cover (see section 3.2.1).

3.1.2 Automatic classification

A supervised classification can automate the process of labeling pixels by making predictions based on a mathematical model. Methods of supervised classification require a set of training data which is used to set the parameters of the classifier.

The simplest approach to this classification problem is to classify each pixel independently of its surroundings. This method is called per-pixel classification and it has some advantages over more complex methods. The advantage is mainly the computation time required to predict pixels of a satellite image. Working with each pixel separately also opens the door for many different existing classifier solutions.

The study [8] compares the performance of known classifiers (k-nearest neighbor, random forest, support vector machine) in a per-pixel land-cover classification of Sentinel-2 images. Per-pixel classification can also be used on smaller areas with high resolution, reaching overall accuracy of over 90 % with the SVM classifier. Another study [9] uses per-pixel classification to achieve higher than 85 % accuracy on a 0.5 m resolution image of North Carolina.

A different method uses contextual information from the neighborhood of a target pixel to predict its label. By utilizing the surroundings, this method has the potential to distinguish pixels with equal single-pixel features from one another. The list of classifiers supporting contextual features is limited, and the prediction phase also requires the surroundings of a pixel, making it computationally more demanding.

A convolutional neural network classifier was used by [10] to learn contextual features from high-resolution satellite imagery. Multiple CNN classifiers are trained using multiple pixel neighborhood sizes (15, 25, 35, and 45-pixel size). The prediction is then an ensemble of all of these classifiers, achieving an overall accuracy of 94 %. A different study [11] uses an iterative decision forest classifier to predict pixel labels based on their neighborhood to achieve 84 % overall accuracy in a satellite image of the German city Rostok.

3.2 Existing land-cover applications

The following sections describe the use-cases and limitations of existing land-cover solutions.

3.2.1 CORINE Land Cover

The CORINE Land Cover (CLC) project initiated in 1985 is an inventory of labeled vector data. Europe is the area of interest of this project. The labeled data is available for download for free. The dataset as well as the methods of obtaining it changed throughout the years. For this reason, only the CLC2018 dataset, which is the latest completed CORINE dataset, is discussed further.

The CLC2018 is a dataset with 44 distinct category classes. It has a minimum mapping unit of 25 hectares and the minimum width of linear elements is 100 meters [12]. This means that the smallest feature polygon of a single class is always larger than 25 hectares in size.

Earth observation satellite imagery is the basis of CLC mapping, providing up-to-date information about the surface of the Earth in proper resolution. Sentinel-2 and Landsat-8 satellite imagery is used in the case of CLC2018.

The following are the examples of the use cases of the CLC2018 data-set: [13]

- the monitoring of the expansion of urban areas over the years in Germany,

- the analysis of scale and type of damage to forest areas,
- analysis reports for territorial development and cohesion, climate change mitigation and adaptation, agriculture, and forestry management.

Even though this existing solution seems like the perfect candidate for this use-case, the CLC2018 data-set aims to help with the classification of huge areas (global landscape of countries or regions). The minimum mapping unit of 25 hectares makes it not precise enough for search and rescue operations, where the optimal size of a single search sector mentioned in section 2.1.1 is 20 hectares. Because of the imprecisions, this dataset can not help us gather training data for our methods of supervised classification.

3.2.2 Vandersat

Vandersat uses its own set of remote-sensing satellites which use passive and active microwave radiation to obtain an image describing the quality of the soil. [14]

The microwaves pass through clouds and can be detected even at night. This makes the sensing window to be 24 hours a day, which allows much more frequent imaging.

The data acquired by Vandersat's satellites include information about soil moisture, biomass, temperature, and vegetation optical depth. Using this information, Vandersat's classifiers are able to predict the quality of soil for agricultural purposes. Automatically identifying areas of draught helps farmers to develop smarter crop insurance, in collaboration with the world's most innovative (re)insurers and brokers. [14]

The dataset is proprietary and thus can not be used in this thesis.

Chapter 4

Used algorithms and structures

The following chapter describes the theoretical concepts behind algorithms and structures used later in the implementation chapter 5. Some background knowledge of statistics is expected from the reader.

4.1 K-means clustering

The k-means algorithm is a an iterative method for partitioning observations into k clusters (where k is a required parameter).

The input to this algorithm is a set of n observations: $D = \{\boldsymbol{x}_i \mid i = 1 \mathinner{..} n\}$. Each observation $\boldsymbol{x}_i \in \mathbb{R}^d$ is a d-dimensional vector of features.

In the initial step, the algorithm chooses k observations as intial centroid locations. There are multiple methods for choosing centroid locations in the initial step - the simplest one is random sampling.

The algorithm then iterates till convergance with two steps:

1. assignment of each observation to its closest centroid,

2. each centroid's location is updated to center (geometric mean) of all observations assigned to that centroid.

The algorithm converges when the centroid locations do not move by at least some small threshold value.

A specification of a distance metric is required for this algorithm to compute the distance between centroid's location and an observation. The default distance metric is the Euclidean distance. Using this metric, we can define the cost function for this algorithm:

$$\sum_{i=1}^{n} \arg\min_{j} \|\boldsymbol{x}_i - \boldsymbol{c}_j\|_2^2 \ ,$$

where $\boldsymbol{c}_j$ is the location of j-th centroid and $\|\cdot\|_2$ is L2 norm (the Euclidean distance metric in space). This cost function is non-negative and decreases with each new relocation of centroids [15]. Once the algorithm converges, a minimum is found. It is, however, only a local minimum, because the cost function is non-convex and the algorithm descends greedily [15].

The main issue of the k-means algorithm is that the assignment of observations to their nearest centroid limits the cluster shape to a hyper-sphere. If the input data can not be clustered reasonably into hyper-spherical clusters, then the local minimum found by the k-means has poor clustering performance. This problem is encountered for example when the features of observations are not of the same scale, and can be mitigated by *whitening* (a process which transforms the input observations so that each of the random variables associated to the features has a unit variance) [16].

4.2 EM-algorithm

Clusters in data can often be accurately represented by a mixture of mathematical models. These mixtures can be fitted to the data via the Expectation-Maximization (EM) algorithm iteratively by the maximum likelihood estimation method.

In case of the normal mixture model-based approach to clustering, the data is assumed to be sampled from a mixture of an initially specified number (g) of multivariate normal densities. The unknown proportions of these normal densities are denoted by $\pi_1, \ldots, \pi_g$. The combined probability density function, which each data point is assumed to be taken from, is:

$$f(\boldsymbol{y}; \Psi) = \sum_{i=1}^{g} \pi_i \cdot \phi(\boldsymbol{y}; \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i) \ ,$$

where $\phi(\boldsymbol{y}; \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)$ denotes the p-variate density probability function with mean vector $\boldsymbol{\mu}_i$ and covariance matrix $\boldsymbol{\Sigma}_i$. Ψ represents the unknown parameters: mixing proportions $\pi_1, \ldots, \pi_g$, elements of component means $\boldsymbol{\mu_1}, \ldots, \boldsymbol{\mu_g}$, and distinct elements of the component-covariance matrices $\boldsymbol{\Sigma}_1, \ldots, \boldsymbol{\Sigma}_g$. [17]

Once the models are fitted, a probabilistic clustering of the data into g clusters can be obtained. Each data point is assigned to the component to which it has the highest estimated posterior probability of belonging. Posterior probability that an observation $\boldsymbol{y}_j$ belongs to the i-th component is denoted by $\tau_i(\boldsymbol{y}_j)$ and can be computed by:

$$\tau_i(\boldsymbol{y}_j, \Psi) = \frac{\pi_i \cdot \phi(\boldsymbol{y}_j; \boldsymbol{\mu}_i, \boldsymbol{\Sigma})_i}{\sum_{k=1}^{g} \pi_k \cdot \phi(\boldsymbol{y}_j; \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)} \ .$$

4.2.1 Maximum likelihood estimation

The maximum likelihood estimate of Ψ can be obtained as an appropriate root of the likelihood equation:

$$\frac{\partial \log L(\Psi)}{\partial \Psi} = 0 \ .$$

Here, $L(\Psi)$ denotes the likelihood function for Ψ formed from the observed random sample $\boldsymbol{y}_1, \ldots, \boldsymbol{y}_n$, and so the log-likelihood is obtained by:

$$\log L(\Psi) = \sum_{j=1}^{n} \log f(\boldsymbol{y}_j; \Psi) \ .$$

4.2.2 Expectation-Maximization steps

The Expectation-Maximization algorithms is an iterative procedure of two steps:

1. the E-step computes posterior probabilities of belonging to each mixture component for each data point,
2. the M-step estimates new parameters Ψ from the assignment of the E-step.

Formally, the E-step in iteration k computes: [15]

$$\tau_{i,j}^{(k)} = \frac{\pi_i^{(k)} \cdot \phi(\boldsymbol{y}_j; \boldsymbol{\mu}_i^{(k)}, \boldsymbol{\Sigma}_i^{(k)})}{f(\boldsymbol{y}_j; \Psi^{(k)})} \ ,$$

and the M-step in the same iteration computes new parameters for each mixture component. The following are the computations of the M-step in k-th iteration for i-th component: [15]

$$\pi_i^{(k+1)} = \frac{\sum_{j=1}^{n} \tau_{i,j}^{(k)}}{n} \ ,$$

$$\boldsymbol{\mu}_i^{(k+1)} = \frac{\sum_{j=1}^{n} \tau_{i,j}^{(k)} \boldsymbol{y}_j}{\sum_{j=1}^{n} \tau_{i,j}^{(k)}} \ ,$$

$$\boldsymbol{\Sigma}_i^{(k+1)} = \frac{\sum_{j=1}^{n} \tau_{i,j}^{(k)} (\boldsymbol{y}_j - \boldsymbol{\mu}_i^{(k+1)})(\boldsymbol{y}_j - \boldsymbol{\mu}_i^{(k+1)})^T}{\sum_{j=1}^{n} \tau_{i,j}^{(k)}} \ .$$

These E- and M-steps alternate until the log-likelihood or the changes in the estimated parameters are lower than a specified threshold.

4.3 k-nearest-neighbor classifier

The kNN classifier is one of the simplest classifiers out there. It makes a prediction for a test object using two steps:

1. find k objects in the training set which are closest to the test object based on a distance metric (create a neighborhood of k objects),
2. assign a label to the test object based on the majority class in this neighborhood.

Formally, given a test object $z = (\mathbf{x}', y')$ and a training set D with labeled training objects $(\boldsymbol{x}, y) \in D$, where $\boldsymbol{x}$ is a vector of features and y is the object's assigned label, the algorithm computes a distance (or similarity) between z and all training objects and thus determines its nearest-neighbour list D_z. After obtaining this nearest-neighbour list, the predicted label y' is chosen by majority voting:

$$y' = \arg\max_{v} \sum_{(\boldsymbol{x}_i, y_i) \in D_z} I(v = y_i) \ ,$$

where v is a class label and I is an indicator function that returns 1 if its argument is true, 0 otherwise. [18]

The training procedure only requires loading training objects into memory, which is significantly faster than training procedures of other classifiers. The preparation time of this classifier makes the kNN useful as a baseline for comparison with other classifiers.

There are, however, some considerations when setting up a kNN classifier. The first one is the choice of k, which typically needs to be found by an empirical search. Another one is the choice of the distance metric, which influences the behavior of the classifier significantly and is domain-specific.

An example distance metric for spatial data can be the Euclidian distance. Note that this distance metric becomes less discriminating as the number of features increases. [15]

Although the training time of kNN is insignificant, the prediction time is computationally demanding. The classifier needs to compute the distance metric to all of its training objects. The search for the nearest neighbors can be done efficiently using the K-d tree algorithm [19] in logarithmic time. However, if the number of dimensions increases and 2^K becomes larger than the number of objects to search for, the search of the K-d tree algorithm becomes linear with respect to the number of data points (training objects in case of kNN) [20].

4.4 Convolutional neural network

The convolutional neural network (CNN) is our main classifier. CNN is used for classification tasks, where the relative position of features is important and can significantly influence the decision. The most common use-case for CNN is the image classification task.

This section describes the individual components of CNN. Typically, this network consists of alternating convolution and pooling layers, followed by a block of fully connected layers. Other structures for optimizations are also described here. The definitions for these components were adopted from [21].

4.4.1 Convolutional layer

The convolution layer consists of a set of learnable kernels (sometimes referred to as filters). The kernels are small matrices with just a couple of pixels in width and height and the same depth as the input image. Each kernel has a set of weights that are learned during the training process of the network. Convolutional kernel works by dividing the input image into slices of the same size. This division is called the receptive field.

Each kernel is then applied to each slice of the image, where it multiples the pixels of the slice with its weights and sums the results into a single value output for each slice. This operation for a specific kernel in the convolutional layer can be expressed like so:

$$f(p,q) = \sum_{c} \sum_{i,j} R_{p,q,c}[i,j] K[i,j] \ ,$$

where c is the image channel index, i,j are indices of the kernel, K is the matrix of the kernel, $R_{p,q,c}$ is the receptive field of the input image for channel c, corresponding to the position $[p,q]$ in the ouput matrix, and finally, $f(p,q)$ is a single value in the output matrix at position $[p,q]$.

Parameters for this layer include a stride value and optionally padding handling. The horizontal and vertical stride values determine by how many pixels the kernel is moved in each direction. An edge-case scenario can happen for stride values bigger than one because the kernel would be shifted out of the input image. This can either be prohibited (values near the edges are not considered for classification) or permitted by specifying a default value for the necessary padding that is added beyond the edges.

Figure 4.1 shows an example of a convolutional layer with two kernels of size 2x2 sliding over a single-channel image. Both horizontal and vertical strides equal to 1 in this example.

The main advantage of the convolutional layer is the number of weights that need to be learnt. The weights for a single kernel are shared for the whole sliding operation. Different sets of features within an image can still be extracted and the number of weights is significantly lower than in the case of fully connected layer (discussed later in section 4.4.5). [22]

4.4.2 Pooling layer

The pooling layer (also known as down-sampling) is designed to lower the resolution of data inside the neural network's structure. The idea behind the pooling layer is that the exact location of features becomes less important as long as its approximate relative position to other features is preserved.

Similarly to the convolutional layer, a receptive field is created based on the selected pooling neighborhood. Pooling then sums up the neighborhood of the

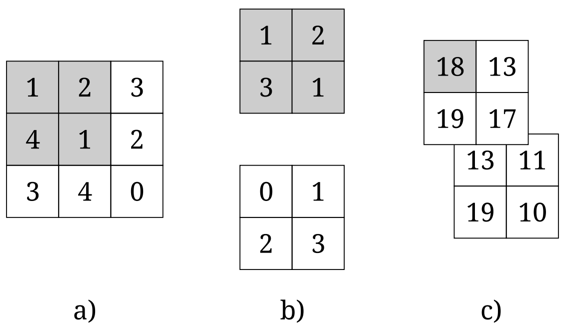


Figure 4.1: Example of a convolutional layer with two kernels. **a)** is a single-channel input image with pixel values, **b)** shows the two kernels and their weights, and **c)** is the resulting output from this convolutional layer (a three-dimensional tensor that shrank in width and height due to kernel size and stride parameters but grew in depth due to the number of kernels for this layer)

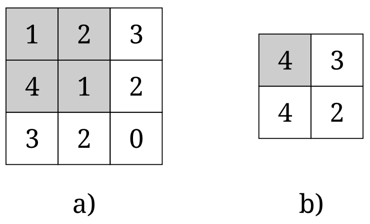


Figure 4.2: Example of a MaxPooling layer. **a)** is a single-channel input image with pixel values, and **b)** is the resulting output from this MaxPooling layer.

receptive field by outputting the dominant response within this region. What response is dominant depends on the pooling function. The most commonly used pooling layer is the MaxPooling layer, which selects the maximum as the dominant feature for each region. However, other functions, such as average and overlapping, are possible. The simplest way to understand this operation is visual. Figure 4.2 shows the MaxPooling layer applied to a 3x3 input matrix.

The use of pooling helps to extract a combination of features which are invariant to translational shifts and small distortions. This does not only reduce the complexity of the network but it also leads to better generalization. [23]

4.4.3 Batch normalization

Batch size. A batch size of a neural network is the amount of training observations that are passed through the network during the training proccess before the weights of the network are updated. The training data-set can be split into multiple batches (this method is called mini-batch). This is a meta-parameter and must be decided by the designer of the neural network.

The internal covariance shift is a change in the distribution of values of hidden units. This shift slows down the convergence by forcing learning rate to small value. Batch normalization unifies the distribution of feature-map values by setting them to zero mean and unit variance [24]. This is achieved by:

$$y_i = \frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}} \ ,$$

where x_i is the value of feature-map input for this layer, μ_B is the mean of input values for a batch, σ_B is the standard deviation of input values for a batch, ϵ is a small constant added to avoid division by zero, and y_i is the normalized feature-map output.

This operation smoothens the flow of gradient during training, and thus helps with the generalization.

4.4.4 Dropout

This method temporarily removes randomly selected neurons and their connections from the networks. Dropout can help to avoid overfitting during the training of neural networks [25].

4.4.5 Fully connected layer

Unlike the convolutional and pooling layer, which operated on a local level, the fully connected layer applies global operations. This layer consists of a specified number of output neurons and it connects all of them with all of the neurons of the previous layer, assigning a learnable weight to each such connection. Each neuron of the fully connected layer also applies a non-linear function to the sum of its inputs:

$$y_j = f(\sum_i w_{i,j} x_i) \ ,$$

where x_i is the feature of previous layer, $w_{i,j}$ is the weight associated with the connection from i-th feature of the previous layer to j-th neuron of this layer, f is an activation function, and y_j corresponds to the output of j-th neuron of this layer.

The fully connected layer is usually placed as the last layer in a CNN classifier. The number of neurons of the last layer corresponds to the number of classes to classify.

4.4.6 Activation function

The activation function is essential for any neural network, as it introduces non-linearity to the algorithm, which makes it possible to learn intricate patterns.

There is a variety of non-linear functions used for neural network training (such as *sigmoid* or *tanh*). Each such function has a different impact on the learning process. Variants of the ReLU (Rectified Linear Unit) function have low computation time and also help to overcome the vanishing gradient problem, and therefore became the standard activation function for neural networks [26]. ReLU function is defined as $f(x) = max(0, x)$.

Another useful activation function is the Softmax activation function. Input of this function is a vector $\boldsymbol{x}$ of n and the function for i-th neuron of a fully connected layer is defined as:

$$f_i(\boldsymbol{x}) = \frac{e^{x_i}}{\sum_{j=1}^{n} e^{x_j}} \ .$$

Placing a fully connected layer with Softmax activation function as the last layer of neural network results in an output of positive values that sum up to one, which can then be interpreted as prediction probabilities.

Chapter 5

Implementation

In this chapter, the process of implementing previously defined methods is described in detail. Also, the benefits and drawbacks of each classifier are discussed in this chapter.

5.1 Obtaining data

Obtaining satellite imagery can be achieved in multiple ways. The Copernicus Open Access Hub provides a free and open access to Sentinel-1, Sentinel-2, Sentinel-3 and Sentinel-5P user products. [27]

There is a user-friendly way to select images and also and application interface for bulk-downloading large amounts of images at once.

5.2 Area of interest

To be able to test any classification method, some specific data source is required. For all of the following implementation steps, three areas of interest were chosen inside the Czech Republic:

Name	center latitude (degrees)	center longitude (degrees)	area (km squared)
Krkonoše	50.7171	15.6541	140.13
Vysočina	49.5927	15.9765	176.27
Morava	48.8283	16.7706	168.89

Table 5.1: Geographic properties of chosen areas of interest.

1. an area near the city Špyndlerův Mlýn in the *Krkonoše* mountain complex,
2. an area near the city Nové Město na Moravě in the *Vysočina* highlands,
3. an area near the city Hustopeče in the Moravian plains.

Even though these areas were chosen arbitrarily, they are representatives of three very different landscape types.

The first area of interest is in the highest mountain complex in the country. It is a very likely candidate for a real scenario for the rescue operation. The goal of this area is to test the terrain classification on most relevant data, as search-and-rescue operations have high likelihood of occurrence in this area. The test results for this area are required to be as satisfying as possible.

The second area of interest is from the middle part of the country. The landscape is highlands, which is quite different from the first mountain area. The average elevation is therefore lower. This area is also an expected region for real rescue missions. The classification method should provide good test results for this area as well.

The last area of interest is from flatlands. Although the likelihood of having a real rescue mission scenario in this area is significantly lower than in the previous two, it is still a possibility. Therefore it is representative of an edge case to further test the terrain classification approaches in an uncommon environment. The classification results are not required to be perfect for an area of this type. On the other hand, the results still need to be reasonable.

The figure 5.1 shows the geographic location and an aerial preview of chosen areas of interest.

Detailed properties of these areas can be seen in table 5.1.

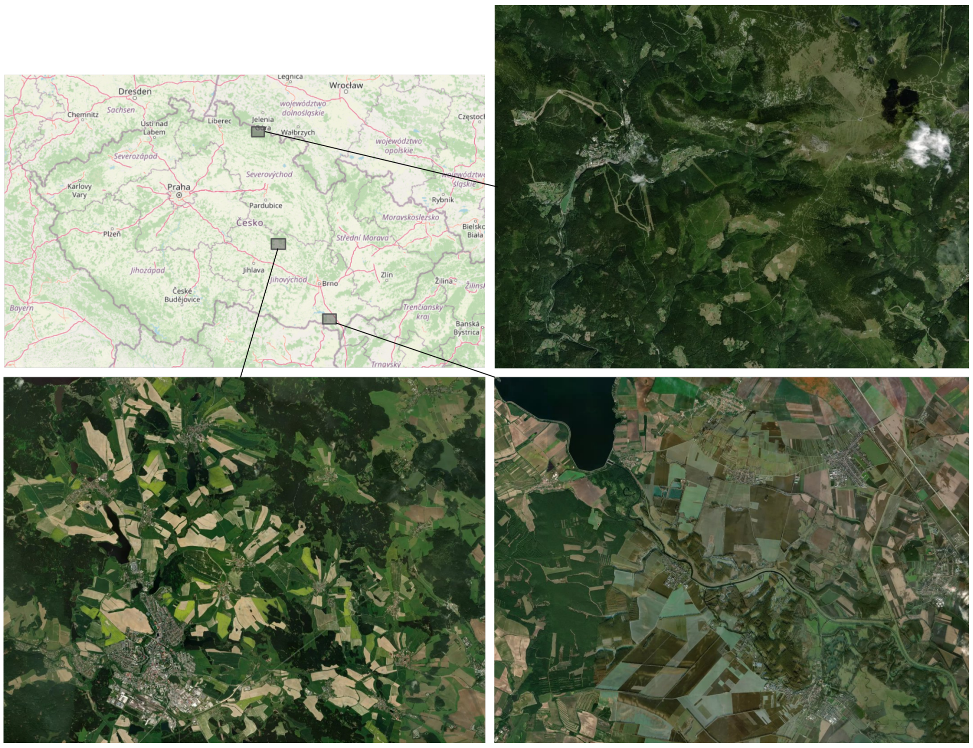


Figure 5.1: The chosen areas of interest. *Krkonoše* in the top right, *Vysočina* in the bottom left and *Morava* in the bottom right.

5.3 Bulk-download

The Python package SentinelSat [28] was used to download Sentinel-2 satellite images.

To download data from Copernicus Open Access Hub, the SentinelSat package needs to specify the satellite sensing period, the geographical image tile, and the maximum allowed percentage of clouds in the image.

The goal is to download usable satellite images of chosen areas of interest. For an image of a specific area to be usable, the image must not contain many clouds. It is also important to obtain an image for all of the seasons of the year because the environment in the area changes drastically throughout the year.

The chosen period was the whole year 2020. The geographical tile depends on the area of interest. And the maximum cloud percentage was arbitrarily set to 70 %. The reason behind choosing 70 % as the maximum allowed

area name	sensor	tile id	number of images	data size (GB)
Krkonoše	L1C	T33UWS	73	66
Krkonoše	L2A	T33UWS	73	72
Vysočina	L1C	T33UWQ	33	32
Vysočina	L1C	T33UWR	29	31
Vysočina	L2A	T33UWQ	32	33
Vysočina	L2A	T33UWR	26	31
Morava	L1C	T33UXQ	78	76
Morava	L2A	T33UXQ	76	84

Table 5.2: Summary of downloaded Sentinel-2 imagery.

percentage was that from empirical observation of many satellite images it was clear that the likelihood of obtaining a cloudless area of interest in an above 70 % cloudy image was so low, that it was not worth downloading such images.

Table 5.2 contains the summary of the downloaded data.

5.4 Data pre-processing

Following data pre-processing steps were applied to the downloaded data to reduce its size, correct band units, filter out unusable data, and analyze the distribution of the data.

5.4.1 Cropping

The Sentinel-2 tiles images are huge. It is not practical to keep the whole image on the drive when only a small area of interest is needed later.

Using the Rasterio Python package [29], it is simple to crop each channel of the Sentinel-2 image. To crop a geospatial image properly, it is necessary to know its projection and the projection of the coordinates of the polygon to which to crop. Rasterio then does all the necessary projection transformations under the hood, resulting in a correctly cropped geospatial image.

Cropping an image right after the download of a Sentinel-2 tile image is essential for storing a large number of images of a specific area. The difference in image sizes is captured by figure 5.2.

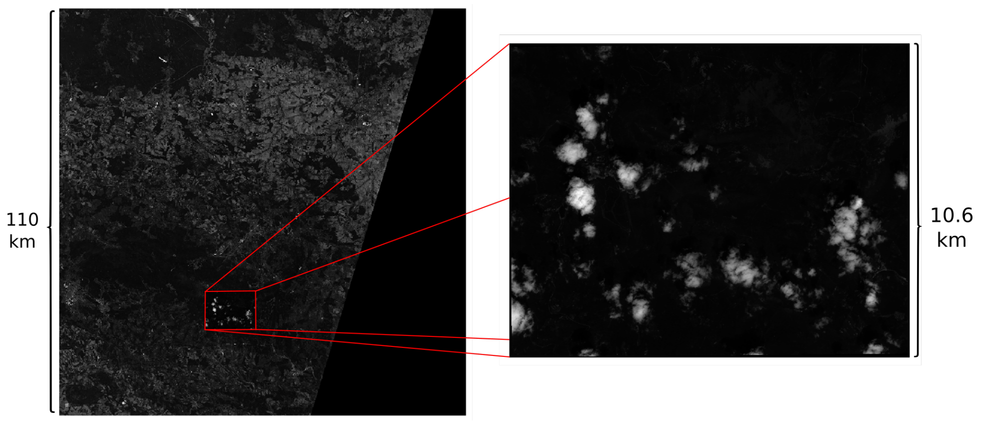


Figure 5.2: Overview of the cropping of Sentinel-2 tile image.

5.4.2 Correcting units

The data for each band of the geospatial image is stored as an array of type uint16. In order to get true sensor data, it needs to be converted to float32 and scaled down by a factor of 10 000.

5.4.3 Detecting clouds

A Sentinel-2 product comes with a computed cloud mask. This is useful for filtering products with limited cloud cover and it gives a rough estimate of how cloudy the satellite image is. The algorithm used by Sentinel-2 is the Sen2Cor algorithm [30]. Although this algorithm provides good accuracy for clear areas (areas without clouds), according the study [31] founded by NASA, the Sen2Cor algorithm performs poorly in identifying cloudy/shadowed observations. The provided Sentinel-2 cloud mask was used only to reduce the number of downloaded images - only images with less than 60 % clouds detected by the default Sen2Cor cloud mask were downloaded. For further processing and classification, a better approach for detecting clouds is needed.

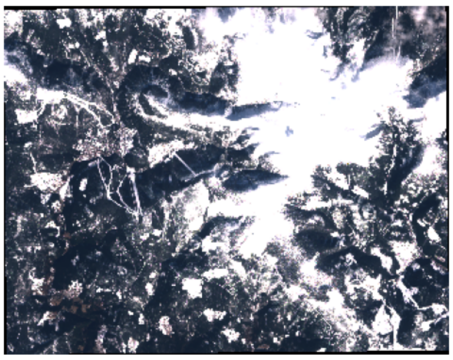
One of the better-performing algorithms used for comparison in the previously-mentioned study [31] is the Fmask algorithm proposed by Z. Zhu and E.

Woodcock [32]. This algorithm showed better accuracy in identifying thin clouds and cirrus. The list of output classes from this classifier is clear, cloud, shadow, snow, and water. Cloud pixels are identified by several spectral-based methods. Using the detected cloud objects, a cloud-shadow mask can be computed using view angles, solar angles, and estimated cloud height. The Fmask algorithm also has a Python interface, which makes it convenient for accurate computation of cloud masks in the experiments chapter 6 of this thesis.

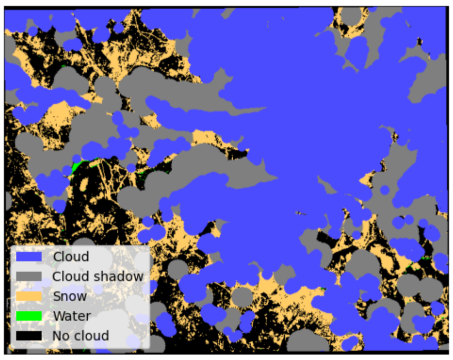
The Sentinel Hub offers a list of public scripts [33] for their application Sentinel EO Browser. This is useful because the scripts are open-source and can be used to process geospatial data even outside of the Sentinel EO Browser web application. These EO Browser scripts work on a per-pixel basis. For each pixel, the script has access to every band of the geospatial image. The output of a script is a three-channel RGB color value, which can then be displayed instead of the given pixel. An example of an EO Browser script is the True Color script, which, for each pixel, returns the values of bands B04, B03, and B02 as red, green, and blue channels. The main advantage of these scripts is their single-pixel computation nature. The computation can be easily parallelized and does not depend on additional information, unlike the Fmask algorithm.

Braaten-Cohen-Yang cloud detector is one of the Sentinel-2 EO browser scripts. The result of the research paper [34], based on which this script is implemented, is that a very accurate cloud classification can be obtained just by simple band indexing and thresholding. The main problem with the Braaten-Cohen-Yang cloud detector turned out to be the winter season. This detector performs poorly in snowy areas because it can not reliably distinguish a cloud from the snow. The nature of our problem, however, requires proper snow detection because the snow is very likely to appear in previously chosen areas of interest.

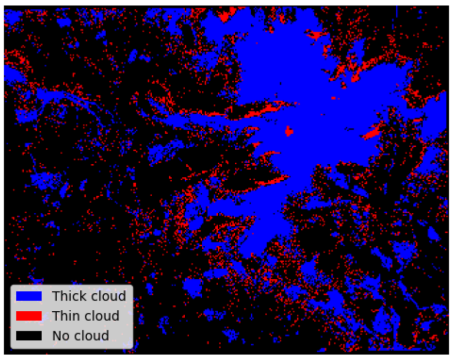
Another could detector from the list of Sentinel Hub scripts is Hollstein's cloud classification script, which is also based on a research paper. [35] This script can differentiate between clouds and now and thus is more suitable for this work. Figure 5.3 shows the difference between Hollstein's and Braaten-Cohen-Yang detectors.



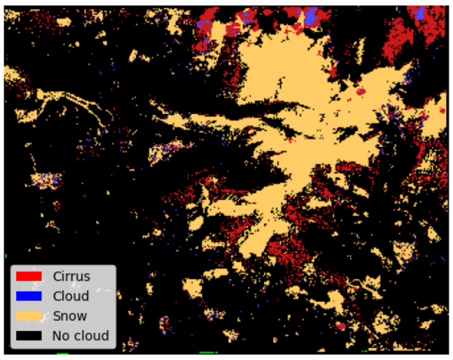
(a) : True color image.



(b) : Fmask cloud image.



(c) : Braaten-Cohen-Yang cloud image.



(d) : Hollstein's cloud image.

Figure 5.3: Comparison of different cloud filter scripts.

5.4.4 Cloud analysis

To get an overview of the distribution of clear and cloudy satellite images throughout the year, the following analysis was conducted.

Firstly, the default Sentinel-2 cloud mask (Sen2Cor) was used to download only Sentinel-2 products with less than 70 % pixels labeled as clouds. From empirical observation of satellite images, it became apparent that products with higher cloud percentages can not be used for meaningful land-cover classification.

After obtaining all Sentinel-2 L1C products satisfying the previously mentioned condition throughout the year 2020 for all three areas of interest, the discussed Hollstein's cloud filter EO script was applied to each image.

All of the output classes of Hollstein's cloud filter script (cirrus, cloud, and snow) completely remove the ability to classify land for such pixels. Therefore, only the percentage of clean pixels is a valuable output of this analysis. The appendix A visualizes the distribution of clean and not-as-clean satellite images produced by the Sentinel-2 mission in the year 2020.

In conclusion, the number of clean images depends heavily on the terrain type and the time of the year. The area in the mountain complex *Krkonoše* contains often only a single clean satellite image each month during spring, summer, and fall. During the winter season, the expected number of usable satellite images for land-cover classification should be minimal. In highlands and flatlands (*Vysočina* and *Morava*), the likelihood of obtaining a clean image for each month throughout the year is higher. During the summer season in these areas, a clean Sentinel-2 image can be obtained every week.

5.4.5 Obtaining features

Following unsupervised and supervised classification methods require a set of observations and their features.

In case of Sentinel-2 imagery, the input observations are the pixels of the satellite image and the features are the wavelength channels of each pixel.

All of the methods used for both unsupervised and supervised classification are able to work with an arbitrary number of features. This means that all of the 12 channels of Sentinel-2 images can be used.

Some channels, however, have different image resolution than others. The sharpest resolution of Sentinel-2 imagery is one pixel per 10 meters squared, the lowest resolution is one pixel per 60 meters squared. This means that all of the channels with lower resolution must be up-scaled without interpolation so that all of the channels have the same amount of pixels in order to create observations with the same number of features each.

The problematic question is, which features are relevant for the classification purpose. The problem of different resolutions can have a negative impact on classification near the edges between two classes in the image.

5.5 Unsupervised classification

In this section, the implementation of three unsupervised classification methods are described in detail.

- The k-means clustering method,
- the Expectation Maximization algorithm,
- and the Simple Linear Iterative Clustering method.

The reason behind doing the unsupervised classification is to obtain a segmentation of the image, which can later be used to either acquire more labeled data for the supervised classification or to post-process the results of a supervised classification in ways like:

- sharpening the edges between two segments of different classes,
- reducing the per-pixel classification noise by labeling whole segments with one class.

5.5.1 K-means clustering

For the implementation of this method, the Scikit-Learn library for Python was used. [36] This library creates an abstraction over the used algorithms and provides an easy to use interface for using the k-means clustering method with many optional parameters.

In this task, an observation is a pixel of the satellite image which consists of 12 floating-point numbers associated with the 12 wavelength bands.

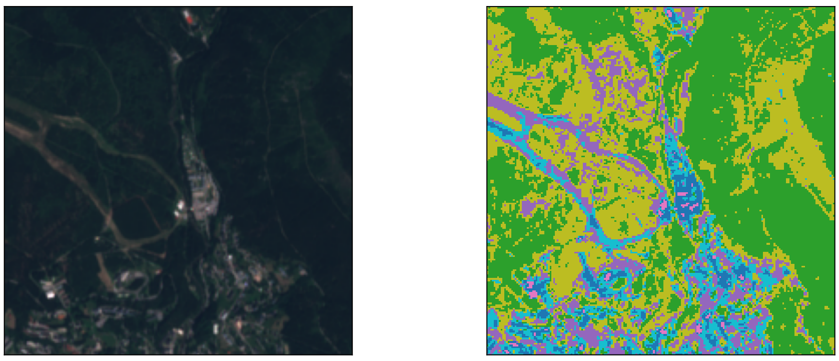
K-means clustering method depends heavily on its initialization method. The method of selecting initial cluster centroids. The k-means ++ initialization method was used in this case. This initialization tries to set the cluster centroids far apart at the beginning, which helps to avoid finding a local minimum.

Once the k-means clustering algorithm converges, it returns K centroids and also labels for each observation.

The centroids can be used as representatives of their corresponding clusters. In case of a satellite image, the centroid is the pixel with channels with an average value in its cluster.

The labels are used to assign an unspecified class to every pixel. Image pixels of the same class have similar channel values and therefore are considered to be pixels of similar meaning in the satellite image.

Figure 5.4 shows the k-means method for five clusters and only some selected channels.



(a) : True color image **(b) :** Labeled image

Figure 5.4: Unsupervised k-means clustering for 6 clusters. The colors of the labeled image are arbitrary.

5.5.2 Expectation maximization

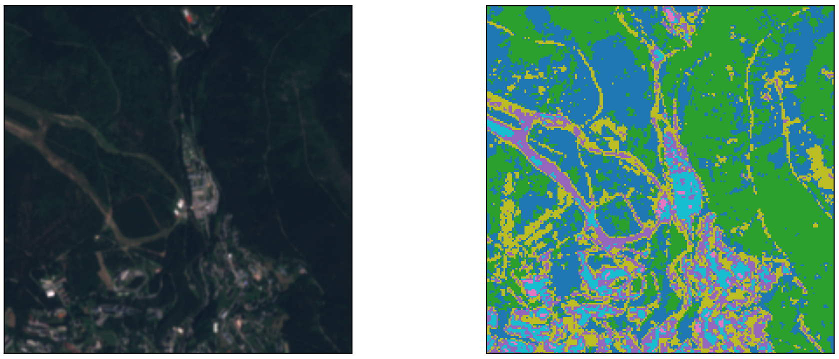
Scikit-learn library contains the implementation of Gaussian-Mixture-Models. One of the methods working with such models is the Expectation-Maximization algorithm.

This method requires one parameter K, the number of clusters (components or classes), for which it tries to fit the Gaussian mixture. K-means clustering method was used as the initialization method for components.

Selecting the number K and the subset of features has again a big impact on the classification result.

The Expectation-Maximization algorithm requires more computation and therefore is slower than the k-means clustering method, however, the EM algorithm models can fit the underlying random distributions of classes and appears to give a better classification results.

Figure 5.5 shows the results of the EM algorithm for 10 clusters.



(a) : True color image **(b) :** Labeled image

Figure 5.5: Unsupervised EM algorithm clustering for 6 clusters. The colors of the labeled image are arbitrary.

5.5.3 Simple linear iterative clustering

For this method, no existing Python library was available. The Skimage library does contain the implementation of this method, however, it uses only the three RGB image channels and a specific measure function to measure correct distances between two LAB color-scheme RGB colors, which makes it perfect for standard image clustering, but not for geo-spatial-image clustering. For this reason, a custom implementation of SLIC needed to be programmed manually. This implementation uses the L2-norm to compute distances between two pixels based on their channel values and thus works with an arbitrary number of channels.

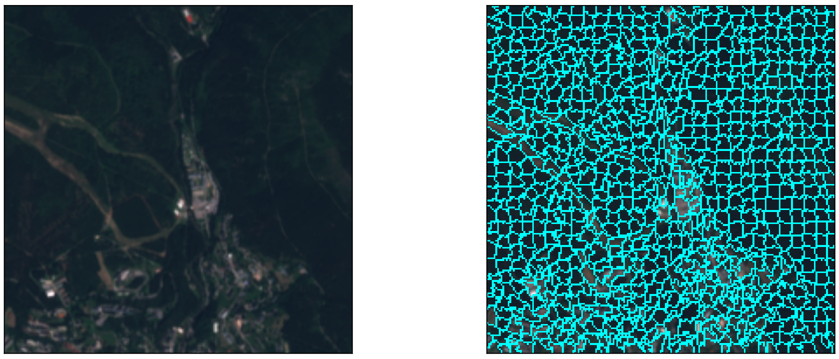
The implementation is written in Python and is used as a proof of concept. There are two ways how to speed up the process:

- write the algorithm in a faster language (like C),
- use multi-threading with the map-reduce paradigm to compute most of the algorithm in parallel.

For this proof of concept, however, the slow implementation is sufficient.

SLIC method requires two parameters. The number of segments is the number of super-pixels, in which the image is segmented. The compactness is the balance between preferring the distance of pixel channel values instead of the spatial distance of pixels in the image for the cluster assignment. Selecting these two parameters depends on the specific image. For purpose of segmentation of Sentinel-2 satellite imagery, the number of segments needs to be very high to obtain very frequent segmentation and the compactness needs to be low, in order to obtain segments that divide areas of similar-looking terrain.

Results of the SLIC method are shown in the figure 5.6. The edges represent a border between two different image segments. These local segments can be used in post-processing to minimize the pixel noise of supervised classification methods.



(a) : True color image **(b) :** Segment boundaries

Figure 5.6: Segmentation of a satellite image using the Simple Linear Iterative Clustering method with 650 segments and compactness of 0.05.

5.6 Supervised classification

In this chapter, the individual steps of a supervised classification pipeline used in the implementation are described.

The pipeline can be summarized into the following parts:

1. gather labels for data,

2. pre-process data so that the specific classifier is able to label it,
3. split data into training and testing set,
4. train the classifier using the training set and predict labels for the testing set,
5. validate results by measuring the accuracy of predictions.

Each individual part is discussed in the following sections.

5.6.1 Gathering labels for data

Because of the lack of resolution of existing Land cover data sets and lack of recently updated regional data sets of selected areas of interest, it was necessary to collect labels for data manually.

Vector method

Using existing tools like QGIS [37] or SNAP [38], it is possible to create polygons over certain areas of a geospatial image. This way, it is straight forward to label some areas with a corresponding class name. Unfortunately, it would, however, take a huge amount of manual work to label the whole area. For this reason, just a little part of the image was labeled to provide labels for at least a few thousand pixels in each of the three selected areas of interest.

Once there are polygons with class labels assigned to them, it is possible to obtain pixels of a geospatial image which are inside these polygons using Rasterio [29] and Fiona [39] libraries. These libraries compute all of the necessary projection calculations to align polygons to image pixels. This process is called rasterization and can also be computed inside the QGIS application.

Raster method and the Labeling application

Manually creating polygonal shapes in any software proved to be too time-consuming for obtaining a large amount of data. For this reason, a different approach was needed. The idea is to use the results of unsupervised classification to help with the creation of a training data-set for the supervised classification.

A GUI application written in Python was created for this purpose. This application can load Sentinel-2 product and apply unsupervised classification using the k-means or the EM algorithm to its red, green, blue, and NIR bands. The result of the unsupervised classification is a labeled image, where each pixel has a label number assigned to it.

Then, the true-color image composite is shown to the user of the application. A new layer representing a class of labels for supervised classification can be created by the user. The user has a circular raster brush tool with variable radius size in pixels. Whenever the user clicks on the true-color image, the pixels inside the brush's circle are selected. Those pixels, which have the same unsupervised label as the clicked pixel are labeled to the active class. Visual feedback of what pixels belong to which class is provided for the user in real-time.

This way, the user simply clicks somewhere in the area designated to the active class and does not have to worry about complex shapes of the terrain, because the shapes are already created by the labeled image of the unsupervised classification.

The output of this application is a set of raster mask files. These are geospatial images with a single band with just one-byte values, 0 - this pixel does not belong to this class, 255 - this pixel does belong to this class. Applying these mask files to the Sentinel-2 band raster files, it is possible to obtain all pixels associated with each labeled class.

Appendix B contains a few selected screenshots of our labeling application.

class name	number of pixels labeled using vector method	number of pixels labeled using raster method
Respondent 1		
Water	9230	10472
ManMade	1104	1260
GroundSoil	423	2085
Grass	847	6607
YellowGrass	1013	1487
DenseForest	6087	36639
ThinForest	1128	9596
GreenField	0	0
YellowField	0	0
Respondent 2		
Water	8144	17362
ManMade	5603	7082
GroundSoil	0	7645
Grass	1448	22719
YellowGrass	0	0
DenseForest	10630	487189
ThinForest	1223	15421
GreenField	3121	8216
YellowField	0	0

Table 5.3: Results of the labeling experiment

Comparison of mentioned methods

To compare both of these methods for obtaining pixel labels, the following experiment was conducted.

Two respondents with a GIS background were given an area of interest. The area is a 100 squared kilometers region in the Jizera mountains. This area was chosen as a validation area for the mountain terrain classifier. For each of the respondents, a satellite image from a different season was provided. The respondents had a time limit to label as many pixels as they could using both the vector method (via the QGIS application) and the raster method (via our labeling application). The time limit was 10 minutes for each method. The respondents were also given a set of possible classes they might use during the labeling. Not every class had to be used.

Table 5.3 shows the number of pixels labeled by each respondent. Note that the result of the vector method is a set of polygons that had to be rasterized in order to obtain pixel counts for this method.

To summarize this experiment, it is clear that the raster method utilized the unsupervised classification labels to speed up the labeling process. Responders reported that the biggest advantage of the raster method was in places where the shape of the terrain was very complex. Defining a proper vector polygon in these places turned out to be too time-consuming.

The disadvantage of the raster method was unsurprisingly our labeling application itself, as it can not compete with the user-friendliness and quick responsiveness of the state-of-the-art QGIS application. However, for this single purpose of obtaining labels, our application turned out to be sufficient. Implementing the logic of the raster method as a plugin to the QGIS application could be a possible solution to the software limitations of our labeling application. That is, however, out of the scope of this thesis and perhaps a project for future work.

5.6.2 k-nearest neighbours classifier

The simplest possible approach of supervised classification used in this project is the k-nearest neighbors algorithm. This method is used as a baseline for other classifiers.

Python's library Scikit-Learn [36] was used as the implementation of this algorithm. The Euclidean metric was chosen as the metric used for computing distances between pixels. The algorithm for an efficient search for nearest points in n-dimensions was chosen to be KD-tree. The parameter, that needs to be found, is the number of neighbors to consider during prediction. This parameter was chosen based on multiple runs of k-nearest neighbors classifier with different K. $k = 3$ performed the best during cross-validation.

This algorithm requires no training computation. This is particularly useful for a baseline classifier, as it can be set up very easily and quickly.

One disadvantage of the k-nearest-neighbors is the required computation for making predictions. Because the classifier needs to find the closest neighbors in all of the training observations, the number of observations and also the number of features of these training observations has a big impact on the prediction computation performance.

5.6.3 Convolutional neural network classifier

For the implementation of convolutional neural network classifier, the Tensorflow Python library was used [40]. This library provides optimized C++ implementations of neural network-related operations available through a Python programming interface. Especially the Keras Python interface makes it very convenient to build a neural network from individual layers.

Input data

In the most common use case of convolutional neural network classifier, the input is the whole image. That is, however, not the case in terrain classification.

The input images are small chunks of pixels in the close neighborhood of the target pixel. The size of the neighborhood determines what possible patterns is the convolutional neural network able to learn.

To obtain the labeled observations, for each pixel with a label, the chunk with a center at this pixel is considered to be an observation. There are two approaches for deciding if the chunk is a valid observation for a label. The first approach requires all pixels inside the chunk to have the same label as the center pixel. The second approach does not take other pixels of the chunk into account and only depends on the center pixel of the chunk. Both approaches were implemented and tested. The second approach turned out to have better results.

The chunk size of 5x5 pixels was chosen for this classifier. The reason behind this choice is that the size of objects and texture patterns in the satellite image at a 10 m resolution is mostly smaller than a chunk of this size. Therefore, the 5x5 pixels chunk provides enough contextual information for the classifier to make a proper decision.

Data augmentation

Because of the simple-pattern nature of the pixel chunk images, it is possible to add artificially created chunk images to the set of labeled observations.

The images contain Earth's terrain viewed from above. This means that they are not prone to rotation and mirroring. For each pixel chunk, its three more rotation symmetries, as well as a horizontal and vertical flip can be added as an observation for the same label. This approach multiplies the number of provided observations by a factor of six.

Layers of the neural network

Similarly to [10] our network topology consists of multiple parallel groups of layers.

Based on the idea proposed in [10], the classifier makes a decision based on the output of several neural networks, where each neural network classifies a chunk image of different size. Using the Keras library, it is possible to combine all of these networks into a single neural network topology by connecting the layers of individual neural networks as parallel groups.

Each layer-group can be viewed as an independant neural network, which can be trained and validated individually.

The first layer-group takes the whole 5x5 chunk as the input and applies a convolution layer followed by a MaxPooling layer. BatchNormalization layer is present in this neural network to reduce the number of training epochs. Full topology of the classifier which contains this layer-group is displayed in figure 5.7.

The second layer-group crops the input chunk by 1 pixel and applies a single convolutional layer to the resulting 3x3 chunk. The convolutional layer has 16 filters followed by the ReLU activation function and BatchNormalization layer. The topology of a neural network classifier that is based on the 3x3 pixels neighborhood is shown in figure 5.8.

Last layer-group consists of a 2 pixel cropping and a basic fully connected dense layer applied to the single remaining multi-channel input pixel. The dense layer has 13 output units and the ReLU activation function. Figure 5.9 presents the topology of this layer.

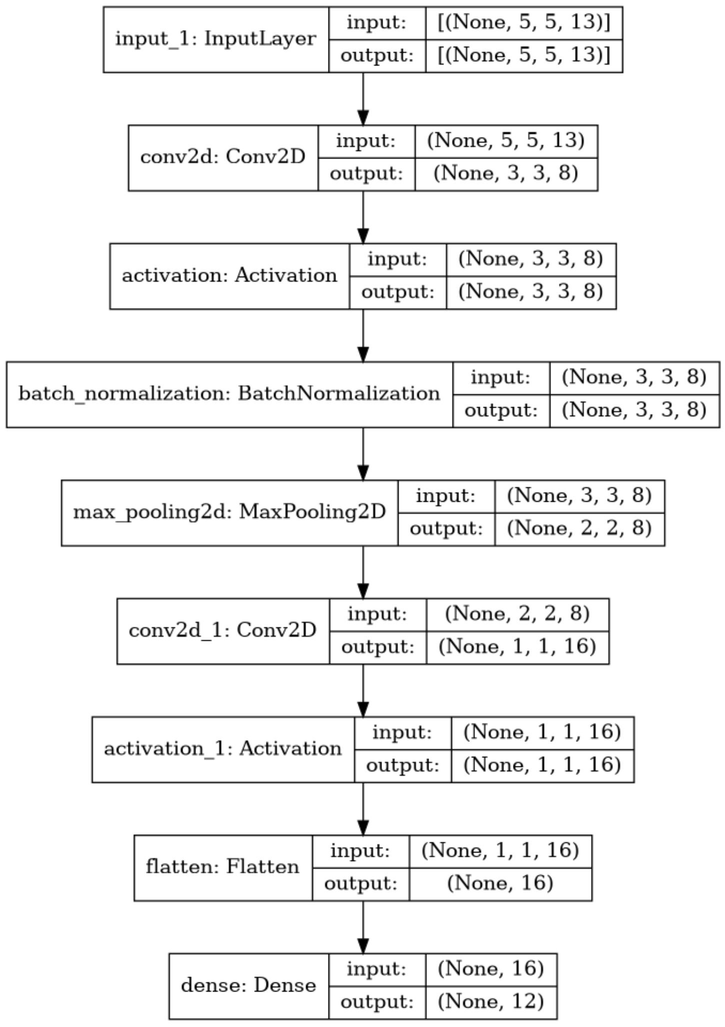


Figure 5.7: Layer topology of the 5x5 pixel chunk neural network classifier.

And finally, the one-dimensional outputs of layer-groups are concatenated into the last fully connected dense layer with the softmax activation function. The final topology of the neural network is shown in figure 5.10.

Training. Training of a neural network requires two additional parameters. It is the batch size - the number of training observations for each forward and backward pass of the network - and the number of epochs - how many times the network needs to see all of the training observations. Each of the previously discussed neural networks was trained for 250 epochs and the batch size was set to 20,000.

The metric used for performance evaluation is the categorical cross-entropy loss. The training set was split into 5 folds, where four of these folds were

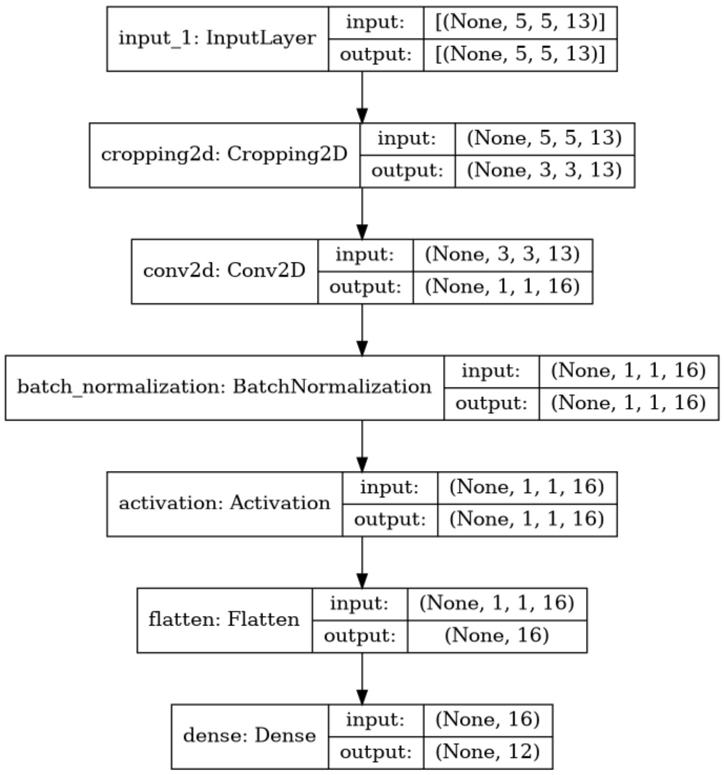


Figure 5.8: Layer topology of the 3x3 pixel chunk neural network classifier.

used to train the parameters of the neural network and the remaining fold was used to monitor its validation loss. A model checkpoint method was used to constantly create snapshots of the parameters throughout the training process after each epoch. Whenever a new lowest validation loss is encountered, a new model of the neural network is saved as the resulting model.

In order to compare previously discussed neural network topologies, a training experiment was conducted. Each neural network was trained for 250 epochs and a batch size of 20000. The learning rate was set empirically to achieve a similar curve of training loss for each classifier. Appendix C visualizes all training histories obtained from this comparison experiment. The training data-set consists of labeled pixels from the first area of interest collected from 7 clear satellite images distributed over spring, summer and fall seasons of the year 2020.

To summarize this experiment, the single-pixel classifier reached the lowest validation accuracy of 93.6 %. The 3x3 chunk classifier scored a better performance of 96.4 %, however, from the plots of appendix C, it is also clear that this classifier started overfitting the training data after the 100th epoch. The 5x5 chunk classifier was the best-performing individual classifier with an overall accuracy of 97.3 %. This classifier showed large differences in

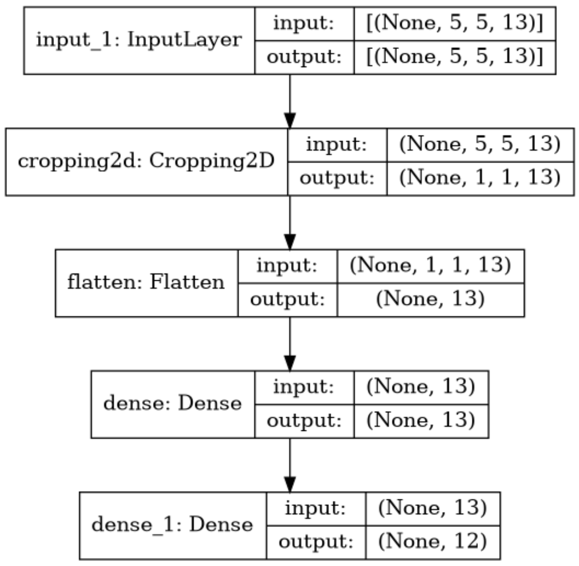


Figure 5.9: Layer topology of the single-pixel neural network classifier.

validation loss throughout the training process. And finally, the combined classifier reported the best overall accuracy of 97.5 %, slightly improving the accuracy of the 5x5 chunk classifier, but also significantly improving the inconsistent validation loss throughout the training process. Also, note that the combined classifier has much more parameters that need to be set during the training process and thus can benefit from longer training. With a smaller training rate of $1 * 10^{-4}$, the combined neural network reached overall validation accuracy of 98.2 % after 1000 training epochs.

Interpreting prediction results. The softmax layer used at the end of the neural network returns, for each class, the probability of assigning that specific class to the input observation. Computing the arg-max of these probabilities gives the final classification label. Computing the maximum of these probabilities gives the confidence, with which the network predicted such a label.

5.6.4 Post processing

Using some methods of the previously described unsupervised classification in section 5.5, it is possible to post-process the labeled image to get rid of eg. per-pixel classification noise. The noise is most noticeable close to the borders between two different classes.

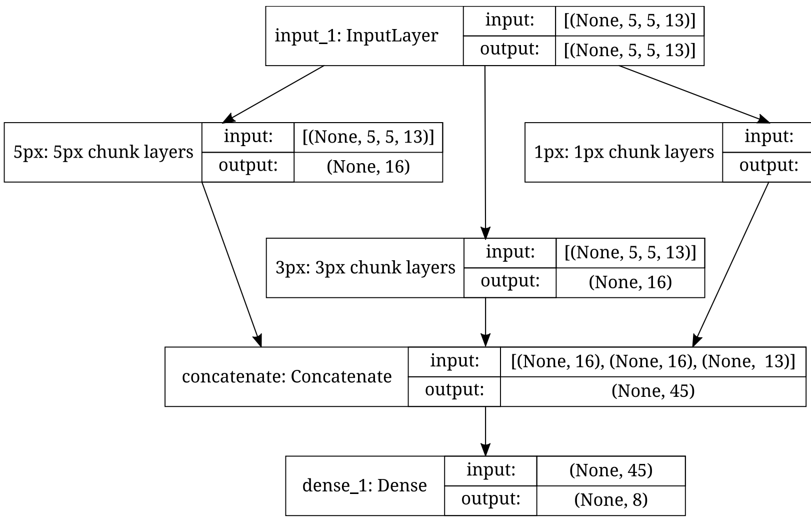
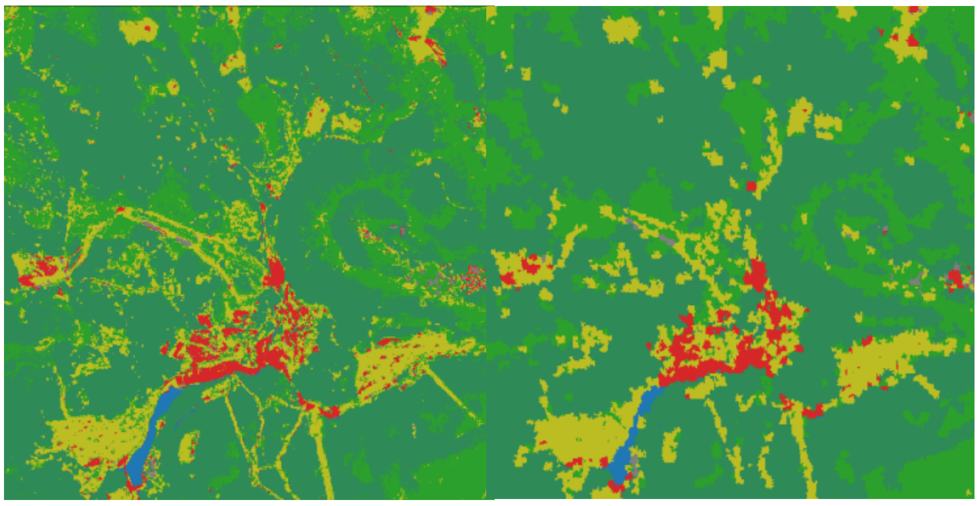


Figure 5.10: Full network topology of the convolutional neural network classifier.

A smoother image can be obtained using the SLIC method mentioned in section 5.5.3. Each segment of the SLIC image is labeled with the most used label of the supervised classification labeled image inside that segment. This removes the single-pixel label noise.

Results of this post-processing method are shown in the figure 5.11.



(a) : Standard result of k-nearest neighbor classifier (no post-processing).

(b) : SLIC post-processing applied to the result of k-nearest neighbor classifier.

Figure 5.11: Visualization of post-processing using the SLIC method.

Chapter 6

Evaluation experiment

The evaluation experiment described in this chapter consists of an accuracy assessment for all of our implemented classifiers from the previous chapter. These classifiers were evaluated in multiple validation areas.

6.1 Experiment setup

This section defines the data-set used for evaluation of implemented classifiers and also the metrics used for the accuracy assessment.

6.1.1 Validation data

Similarly to efficient software testing, the validation dataset must contain a representative set of input data expected from real scenario. [41]

For each of the three selected areas of interest (*Krkonoše*, *Vysočina*, and *Morava*), another area with a similar terrain type and scale was selected as an evaluation area. Table 6.1 lists the areas used for validation.

Type	area (km squared)	distance from training area (km)
mountains	102.77	30.25
highlands	100.94	17.79
flatlands	106.24	13.51

Table 6.1: Geographic properties of chosen validation areas.

Two satellite images from each season of the year 2020 were downloaded from the Copernicus hub. The first image is the cleanest image of the season (with cloud coverage below 1 %), and the second image is a partially clouded image with the Fmask algorithm applied to it for precise cloud detection.

Validation labels were manually created for each image. A total amount of 100 - 300 pixels for each present class.

Experiment classes. The following list of classes was chosen empirically for their easy distinction from Sentinel-2 satellite imagery.

- **Grass** - plains and grasslands with green-colored grass,
- **Water** - lakes, ponds, water reservoirs and large rivers,
- **DenseForest** - deciduous, coniferous or mixed forest areas with green leaves/needles,
- **ThinForest** - bushes and thickets of young trees,
- **BrownForest** - a deciduous forest with dry brown leaves that are either still attached to the trees or laying on the ground,
- **Rock** - rocks and boulders,
- **GroundSoil** - dirt, soil or muddy areas,
- **YellowGrass** - plains and grasslands with dry grass that appears yellow in a satellite image,
- **YellowField** - agricultural fields with yellow-looking plants such as the rape plant,
- **GreenField** - agricultural fields with green-looking plants such as potato fields,
- **PurpleField** - agricultural fields with purple-looking plants such as some specific stages of the red vine field.

Man-made structures. During this experiment, the class of the man-made structures was not considered and a man-made mask was constructed for each image using the data of Open Street Map [42]. The data of man-made structures from the OSM are not perfectly accurate. However, there is low importance of the man-made class for the purpose of our classification task aimed specifically at search-and-rescue operations. For this reason, the OSM mask turned out to be sufficient. Note that in areas, where the OSM mask is inaccurate, the classifier is expected to make mistakes because it is not trained for this type of data. Typically, man-made structures such as parking lots and roads do not reflect any infrared radiation, and therefore have similar properties to rocks and water. The validation pixels selected in areas of interest for this experiment are not located near these man-made structures.

6.1.2 Training data

Training labels for each training area of interest (*Krkonoše*, *Vysočina*, and *Morava*) were created manually using the raster method (5.6.1). An average of 7 cloud-free images for each area were chosen for labeling. These satellite images were taken during seasons spring, summer, and fall of the year 2020.

6.1.3 Classifiers

Two classifiers are compared in this experiment. A per-pixel classifier used as a baseline classifier and a contextual classifier.

The baseline classifier is the k-nearest neighbors classifier discussed in section 5.6.2. Note that similarly to [8], several per-pixel classifiers were tried during the preparation phase of the experiment. Both the support vector machine and the random forest classifiers scored very high overall accuracy on the validation split during training. However, the results from the validation image were, unlike reported in [8], similar or - in some cases - worse than the results of the simple k-nearest neighbors classifier. For this reason, the kNN classifier has been chosen as the representative of the baseline per-pixel classifier for this experiment. During the training process of the k-nearest neighbors classifier, there was an upper bound for the number of training observations set to 5000 observations for each class. These observations were sampled randomly from the training data-set.

The relevant classifier is the convolutional neural network classifier which utilizes contextual information from the 5x5 pixel neighborhood of each pixel. This classifier, discussed in detail in section 5.6.3, is expected to achieve better results in this experiment. The upper bound for the number of training observations for each class was set to 100000. If a class had more training observations in the training data-set (e.g. the DenseForest class), then the observations were sampled randomly. In the other case, the data augmentation step was applied to produce up to eight times more training observations.

Classifiers were either trained using the whole training data-set - in this case, we refer to these classifiers as annual classifiers - or, only a subset of training data from a specific season of the year was selected for training - then we refer to them as seasonal classifiers.

6.1.4 Accuracy assessment

Accuracy assessment is a way of evaluating the performance of a classifier for a given task. In this experiment, we are interested in several assessment metrics for each classified task, as well as overall metrics, which can be used to compare two classifiers.

After a classifier predicts labels for the satellite image, the predictions are compared with the true validation labels constructed for the image. A confusion matrix can then be computed. Using the confusion matrix, the following assessment metrics can be acquired:

Support. Not a metric by itself, support has a great influence on upcoming metrics and is therefore included in the accuracy assessment for each class. Support states the number of validation pixels labeled as a specific class.

Precision. [43] defines precision as:

$$P = \frac{\#(relevant\ items\ retrieved)}{\#(retreived\ items)}.$$

In our case, there is a number of labeled validation pixels. Precision can then be interpreted in the following way: from all of the validation pixels that the classifier predicted as a specific class, what percentage of them was correctly predicted as such class?

Recall. Formally defined in [43] as

$$R = \frac{\#(relevant\ items\ retrieved)}{\#(relevant\ items)}$$

states what percentage of validation pixels truthfully labeled as a specific class was correctly predicted as that class by the classifier.

***F*-measure.** A single measure that trades off precision versus recall is the F-measure, which is the weighted harmonic mean of precision and recall. In this experiment, precision and recall are weighted equally, giving us an F_1-measure with a following simplified formula:

$$F_1 = \frac{2 \cdot P \cdot R}{P + R}.$$

Overall accuracy. The simplest metric for evaluation of a whole classifier is the overall accuracy. In our experiment, it denotes the number of correctly predicted validation pixels by the classifier divided by the number of all validation pixels. This metric is useful for a quick assessment of the situation, however, it does not take various factors (such as imbalanced support of individual classes) into consideration and thus may lead to misleading results.

Kappa statistic. As a common measure for agreement between judges in the social sciences, the kappa statistic measure is used in categorical classification tasks to compare the observed accuracy of a classifier with an expected accuracy of a random-chance classifier. The formula for computing the kappa statistic stated in [43] is:

$$\frac{P(A) - P(E)}{1 - P(E)},$$

where $P(A)$ is the overall accuracy of a classifier and $P(E)$ is the expected accuracy that a random classifier would achieve based on the confusion matrix. The value obtained by kappa statistic is equal to one, if the classifier predicts everything perfectly, zero if the classifier is no better than a random-chance classifier would be, and negative if the classifier is even worse than random predictions. [43] also mentions, that values above 0.8 are considered as a *good agreement* (meaning that the classifier shows good performance in this classification task), values between 0.67 and 0.8 are a *fair agreement*, and values below 0.67 suggest that the classifier may not be the correct solution for this task. These thresholds should, however, only be considered as a rule of thumb because precise interpretations depend on each classification task.

6.2 Experiment results - clean images

This section briefly concludes the results of the evaluation experiment. A more in-depth description of results obtained from individual areas can be found in appendix D. The complete list of evaluation statistics is listed in appendix E.

In general, our implemented classifiers achieved decent overall accuracy during the spring and summer seasons. The kappa statistic during these seasons was near the edge between *good agreement* and *fair agreement* defined in the previous section. A noticeable deterioration of accuracy was reported by the results from the fall season - still, however, in the range of *fair agreement* of the kappa statistic. Meaningful classification of snowy images during the winter season turned out to be impossible using our classifiers.

The differences in overall accuracy scored in the accuracy assessment between our implemented classification methods were negligible. However, upon further investigation of confusion matrices, the CNN classifier reported less confusion among classes that require good separation for the purpose of search-and-rescue operations. The output of the contextual CNN classifier produces a smoother labeled image as shown in figure 6.1. This image also shows how the accuracy assessment may end up with very similar results even when the images differ. The CNN classifier is also able to produce a confidence image - how certain its prediction is for each pixel - that can be further used in rescue planning systems. These mentioned reasons make the CNN classifier a better solution for this task.

6.3 Experiment results - cloudy images

For this experiment, partially clouded images were downloaded and used for classification. The Fmask algorithm was used to create a no-data value masks in places, where a cloud or a cloud's shadow is detected. It became clear, that the classifier works in the exactly the same way as for the cloud-free images in places not affected by clouds, and that the classifier predicts wrong labels in places where the clouds interfeared with the satellite images. Therefore, creating the accuracy assessment using a subset of validation pixels which were not covered by clouds would not so much validated the accuracy of the classifier, as it would asses the accuracy of the Fmask algorithm. Validating the accuracy of the Fmask algorithm is not in scope of this thesis. However, a conclusion can be made from observing labeled images produced by the classifiers for partially clouded satellite imagery.

Figure 6.2 shows an example, where the output of the Fmask algorithm did not successfully label all cloud shadow areas. As a result of the misclassification of the Fmask, our classifier confused an area of **DenseForest** in the shadow of a cloud for a body of water. In places, labeled by the Fmask algorithm as clear, the predicted labels seem to be relevant and an accuracy assessment - using validation pixels in places where the output of the cloud mask was verified by a human - achieved similar results to a cloud-free image. The problem is, that we do not have the information about cloud mask failures in advance.

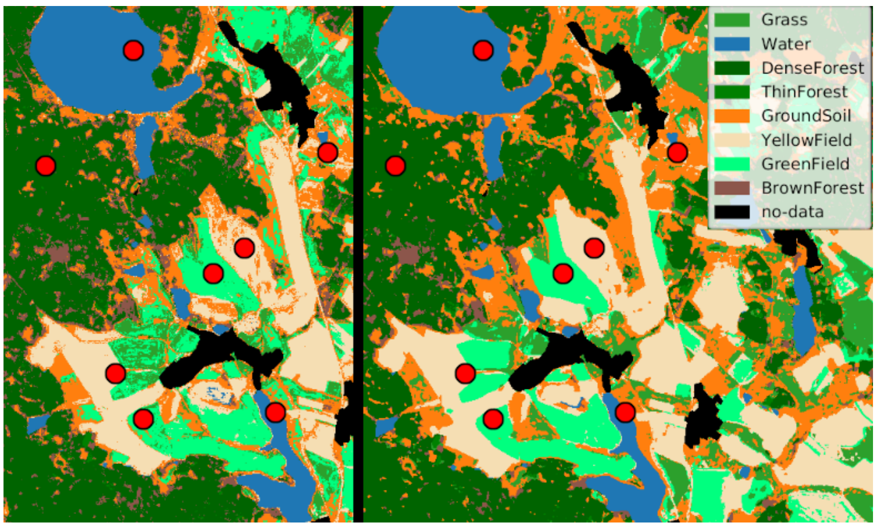


Figure 6.1: The labeled image of the kNN classifier (left) compared to the labeled image of the CNN classifier (right). Red dots mark points selected for accuracy assessment.

From empirical observations, the Fmask algorithm successfully identified cloud objects in the majority of cases. The predicted shadows cause most of the classification problems. This suggests that a for example a 50% clouded image, with most of the clouds located on one side of the image, does not necessarily need to be discarded. The pixels far enough from the clouds can be classified correctly.

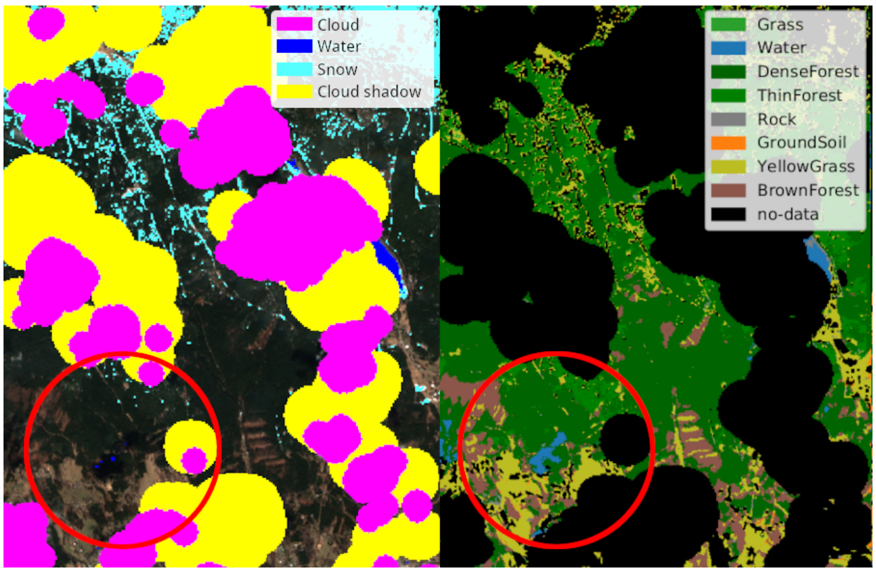


Figure 6.2: The output of the Fmask algorithm overlayed on top of a true-color image (left) compared to the labeled image predicted by a CNN classifier (right). The red circle marks an undetected shadow area.

Chapter 7

Conclusion

In this thesis, we described our land-cover classification task and provided a report of available data source which can be used to implement a solution. We focused our thesis on Sentinel-2 satellite imagery classification and described the properties of geospatial images. We provided an analysis of existing methodology and datasets for satellite imagery land-cover classification.

Several algorithms for supervised classification were introduced and then implemented using Python's libraries.

An experiment was conducted to provide an accuracy assessment of implemented classifiers in several areas of interest including mountains, highlands, and flatlands. We achieved satisfactory accuracy results using a manageable amount of manually created training data.

Classifying images from the fall season turned out to be less accurate than in the spring or the summer seasons. During the winter season we were unable to provide meaningful classification results.

Our methods need to have man-made structures, clouds, and snow filtered out. Possible improvement of detecting man-made structures can certainly be made using more precise datasets than the convenient Open Street Map dataset used in our experiment. Classifying areas near detected clouds turned out to be unreliable. The snow also caused lots of problems as it was often confused with clouds. Further research of snow detection methods needs to be conducted in this regard.

Based on the results of the experiment, a field test may be executed in the future by the search-and-rescue services of the Czech Republic. To achieve high accuracy of a classifier, the classifier needs to be trained using as close to the target image as possible. This suggests creating specific classifiers for specific areas with a high frequency of search-and-rescue operations. Also, seasonal classifiers - classifiers trained using images from a specific season of the year - yield better results than annual ones. For this reason, a significant amount of manual labeling must be done by the search-and-rescue services beforehand.

With seasonal models trained for specific areas, our method can provide reliable and recent information about the terrain to the *Pátrač* rescue planner system and thus lower the time of the planning phase of these operations.

Appendix A

Cloud analysis results

The following plots show the results of cloud analysis of Sentinel-2 satellite images throughout the year 2020 in three areas of interest. For further details about the analysis, see section 5.4.4.

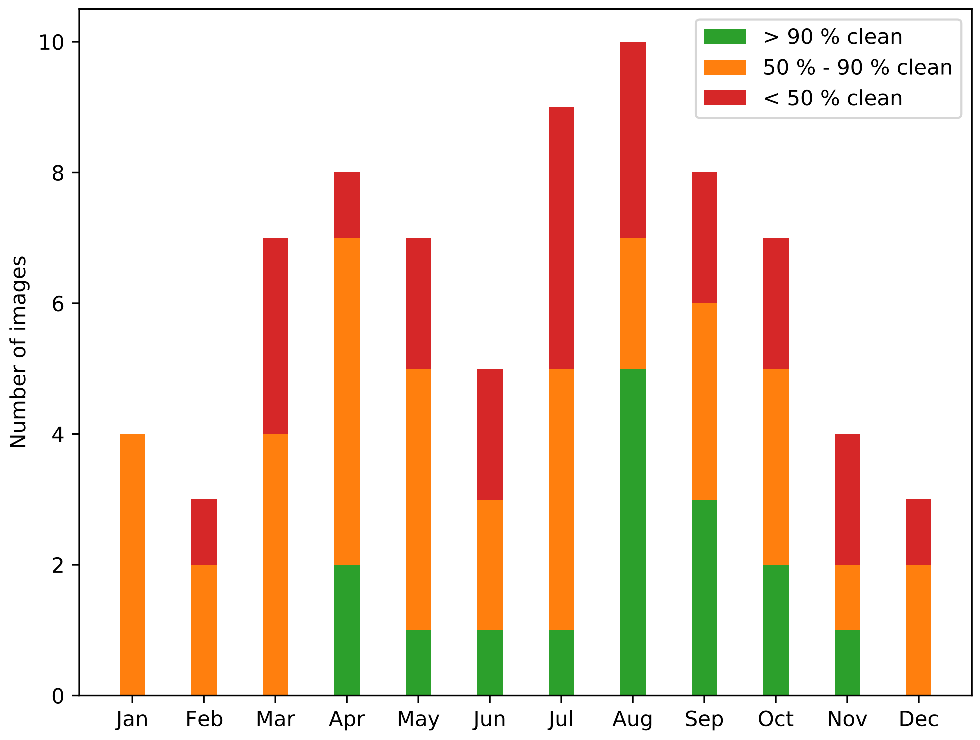


Figure A.1: Distribution of clean satellite images of the area of *Krkonoše*.

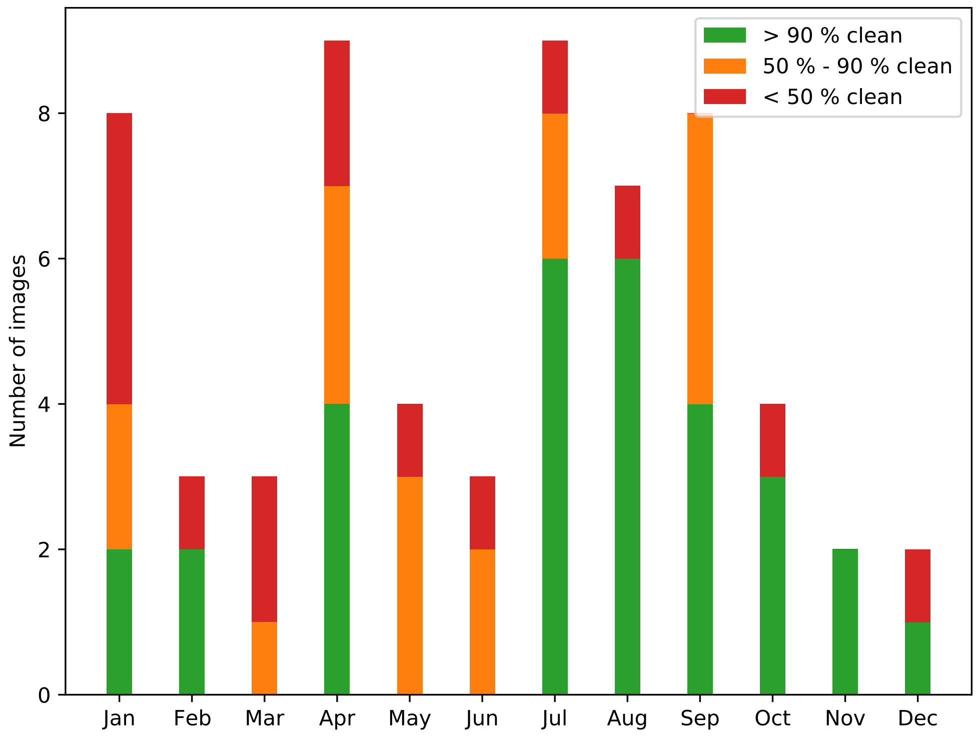


Figure A.2: Distribution of clean satellite images of the area of *Vysočina*.

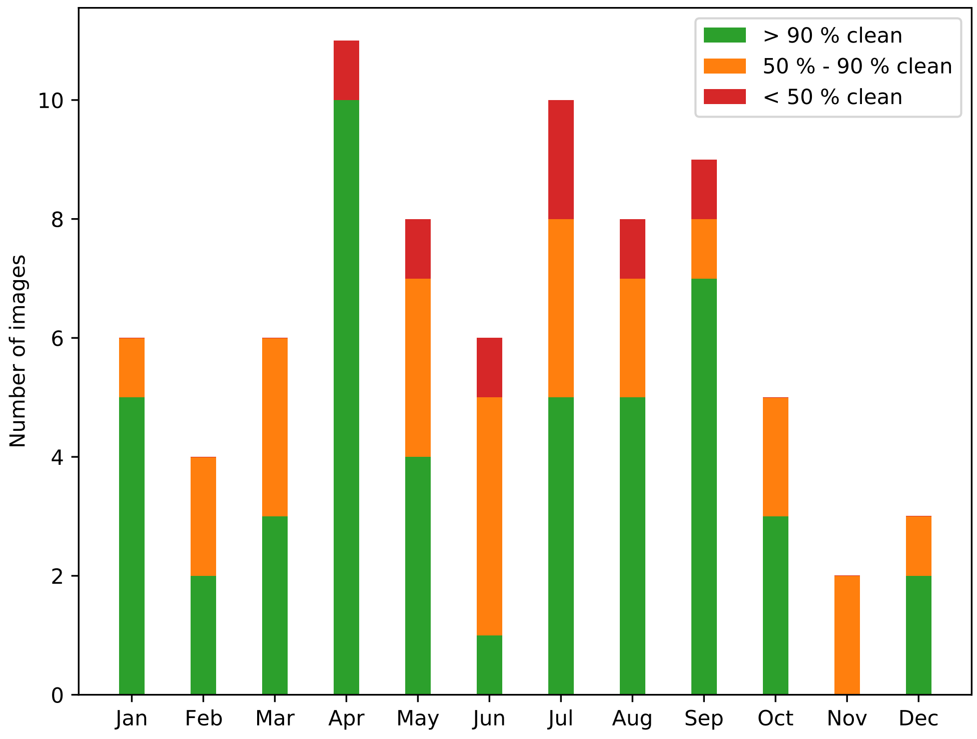


Figure A.3: Distribution of clean satellite images of the area of *Morava*.

Appendix B

Labeling application screenshots

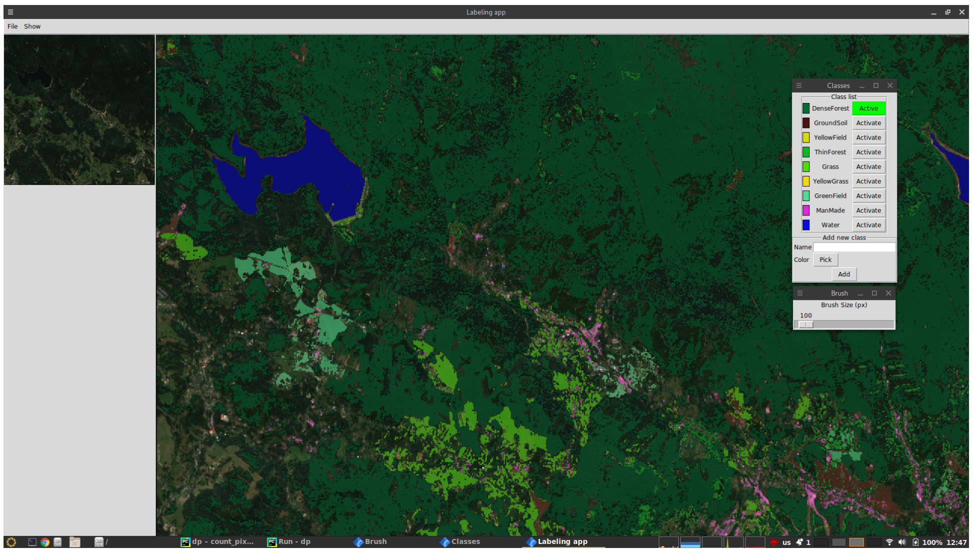


Figure B.1: Class overlay on top of a true-color image.

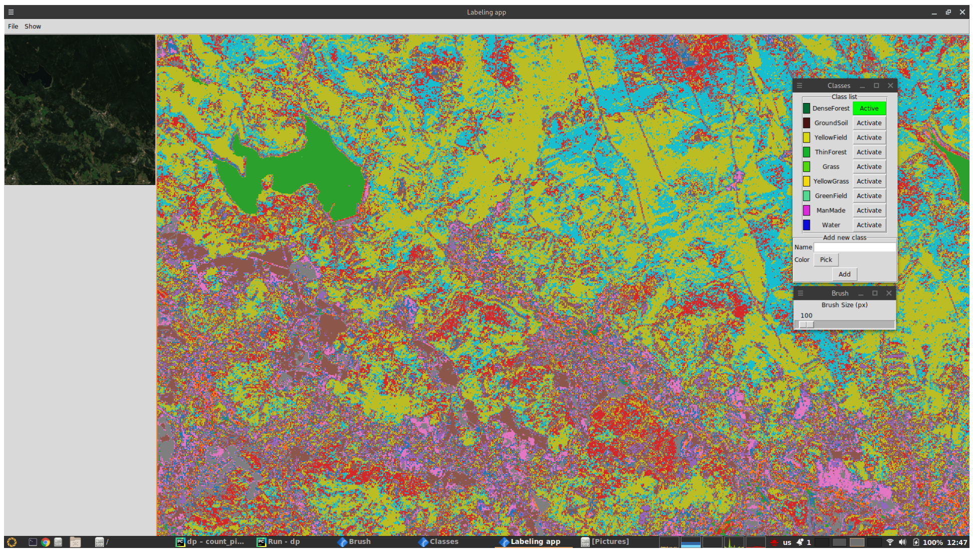


Figure B.2: Labels of the unsupervised classification. Color of each label has no particular meaning.

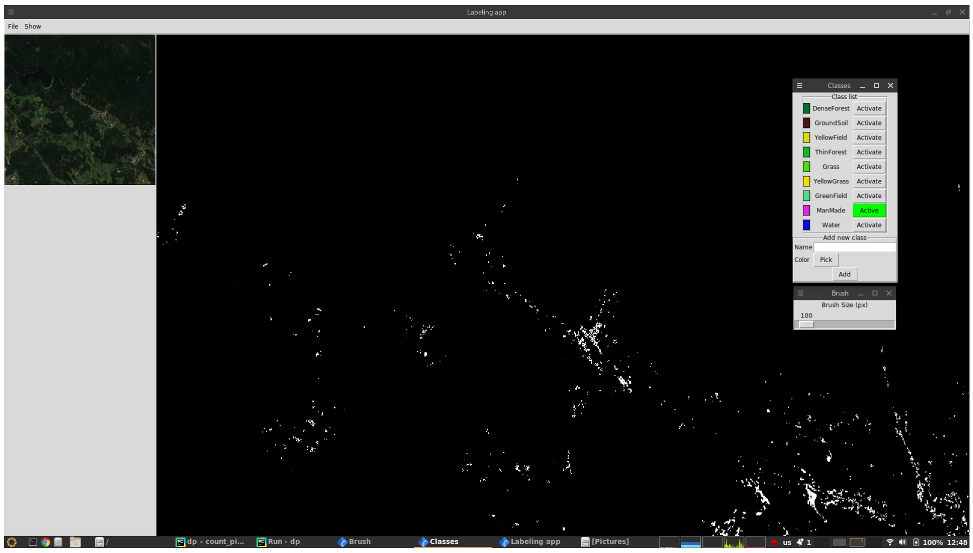


Figure B.3: Binary mask marking all pixels that belong to the ManMade class.

Appendix C

CNN training insight

The following plots show the history of the training process of neural network classifiers used for this task. See section 5.6.3 for more information regarding the training process.

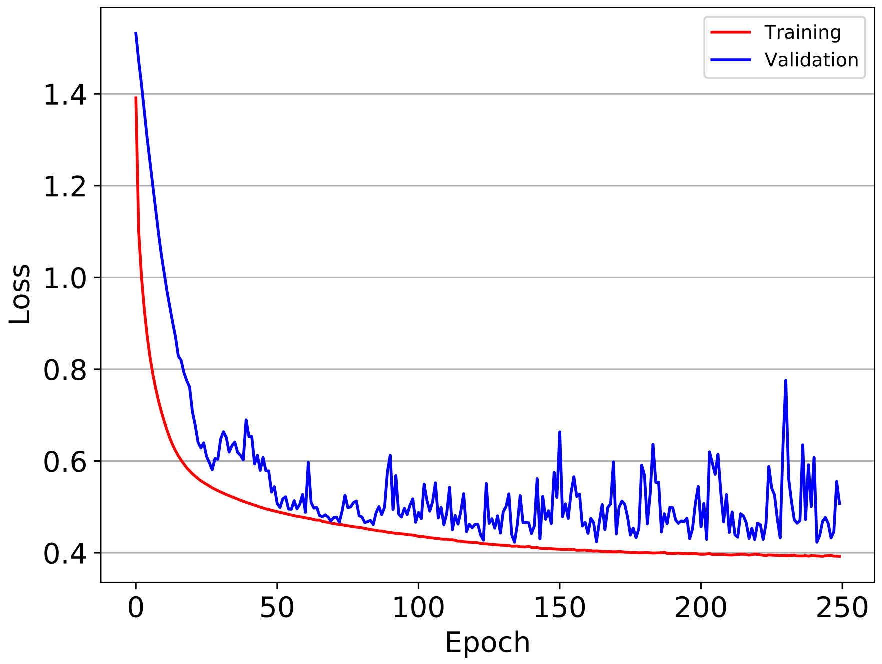


Figure C.1: History of the training and validation loss metric of the 5x5 pixel chunk neural network classifier throughout the training process.

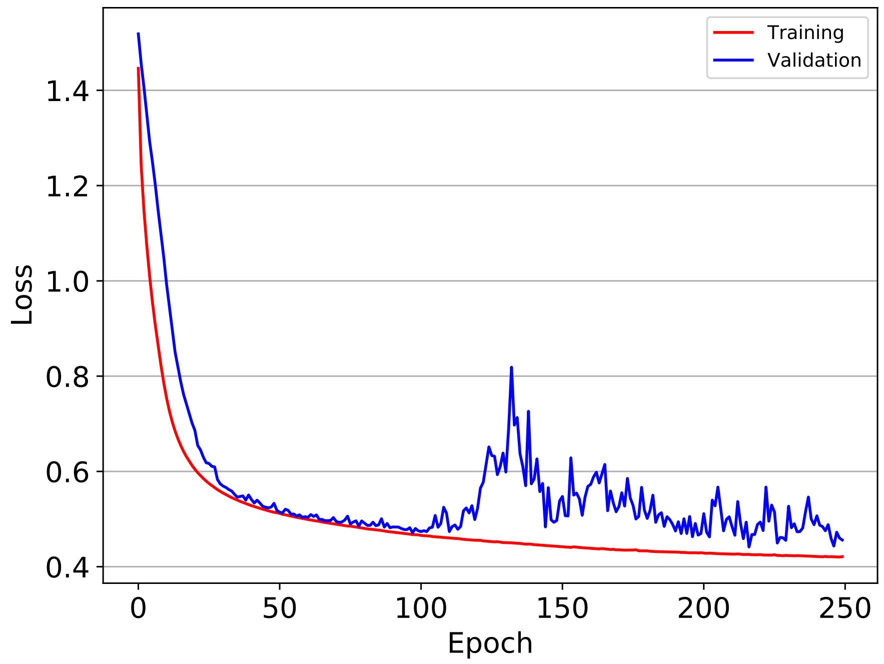


Figure C.2: History of the training and validation loss metric of the 3x3 pixel chunk neural network classifier throughout the training process.

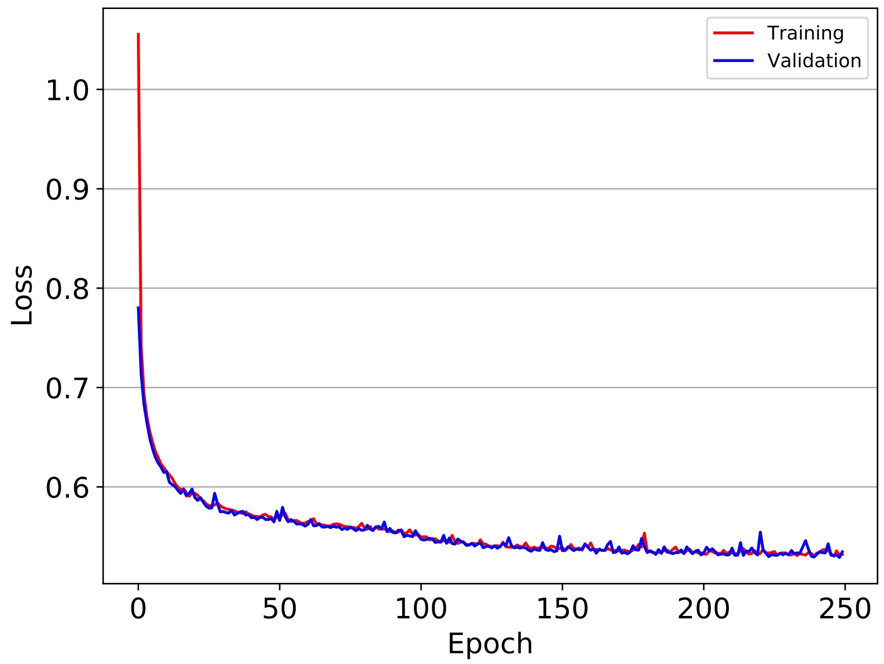


Figure C.3: History of the training and validation loss metric of the single-pixel chunk neural network classifier throughout the training process.

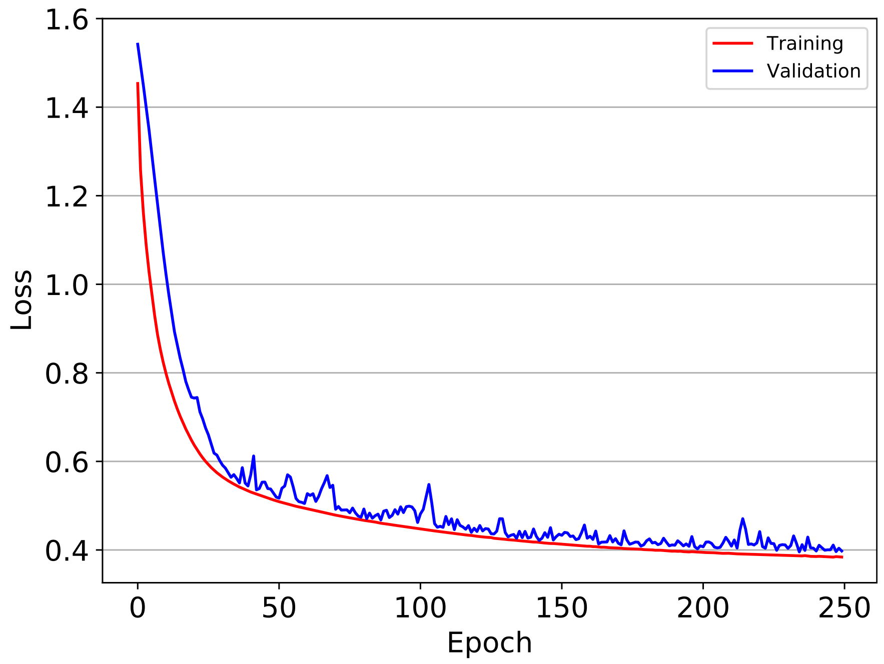


Figure C.4: History of the training and validation loss metric of the combined neural network classifier throughout the training process.

Appendix D

Discussion of experiment results

D.1 Experiment results - mountains

In this section, the results of the most significant area of the experiment - the mountains - are discussed in detail. The complete list of all results obtained from this experiment can be found in appendix E.

D.1.1 The spring season

As an introduction for this season, let us present the side-by-side comparison of the labeled image produced by the annual CNN classifier with the true-color image of the same area. Figure D.1 shows promising results of this classifier.

Firstly, the annual k-nearest neighbor classifier was validated using the manually labeled validation pixels. From the confusion matrix displayed in figure D.2 and the corresponding statistics table from appendix E a conclusion can be derived. The k-nearest neighbor classifier scored an overall accuracy of 85 %, which is a satisfying result. Some problematic misclassification was, however, revealed by the confusion matrix. The class **ThinForest** was often classified as **Grass**, which would have a negative impact on derived terrain accessibility for a search-and-rescue operation. The results also reveal a complete misclassification of the **BrownForest** class for the **GroundSoil**. On the other hand, classes **Water** and **DenseForest** show impressive precision and recall, achieving above 0.99 score of F_1-measure.

The annual CNN classifier appears to achieve almost no improvement over the kNN classifier as the overall accuracy improved by only 1 %. The confusion matrix, visualized in figure D.3, reveals several benefits of this CNN classifier over the annual kNN one. The confusion between classes **BrownForest** and **YellowGrass** is problematic, however, the vast majority of mistakes were made by confusing classes **GroundSoil**, **Grass**, and **YellowGrass**. All of these classes have similar accessibility from the rescue-operation's point of view. The confusion of these different types of grass is probably caused by the various states of grass over the year. This makes it difficult to completely distinguish these classes when creating the training data-set.

An honorable mention goes to the seasonal CNN classifier trained using satellite images taken during the spring season. Even though the overall accuracy drops to 84 % with this classifier, the confusion matrix in figure D.4 reveals interesting results. The confusion of various grass types is mitigated by this classifier. Classes **Grass** and **YellowGrass** are distinguished well. On the other hand, class **GroundSoil** was not recognized by the classifier at all. The reason behind this is that the area of the training data-set may have different properties of the soil located there, especially during the spring season. For this reason, the training data provided for the seasonal CNN classifier did not represent the validation area well enough.

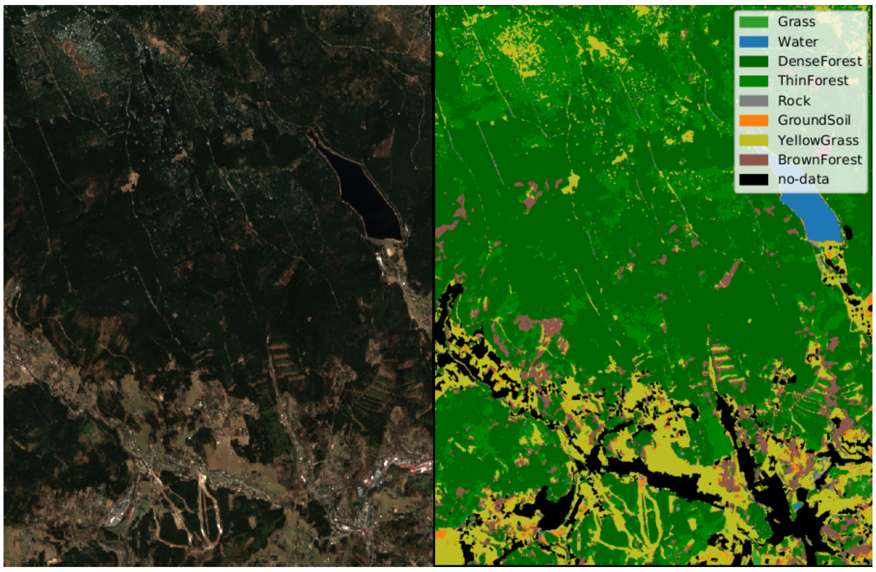


Figure D.1: True color composite of a satellite image of the mountains area taken in April (left) compared to the labeled image output of the annual CNN classifier (right).

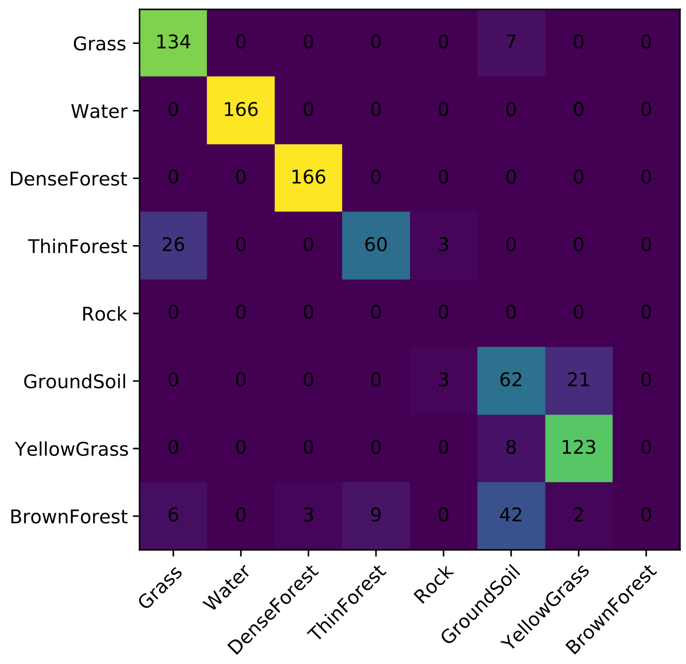


Figure D.2: Confusion matrix. Model: kNN, type: annual, season: spring, area: mountains.

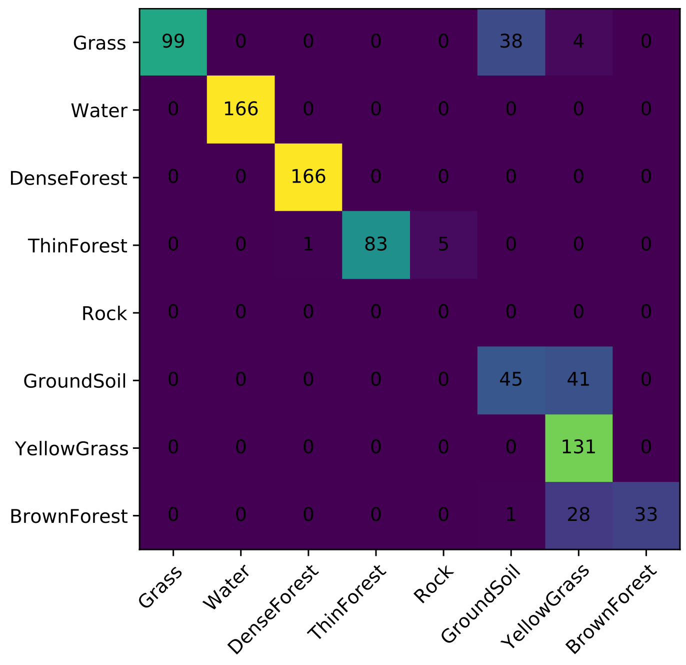


Figure D.3: Confusion matrix. Model: CNN, type: annual, season: spring, area: mountains.

D.1.2 The summer season

The summer season has the highest number of available cloud-free images for classification. The vegetation has high levels of chlorophyll, which means that the infrared light has a more significant impact on the distinction of individual classes. Also, classes such as **YellowGrass** and **BrownForest** are much less frequent during this season and are in fact completely missing from the list of classes used for the set of validation pixels, because no such areas were found in the validation image. For this reason, seasonal classifiers are expected to outperform annual ones.

The annual kNN classifier reached an overall accuracy of 84 %. Similarly to the spring season, this is a satisfying outcome.

In this season, the annual CNN classifier outperformed the kNN classifier substantially with an overall accuracy of 88 %. Figure D.5 reveals the reason behind the performance of the CNN classifier. Apart from a little confusion between **Grass** and **GroundSoil**, the relevant classes are separated very well. The **YellowGrass** class, on the other hand, achieves zero precision, because no **YellowGrass** pixels were found in the validation image. A side-by-side comparison of the true-color satellite image and a label image produced by an annual CNN classifier is shown in figure D.6.

The seasonal classifiers kNN and CNN were not trained with any **YellowGrass** or **BrownForest** data and thus achieve higher overall accuracies of 91 %, and 93 % respectively.

D.1.3 The fall season

The results from the fall season are not as convincing as the results from the summer season. Because of the weather conditions during the fall season, only two clear satellite images were obtained for the training dataset. This could have affected the results of this experiment in a negative way.

Both the kNN annual classifier and the CNN annual classifier showed significant misclassification of the **ThinForest** class, confusing it mostly with the **Grass** class. Also the problem with several types of grass - discussed in the spring season section - returns in the fall season as well. This is also the

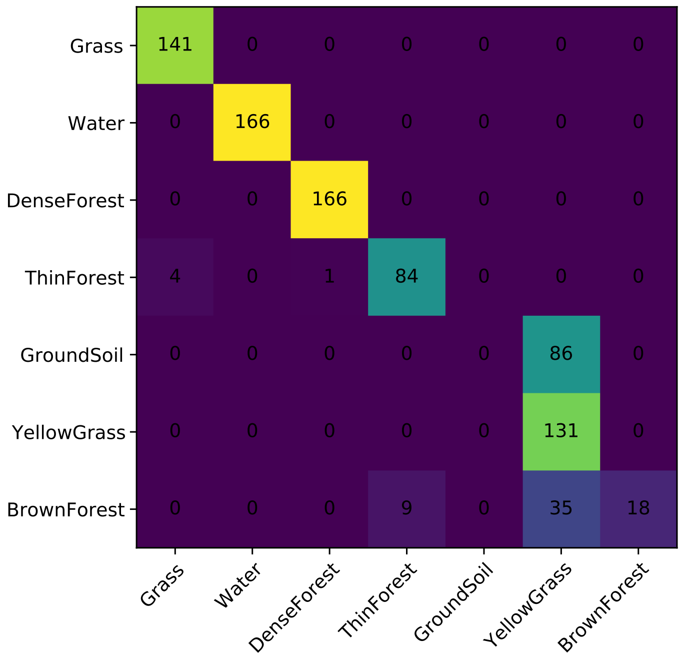


Figure D.4: Confusion matrix. Model: CNN, type: seasonal, season: spring, area: mountains.

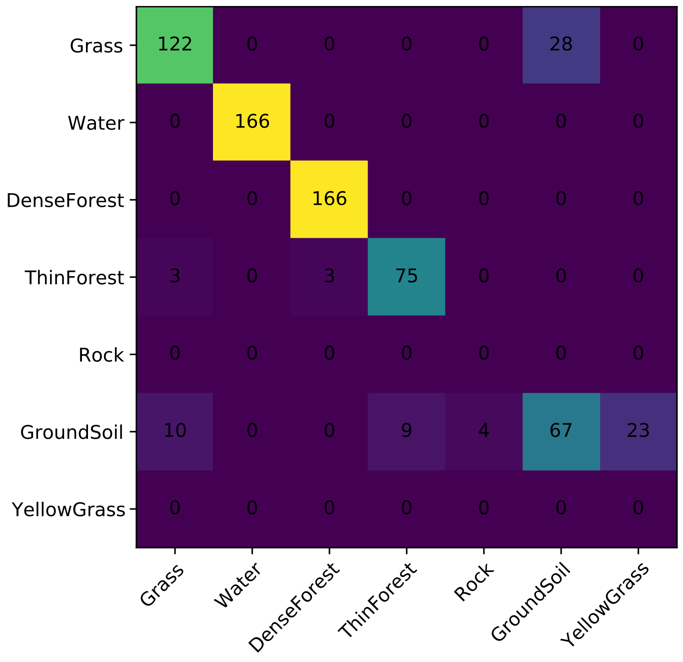


Figure D.5: Confusion matrix. Model: CNN, type: annual, season: summer, area: mountains.

only moment when the kNN classifier outperformed the CNN classifier in the mountains area (kNN's overall accuracy was 79 % and CNN's was 76 %).

Again, the seasonal models showed better performance. The confusion between **Grass** and **ThinForest** was mitigated by both of the kNN and CNN classifiers. **Grass**, **YellowGrass**, and **GroundSoil** were the most misclassified classes, which makes the performance of these classifiers even better for our classification task. Kappa statistic values above 0.83 suggest that these classifiers output relevant predictions. Both classifiers achieved a decent overall accuracy of $\tilde{8}8$ %. Figures D.7 and D.8 visualize the comparison between seasonal CNN classifier and an annual CNN classifier.

D.1.4 The winter season

Satellite images taken during the winter season in the mountains contain the element of snow. This and the very small number of cloud-free images from this season lead to a very unpleasant conclusion that a meaningful classification of terrain types is impossible during the winter season.

Decently accurate detection of snow and water using the Fmask cloud-detection algorithm or the Hollstein's filter is possible. Cloud-free areas that are not labeled as snow or water are usually forests. However, this depends on the amount of snow and the amount of time since the last snowstorm, as even forest areas can be completely covered by the snow.

D.2 Experiment results - highlands

Experiment results from highlands are discussed in this section. This area contains numerous forests and - unlike the previous mountains area - it also consists of many agricultural fields.

D.2.1 The spring season

In the spring season, both annual classifiers managed to separate almost all of the classes very well. The exception being the distinction between **Grass** and

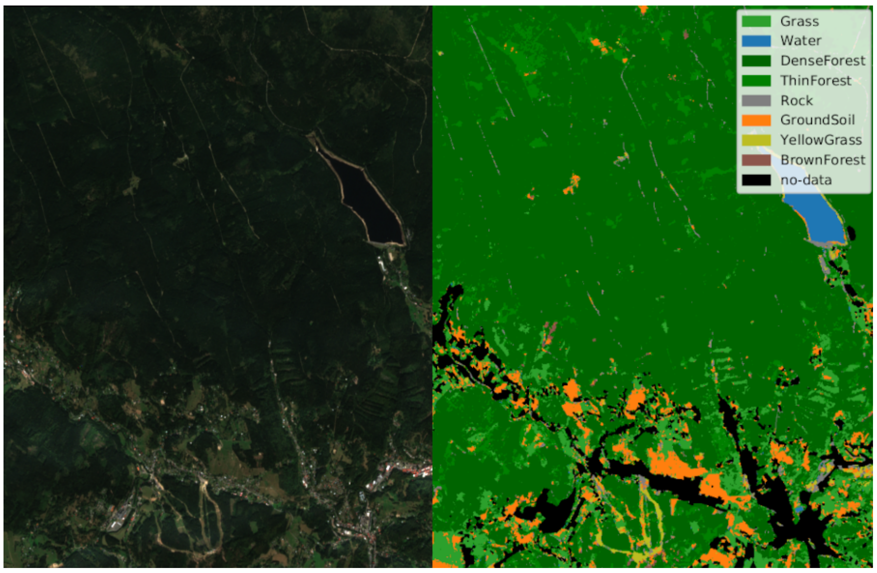


Figure D.6: True color composite of a satellite image of the mountains area taken in August (left) compared to the labeled image output of the annual CNN classifier (right).

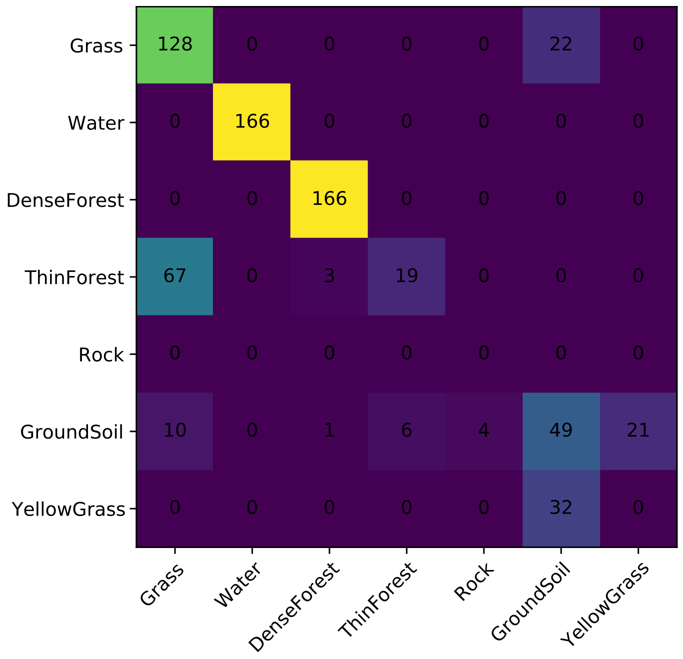


Figure D.7: Confusion matrix. Model: CNN, type: annual, season: fall, area: mountains.

GreenField classes, where there was a lot of confusion in both classifiers. Slightly better overall accuracy was achieved by the annual *k*NN classifier. The confusion matrices of both classifiers were extremely similar. Figure D.9 shows the one associated with the *k*NN classifier.

Again, the seasonal classifiers achieved better classification results. In this case, the CNN classifier scored a higher overall accuracy of 93 %, which is a remarkable performance. Figure D.10 shows that the **GreenField** areas were no longer as misclassified for the **Grass** class.

D.2.2 The summer season

The annual *k*NN classifier achieved surprisingly bad precision for the class of **ThinForest** as it confused it a lot with **GreenField** (as shown in figure D.11). Despite having better overall accuracy than the annual CNN classifier, the results of the CNN classifier turned out to be more convincing.

Surprisingly, in the summer season, the seasonal classifiers achieved slightly worse overall accuracy. The better performing classifier was the seasonal CNN. This classifier confused mostly classes **GreenField** and **Grass** and also classes **YellowField** and **GroundSoil**. The complete confusion matrix of the seasonal CNN is visualized in figure D.12.

The fall season

The overall accuracy of annual classifiers for the fall season was very similar to the summer season (around 85 %). However, a new confusion was brought by this season. Both CNN and *k*NN classifiers achieved poor precision for the **ThinForest** class, making wrong predictions mostly for areas labeled as **Grass** or **GreenField**. This can be seen in figure D.13 which contains the confusion matrix of the annual CNN classifier.

Both seasonal classifiers achieved poor performance for this season. The most likely explanation is that the training images associated with the fall season contained slightly different growing stages for classes like **Grass**, **YellowField**, **GreenField** and **ThinForest**. The problem in this season is that the vegetation changes rapidly. Deciduous forests with green leaves turn brown, the grass loses its saturation and the fields turn from **YellowField** to

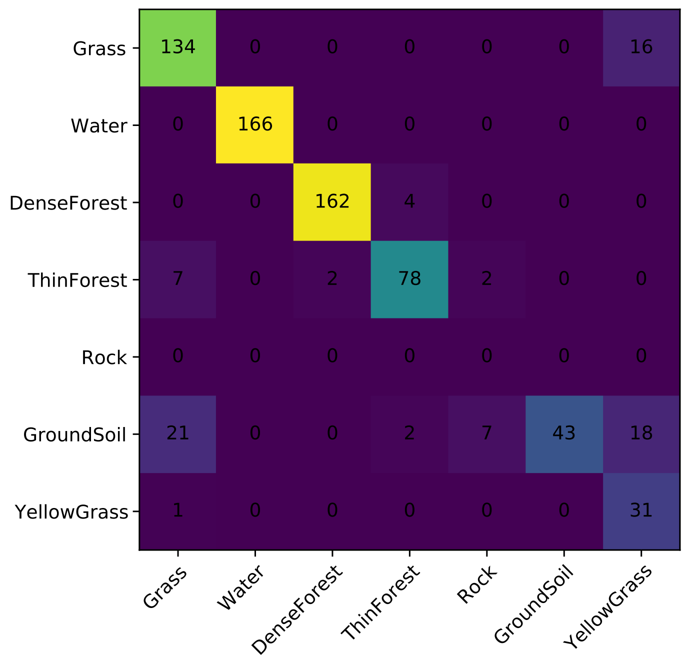


Figure D.8: Confusion matrix. Model: CNN, type: seasonal, season: fall, area: mountains.

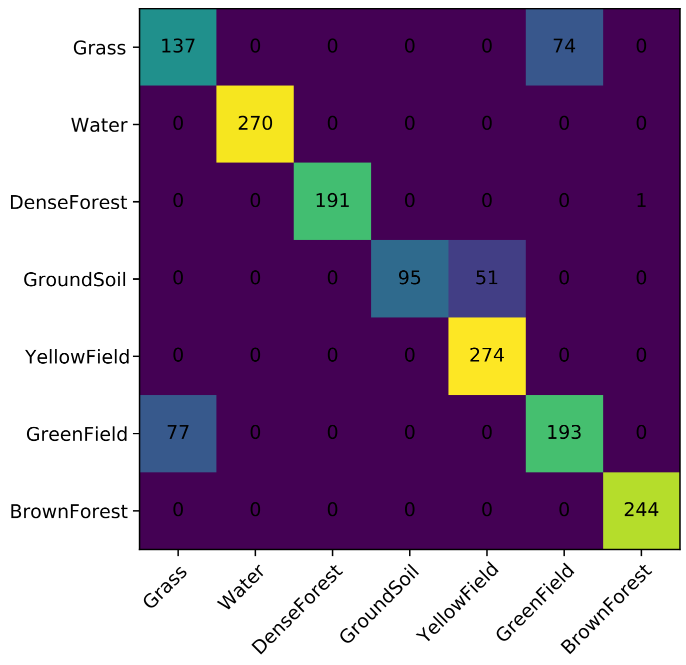


Figure D.9: Confusion matrix. Model: KNN, type: annual, season: spring, area: highlands.

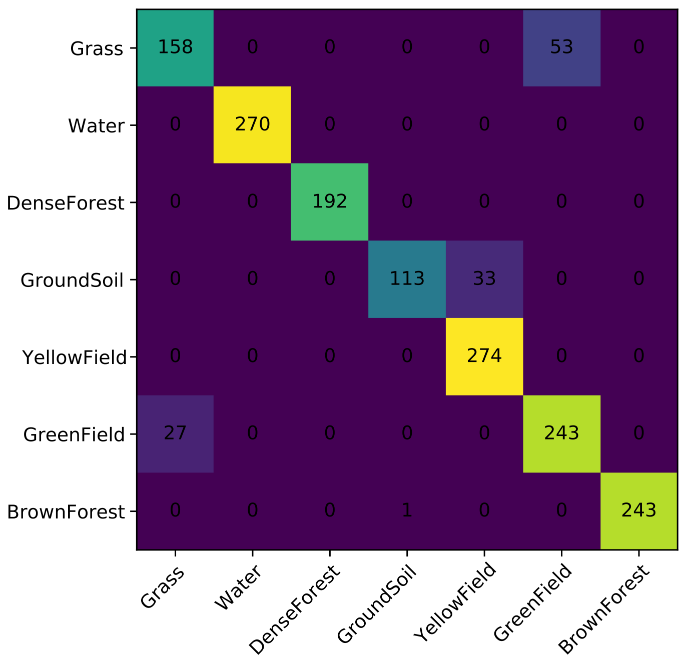


Figure D.10: Confusion matrix. Model: CNN, type: seasonal, season: spring, area: highlands.

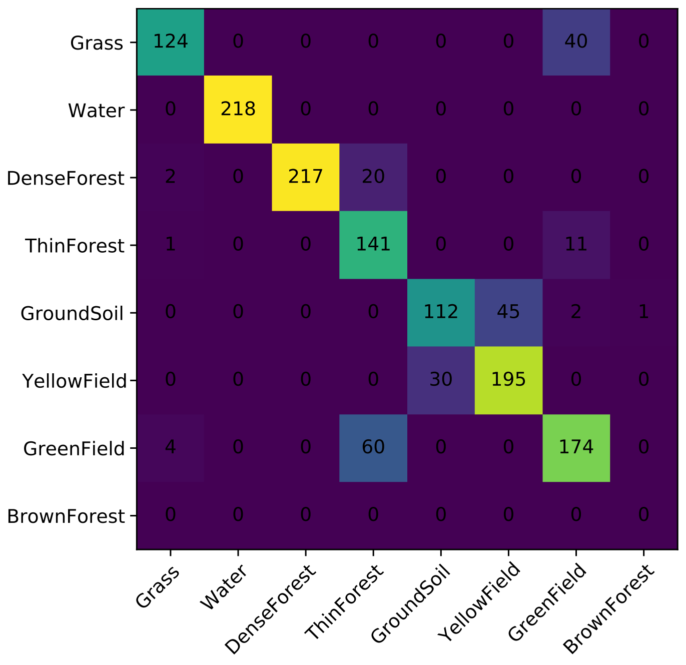


Figure D.11: Confusion matrix. Model: KNN, type: annual, season: summer, area: highlands.

GroundSoil in a matter of days. With values of kappa statistic below 0.65, these classifiers are not well suited for this task.

D.2.3 The winter season

Similar to the mountains area, the highlands area was also completely covered in snow during the winter season. For this reason, the classification was omitted as well.

D.3 Experiment results - flatlands

The area of the South of Moravia in the Czech Republic is unique in several ways. Firstly there is no coniferous forest in this region. Only the deciduous forest, that changes its color depending on the season of the year. Secondly, this area consists almost entirely of agricultural fields, making it a very important aspect of the accuracy assessment. And lastly, a new type of **PurpleField** class is introduced to differentiate among various field types.

D.3.1 The spring season

In general, all four classifiers reported very similar classification results (the distribution inside the confusion matrices was almost identical for all four classifiers). The seasonal classifiers achieved very slightly higher overall accuracy. The best-performing classifier was the seasonal CNN with an overall accuracy of 83 %. Its confusion matrix is visualized in figure D.14.

The common problems in this season was the differentiation among classes **GroundSoil**, **YellowField**, and **YellowGrass** but also between the classes **Water** and **GreenField**. The misclassification of water did not occur in previous areas of interest. In flatlands, however, the training images of water bodies often consist of ponds with still water where the algae thrive and create a green cover over the water body. The algae-covered ponds may get confused for green vegetation growing on the land.

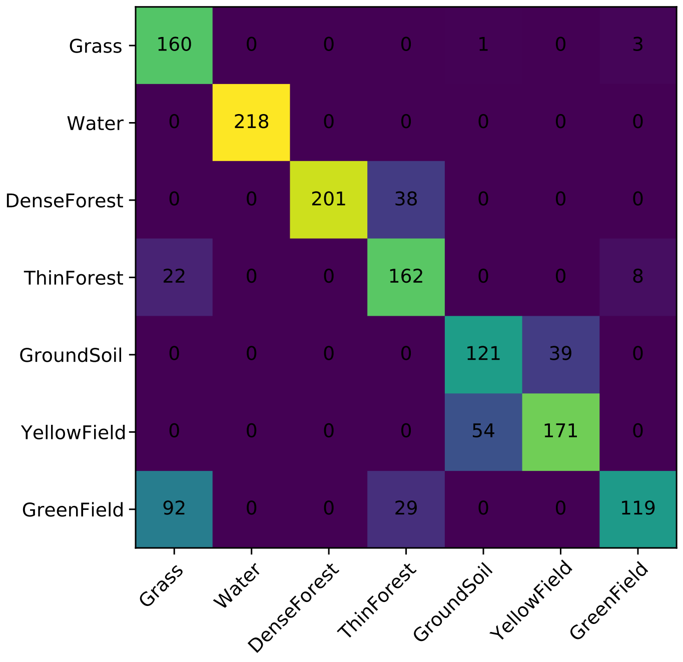


Figure D.12: Confusion matrix. Model: CNN, type: seasonal, season: summer, area: highlands.

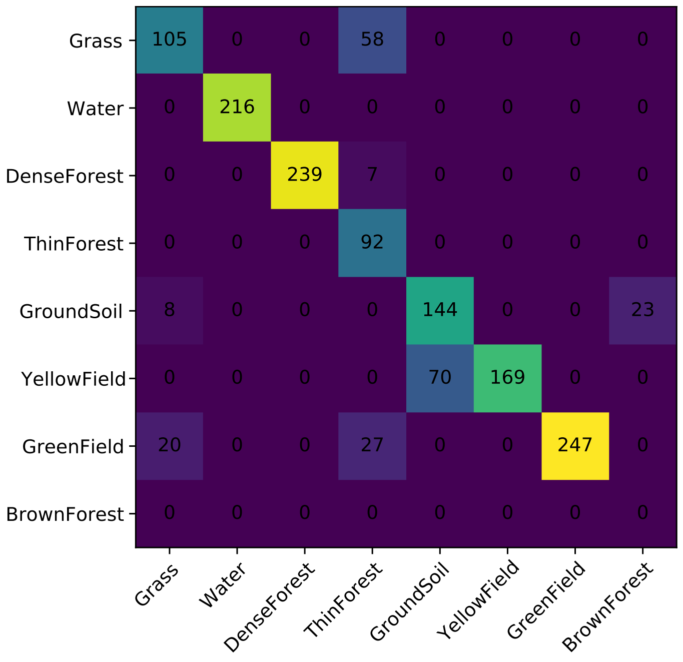


Figure D.13: Confusion matrix. Model: CNN, type: annual, season: fall, area: highlands.

D.3.2 The summer season

Very good overall accuracy was achieved by the CNN classifiers in the summer season. Especially the seasonal CNN classifier scored 84 % in overall accuracy. The annual kNN classifier achieved significantly lower accuracy in this season.

Confusing **Grass** with **GreenField** and **YellowField** with **GroundSoil** were again the biggest problems for the classifiers. The misclassification of water was, however, much less significant in summer than it was in spring. The confusion matrix of the best-performing seasonal CNN classifier is shown in figure D.15.

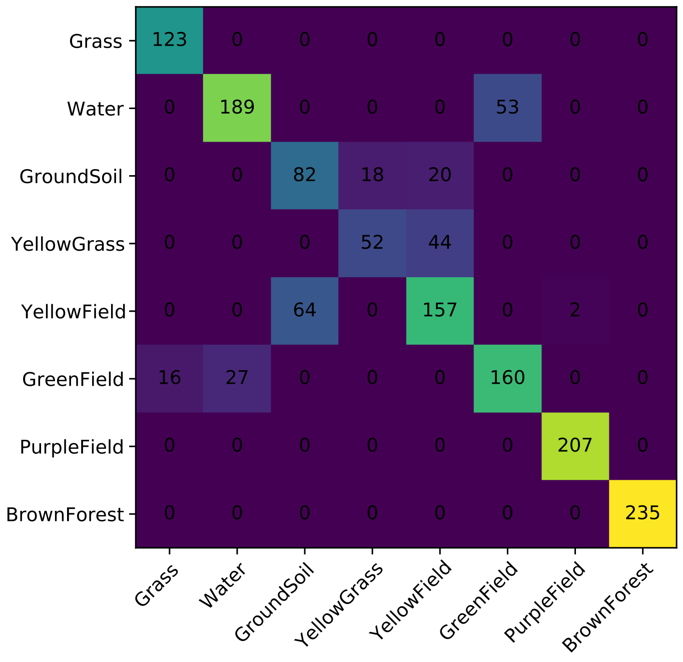
D.3.3 The fall season

The accuracy assessment for the fall season returned great performance for all four classifiers. The seasonal classifiers even exceeded the overall accuracy of 90 %. Good separation of classes can be seen in figure D.16 that visualizes the confusion matrix of the seasonal CNN classifier. However, a slight confusion between classes **BrownForest** and **PurpleField** shown by the matrix is problematic because these classes have very different meanings in the context of search-and-rescue operations.

D.3.4 The winter season

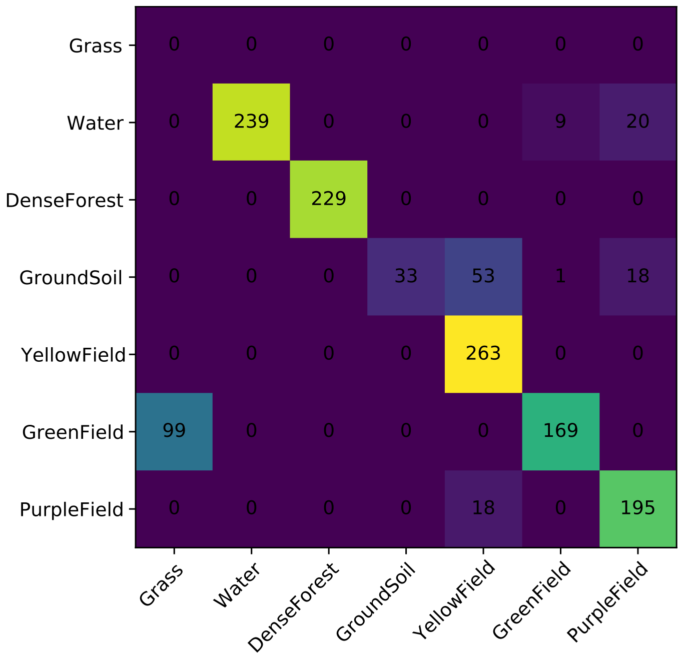
Unlike in previous areas of interest, the flatlands area was not completely covered in snow most of the time during the winter season. Classifying images from this season is therefore possible. On the other hand, obtaining enough training data for this specific season can be an issue as most of the images are cloudy. Because of this, only one satellite image from the year 2020 was available for the purpose of training the seasonal winter classifiers. For this reason, the results may suffer from the lack of training images.

The confusion matrix of the annual CNN classifier (shown in figure D.17) contains very similar values to the confusion matrix of the same classifier evaluated for the spring season. The overall accuracy, however, dropped to only 73 %. Seasonal classifiers improved it by a small amount (75 % in the case of the seasonal CNN).



	Grass	Water	GroundSoil	YellowGrass	YellowField	GreenField	PurpleField	BrownForest
Grass	123	0	0	0	0	0	0	0
Water	0	189	0	0	0	53	0	0
GroundSoil	0	0	82	18	20	0	0	0
YellowGrass	0	0	0	52	44	0	0	0
YellowField	0	0	64	0	157	0	2	0
GreenField	16	27	0	0	0	160	0	0
PurpleField	0	0	0	0	0	0	207	0
BrownForest	0	0	0	0	0	0	0	235

Figure D.14: Confusion matrix. Model: CNN, type: seasonal, season: spring, area: flatlands.



	Grass	Water	DenseForest	GroundSoil	YellowField	GreenField	PurpleField
Grass	0	0	0	0	0	0	0
Water	0	239	0	0	0	9	20
DenseForest	0	0	229	0	0	0	0
GroundSoil	0	0	0	33	53	1	18
YellowField	0	0	0	0	263	0	0
GreenField	99	0	0	0	0	169	0
PurpleField	0	0	0	0	18	0	195

Figure D.15: Confusion matrix. Model: CNN, type: seasonal, season: summer, area: flatlands.

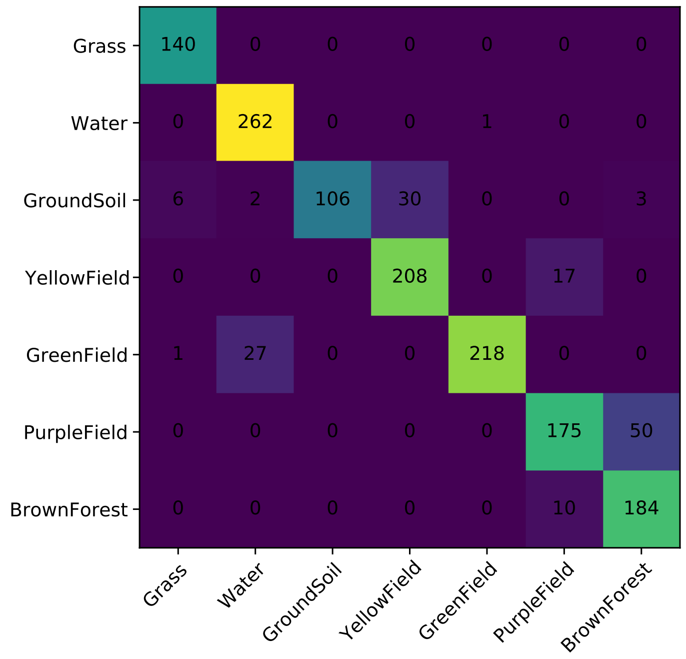


Figure D.16: Confusion matrix. Model: CNN, type: seasonal, season: fall, area: flatlands.

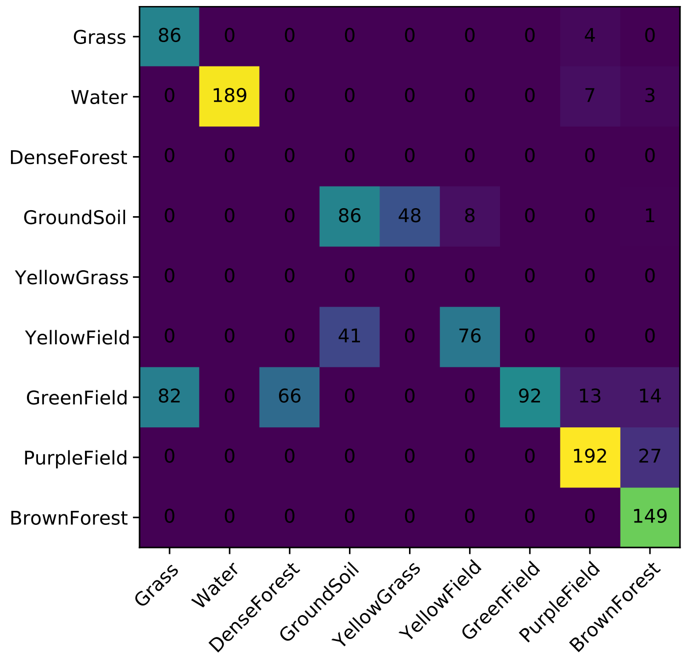


Figure D.17: Confusion matrix. Model: CNN, type: annual, season: winter, area: flatlands.

Appendix E

Experiment results

In this appendix, statistics from all classifications are stated. See chapter 6 for further information regarding used metrics.

Classes	Grass	Water	DenseForest	ThinForest	GroundSoil	YellowGrass	BrownForest
*k*NN annual							
Support	141	166	166	89	86	131	62
Support (%)	16.77	19.74	19.74	10.58	10.23	15.58	7.37
Precision	0.81	1.00	0.98	0.87	0.52	0.84	0.00
Recall	0.95	1.00	1.00	0.67	0.72	0.94	0.00
F_1-measure	0.87	1.00	0.99	0.76	0.60	0.89	0.00
						Overall accuracy	0.85
						Kappa statistic	0.82
*k*NN seasonal							
Support	141	166	166	89	86	131	62
Support (%)	16.77	19.74	19.74	10.58	10.23	15.58	7.37
Precision	0.91	1.00	1.00	0.93	0.48	0.68	1.00
Recall	0.98	1.00	1.00	0.92	0.27	0.99	0.47
F_1-measure	0.94	1.00	1.00	0.93	0.34	0.80	0.64
						Overall accuracy	0.87
						Kappa statistic	0.85
CNN annual							
Support	141	166	166	89	86	131	62
Support (%)	16.77	19.74	19.74	10.58	10.23	15.58	7.37
Precision	1.00	1.00	0.99	1.00	0.54	0.64	1.00
Recall	0.70	1.00	1.00	0.93	0.52	1.00	0.53
F_1-measure	0.82	1.00	1.00	0.97	0.53	0.78	0.69
						Overall accuracy	0.86
						Kappa statistic	0.83
CNN seasonal							
Support	141	166	166	89	86	131	62
Support (%)	16.77	19.74	19.74	10.58	10.23	15.58	7.37
Precision	0.97	1.00	0.99	0.90	0.00	0.52	1.00
Recall	1.00	1.00	1.00	0.94	0.00	1.00	0.29
F_1-measure	0.99	1.00	1.00	0.92	0.00	0.68	0.45
						Overall accuracy	0.84
						Kappa statistic	0.81

Table E.1: Results: mountains (spring).

Classes	Grass	Water	DenseForest	ThinForest	GroundSoil
*k*NN annual					
Support	150	166	166	81	113
Support (%)	22.19	24.56	24.56	11.98	16.72
Precision	0.76	1.00	0.98	0.96	0.75
Recall	0.86	1.00	1.00	0.57	0.56
F_1-measure	0.81	1.00	0.99	0.71	0.64
				Overall accuracy	0.84
				Kappa statistic	0.80
*k*NN seasonal					
Support	150	166	166	81	113
Support (%)	22.19	24.56	24.56	11.98	16.72
Precision	0.89	1.00	0.99	0.94	0.78
Recall	0.80	1.00	0.99	0.80	0.90
F_1-measure	0.84	1.00	0.99	0.87	0.84
				Overall accuracy	0.91
				Kappa statistic	0.89
CNN annual					
Support	150	166	166	81	113
Support (%)	22.19	24.56	24.56	11.98	16.72
Precision	0.90	1.00	0.98	0.89	0.71
Recall	0.81	1.00	1.00	0.93	0.59
F_1-measure	0.86	1.00	0.99	0.91	0.64
				Overall accuracy	0.88
				Kappa statistic	0.85
CNN seasonal					
Support	150	166	166	81	113
Support (%)	22.19	24.56	24.56	11.98	16.72
Precision	0.92	0.99	1.00	1.00	0.84
Recall	0.88	1.00	0.99	0.85	0.87
F_1-measure	0.90	0.99	0.99	0.92	0.86
				Overall accuracy	0.93
				Kappa statistic	0.91

Table E.2: Results: mountains (summer).

Classes	Grass	Water	DenseForest	ThinForest	GroundSoil	YellowGrass
*k*NN annual						
Support	150	166	166	89	91	32
Support (%)	21.61	23.92	23.92	12.82	13.11	4.61
Precision	0.72	1.00	1.00	0.78	0.51	0.00
Recall	0.92	1.00	0.95	0.44	0.48	0.00
F_1-measure	0.81	1.00	0.98	0.56	0.49	0.00
					Overall accuracy	0.79
					Kappa statistic	0.73
*k*NN seasonal						
Support	150	166	166	89	91	32
Support (%)	21.61	23.92	23.92	12.82	13.11	4.61
Precision	0.79	1.00	0.99	0.83	0.90	0.00
Recall	1.00	0.99	0.90	0.85	0.70	0.00
F_1-measure	0.88	1.00	0.94	0.84	0.79	0.00
					Overall accuracy	0.87
					Kappa statistic	0.84
CNN annual						
Support	150	166	166	89	91	32
Support (%)	21.61	23.92	23.92	12.82	13.11	4.61
Precision	0.62	1.00	0.98	0.76	0.48	0.00
Recall	0.85	1.00	1.00	0.21	0.54	0.00
F_1-measure	0.72	1.00	0.99	0.33	0.51	0.00
					Overall accuracy	0.76
					Kappa statistic	0.70
CNN seasonal						
Support	150	166	166	89	91	32
Support (%)	21.61	23.92	23.92	12.82	13.11	4.61
Precision	0.82	1.00	0.99	0.93	1.00	0.48
Recall	0.89	1.00	0.98	0.88	0.47	0.97
F_1-measure	0.86	1.00	0.98	0.90	0.64	0.64
					Overall accuracy	0.88
					Kappa statistic	0.86

Table E.3: Results: mountains (fall).

Classes	Grass	Water	DenseForest	GroundSoil	YellowField	GreenField	BrownForest
kNN annual							
Support	211	270	192	146	274	270	244
Support (%)	13.13	16.80	11.95	9.09	17.05	16.80	15.18
Precision	0.64	1.00	1.00	1.00	0.84	0.72	1.00
Recall	0.65	1.00	0.99	0.65	1.00	0.71	1.00
F_1-measure	0.64	1.00	1.00	0.79	0.91	0.72	1.00
						Overall accuracy	0.87
						Kappa statistic	0.85
kNN seasonal							
Support	211	270	192	146	274	270	244
Support (%)	13.13	16.80	11.95	9.09	17.05	16.80	15.18
Precision	0.78	1.00	1.00	1.00	0.85	0.82	1.00
Recall	0.77	1.00	0.99	0.66	1.00	0.83	1.00
F_1-measure	0.77	1.00	1.00	0.80	0.92	0.83	1.00
						Overall accuracy	0.91
						Kappa statistic	0.89
CNN annual							
Support	211	270	192	146	274	270	244
Support (%)	13.13	16.80	11.95	9.09	17.05	16.80	15.18
Precision	0.42	1.00	1.00	0.99	0.90	0.51	1.00
Recall	0.74	1.00	1.00	0.78	1.00	0.21	1.00
F_1-measure	0.54	1.00	1.00	0.87	0.94	0.30	1.00
						Overall accuracy	0.81
						Kappa statistic	0.78
CNN seasonal							
Support	211	270	192	146	274	270	244
Support (%)	13.13	16.80	11.95	9.09	17.05	16.80	15.18
Precision	0.85	1.00	1.00	0.99	0.89	0.82	1.00
Recall	0.75	1.00	1.00	0.77	1.00	0.90	1.00
F_1-measure	0.80	1.00	1.00	0.87	0.94	0.86	1.00
						Overall accuracy	0.93
						Kappa statistic	0.92

Table E.4: Results: highlands (spring).

Classes	Grass	Water	DenseForest	ThinForest	GroundSoil	YellowField	GreenField
*k*NN annual							
Support	164	218	239	192	160	225	240
Support (%)	11.40	15.16	16.62	13.35	11.13	15.65	16.69
Precision	0.95	1.00	1.00	0.66	0.79	0.81	0.75
Recall	0.76	1.00	0.91	0.93	0.70	0.87	0.68
F_1-measure	0.84	1.00	0.95	0.77	0.74	0.84	0.71
						Overall accuracy	0.84
						Kappa statistic	0.81
*k*NN seasonal							
Support	164	218	239	192	160	225	240
Support (%)	11.40	15.16	16.62	13.35	11.13	15.65	16.69
Precision	0.66	1.00	0.99	0.54	0.64	0.85	0.79
Recall	0.86	0.94	0.64	0.87	0.83	0.68	0.55
F_1-measure	0.74	0.97	0.78	0.67	0.72	0.75	0.65
						Overall accuracy	0.75
						Kappa statistic	0.71
CNN annual							
Support	164	218	239	192	160	225	240
Support (%)	11.40	15.16	16.62	13.35	11.13	15.65	16.69
Precision	0.58	1.00	1.00	0.86	0.81	0.84	0.97
Recall	0.99	1.00	0.87	0.99	0.73	0.88	0.52
F_1-measure	0.73	1.00	0.93	0.92	0.77	0.86	0.67
						Overall accuracy	0.85
						Kappa statistic	0.82
CNN seasonal							
Support	164	218	239	192	160	225	240
Support (%)	11.40	15.16	16.62	13.35	11.13	15.65	16.69
Precision	0.58	1.00	1.00	0.71	0.69	0.81	0.92
Recall	0.98	1.00	0.84	0.84	0.76	0.76	0.50
F_1-measure	0.73	1.00	0.91	0.77	0.72	0.79	0.64
						Overall accuracy	0.80
						Kappa statistic	0.77

Table E.5: Results: highlands (summer).

Classes	Grass	Water	DenseForest	ThinForest	GroundSoil	YellowField	GreenField
*k*NN annual							
Support	163	216	196	92	171	239	236
Support (%)	12.41	16.45	14.93	7.01	13.02	18.20	17.97
Precision	0.62	1.00	0.99	0.65	0.75	1.00	0.86
Recall	0.58	1.00	1.00	0.92	0.95	0.78	0.77
F_1-measure	0.60	1.00	1.00	0.76	0.84	0.88	0.81
						Overall accuracy	0.85
						Kappa statistic	0.83
*k*NN seasonal							
Support	163	216	246	92	175	239	294
Support (%)	11.44	15.16	17.26	6.46	12.28	16.77	20.63
Precision	0.27	1.00	0.99	0.00	0.00	0.53	0.56
Recall	0.45	1.00	1.00	0.00	0.00	0.56	0.83
F_1-measure	0.34	1.00	0.99	0.00	0.00	0.55	0.67
						Overall accuracy	0.64
						Kappa statistic	0.57
CNN annual							
Support	163	216	246	92	175	239	294
Support (%)	11.44	15.16	17.26	6.46	12.28	16.77	20.63
Precision	0.79	1.00	1.00	0.50	0.67	1.00	1.00
Recall	0.64	1.00	0.97	1.00	0.82	0.71	0.84
F_1-measure	0.71	1.00	0.99	0.67	0.74	0.83	0.91
						Overall accuracy	0.85
						Kappa statistic	0.83
CNN seasonal							
Support	163	216	246	92	175	239	294
Support (%)	11.44	15.16	17.26	6.46	12.28	16.77	20.63
Precision	0.41	1.00	1.00	0.00	0.00	0.49	0.77
Recall	0.88	1.00	1.00	0.00	0.00	0.57	0.81
F_1-measure	0.56	1.00	1.00	0.00	0.00	0.53	0.79
						Overall accuracy	0.69
						Kappa statistic	0.63

Table E.6: Results: highlands (fall).

Classes	Grass	Water	GroundSoil	YellowGrass	YellowField	GreenField	PurpleField	BrownForest
kNN annual								
Support	123	242	120	96	223	203	207	235
Support (%)	8.49	16.70	8.28	6.63	15.39	14.01	14.29	16.22
Precision	0.71	0.90	0.38	0.84	0.57	0.96	0.99	0.96
Recall	0.93	0.98	0.50	0.54	0.57	0.48	0.96	1.00
F_1-measure	0.81	0.94	0.43	0.66	0.57	0.64	0.97	0.98
							Overall accuracy	0.78
							Kappa statistic	0.74
kNN seasonal								
Support	123	242	120	96	223	203	207	235
Support (%)	8.49	16.70	8.28	6.63	15.39	14.01	14.29	16.22
Precision	0.89	0.89	0.42	0.74	0.66	0.85	0.99	0.98
Recall	0.85	0.89	0.62	0.53	0.61	0.74	0.97	1.00
F_1-measure	0.87	0.89	0.50	0.62	0.63	0.79	0.98	0.99
							Overall accuracy	0.81
							Kappa statistic	0.78
CNN annual								
Support	123	242	120	96	223	203	207	235
Support (%)	8.49	16.70	8.28	6.63	15.39	14.01	14.29	16.22
Precision	0.90	0.88	0.56	0.61	0.72	0.73	0.98	1.00
Recall	0.98	0.80	0.56	0.54	0.75	0.65	1.00	1.00
F_1-measure	0.94	0.84	0.56	0.57	0.74	0.68	0.99	1.00
							Overall accuracy	0.81
							Kappa statistic	0.78
CNN seasonal								
Support	123	242	120	96	223	203	207	235
Support (%)	8.49	16.70	8.28	6.63	15.39	14.01	14.29	16.22
Precision	0.88	0.88	0.56	0.74	0.71	0.75	0.99	1.00
Recall	1.00	0.78	0.68	0.54	0.70	0.79	1.00	1.00
F_1-measure	0.94	0.83	0.62	0.63	0.71	0.77	1.00	1.00
							Overall accuracy	0.83
							Kappa statistic	0.81

Table E.7: Results: flatlands (spring).

Classes	Water	DenseForest	GroundSoil	YellowField	GreenField	PurpleField
*k*NN annual						
Support	268	229	105	263	268	213
Support (%)	19.91	17.01	7.80	19.54	19.91	15.82
Precision	1.00	0.91	0.73	0.90	0.93	0.74
Recall	1.00	0.93	0.08	0.91	0.84	0.43
F_1-measure	1.00	0.92	0.14	0.90	0.88	0.54
					Overall accuracy	0.77
					Kappa statistic	0.73
*k*NN seasonal						
Support	268	229	105	263	268	213
Support (%)	19.91	17.01	7.80	19.54	19.91	15.82
Precision	1.00	0.90	0.64	0.77	0.78	0.94
Recall	0.97	0.86	0.07	1.00	0.72	0.95
F_1-measure	0.98	0.88	0.12	0.87	0.75	0.95
					Overall accuracy	0.83
					Kappa statistic	0.80
CNN annual						
Support	268	229	105	263	268	213
Support (%)	19.91	17.01	7.80	19.54	19.91	15.82
Precision	1.00	1.00	1.00	0.76	0.98	0.92
Recall	0.99	1.00	0.09	0.98	0.91	0.56
F_1-measure	0.99	1.00	0.16	0.86	0.94	0.70
					Overall accuracy	0.83
					Kappa statistic	0.80
CNN seasonal						
Support	268	229	105	263	268	213
Support (%)	19.91	17.01	7.80	19.54	19.91	15.82
Precision	1.00	1.00	1.00	0.79	0.94	0.84
Recall	0.89	1.00	0.31	1.00	0.63	0.92
F_1-measure	0.94	1.00	0.48	0.88	0.76	0.87
					Overall accuracy	0.84
					Kappa statistic	0.81

Table E.8: Results: flatlands (summer).

Classes	Grass	Water	GroundSoil	YellowField	GreenField	PurpleField	BrownForest
*k*NN annual							
Support	140	263	147	225	246	225	194
Support (%)	9.72	18.26	10.21	15.62	17.08	15.62	13.47
Precision	0.87	0.89	1.00	0.79	1.00	0.76	0.83
Recall	0.79	1.00	0.60	0.99	0.77	0.84	0.91
F_1-measure	0.83	0.94	0.75	0.88	0.87	0.79	0.86
						Overall accuracy	0.86
						Kappa statistic	0.83
*k*NN seasonal							
Support	140	263	147	225	246	225	194
Support (%)	9.72	18.26	10.21	15.62	17.08	15.62	13.47
Precision	0.95	0.90	0.99	0.92	0.96	0.87	0.82
Recall	1.00	0.97	0.82	0.95	0.88	0.83	0.92
F_1-measure	0.98	0.93	0.90	0.93	0.92	0.85	0.87
						Overall accuracy	0.91
						Kappa statistic	0.89
CNN annual							
Support	140	263	147	225	246	225	194
Support (%)	9.72	18.26	10.21	15.62	17.08	15.62	13.47
Precision	0.95	0.91	1.00	0.73	1.00	0.78	0.78
Recall	0.68	1.00	0.39	1.00	0.89	0.76	1.00
F_1-measure	0.79	0.95	0.56	0.85	0.94	0.77	0.88
						Overall accuracy	0.85
						Kappa statistic	0.82
CNN seasonal							
Support	140	263	147	225	246	225	194
Support (%)	9.72	18.26	10.21	15.62	17.08	15.62	13.47
Precision	0.95	0.90	1.00	0.87	1.00	0.87	0.78
Recall	1.00	1.00	0.72	0.92	0.89	0.78	0.95
F_1-measure	0.98	0.95	0.84	0.90	0.94	0.82	0.85
						Overall accuracy	0.90
						Kappa statistic	0.88

Table E.9: Results: flatlands (fall).

Classes	Grass	Water	GroundSoil	YellowField	GreenField	PurpleField	BrownForest
*k*NN annual							
Support	90	199	143	117	267	219	149
Support (%)	7.60	16.81	12.08	9.88	22.55	18.50	12.58
Precision	0.29	1.00	0.58	0.52	0.85	0.83	0.81
Recall	0.92	0.95	0.55	0.53	0.13	0.88	0.93
F_1-measure	0.44	0.97	0.57	0.52	0.22	0.86	0.87
						Overall accuracy	0.66
						Kappa statistic	0.61
*k*NN seasonal							
Support	90	199	143	117	267	219	149
Support (%)	7.60	16.81	12.08	9.88	22.55	18.50	12.58
Precision	0.36	1.00	1.00	0.46	0.80	0.93	0.96
Recall	0.58	0.93	0.03	1.00	0.66	1.00	0.91
F_1-measure	0.45	0.97	0.07	0.63	0.72	0.96	0.93
						Overall accuracy	0.75
						Kappa statistic	0.71
CNN annual							
Support	90	199	143	117	267	219	149
Support (%)	7.60	16.81	12.08	9.88	22.55	18.50	12.58
Precision	0.51	1.00	0.68	0.90	1.00	0.89	0.77
Recall	0.96	0.95	0.60	0.65	0.34	0.88	1.00
F_1-measure	0.67	0.97	0.64	0.76	0.51	0.88	0.87
						Overall accuracy	0.73
						Kappa statistic	0.70
CNN seasonal							
Support	90	199	143	117	267	219	149
Support (%)	7.60	16.81	12.08	9.88	22.55	18.50	12.58
Precision	0.36	0.87	0.00	0.43	0.87	0.88	0.98
Recall	0.34	0.91	0.00	1.00	0.80	1.00	0.82
F_1-measure	0.35	0.89	0.00	0.60	0.83	0.94	0.89
						Overall accuracy	0.75
						Kappa statistic	0.70

Table E.10: Results: flatlands (winter).

Appendix F

Digital attachment

The digital attachment contains the source code of all implemented methods. The complete user guide with more information can be found in the file *README.md*. The code is split into three main modules.

The first module can be used to bulk-download Sentinel-2 imagery from a publicly available datasource.

The second module is the classifier. All mathematical methods discussed in this thesis are supported by this module. The abstract classifier implements all parts of the supervised classification task:

1. training the classifier with labeled training data,
2. classifying selected image using the trained classifier.

The last module is the labeling application that was used for manual labeling of training and validation images for this thesis.

Appendix G

Bibliography

[1] EuropeanSpaceAgency, *Sentinel-2.* ESA Communications, 2012.

[2] H. Chaloupková, I. Svobodová, J. Růžička, V. Makeš, K. Novák, M. Hradec, M. Kouba, V. Bittner, P. Smejkal, and J. Hepnar, “Metodika pro plánování a řízení pátrání po pohřešovaných osobách v terénu za využití it technologií,” *Certifikovaná metodika area – CMA*, 2020.

[3] A. Skidmore, *Environmental Modelling with GIS and Remote Sensing.* Taylor and Francis, 2002.

[4] H. J. Kramer, *Observation of the Earth and its Environment: Survey of Missions and Sensors.* Springer-Verlag Berlin Heidelberg, 2002.

[5] G. D’Souza, A. S. Belward, and J.-P. Malingreau, *Advances in the use of NOAA AVHRR data for land applications*, vol. 5. Springer Science & Business Media, 2013.

[6] J. P. Snyder, *Map projections–A working manual*, vol. 1395. US Government Printing Office, 1987.

[7] K.-T. Chang, *Geographic Information Systems.* Mc Graw Hill India, 2007.

[8] P. Thanh Noi and M. Kappas, “Comparison of random forest, k-nearest neighbor, and support vector machine classifiers for land cover classification using sentinel-2 imagery,” *Sensors*, vol. 18, no. 1, 2018.

[9] Hester, Barry, Cakir, Halil, Nelson, Stacy, Khorram, and Siamak, “Per-pixel classification of high spatial resolution satellite imagery for urban land-cover mapping,” vol. 74, 2008.

[10] M. Längkvist, A. Kiselev, M. Alirezaie, and A. Loutfi, "Classification and segmentation of satellite orthoimagery using convolutional neural networks," *Remote Sensing*, vol. 8, no. 4, 2016.

[11] B. Fröhlich, E. Bach, I. Walde, S. Hese, C. Schmullius, and J. Denzler, "Land cover classification of satellite images using contextual information," *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, pp. 1–6, 2013.

[12] G. Büttner, B. Kosztra, T. Soukup, A. Sousa, and T. Langanke, "Clc2018 technical guidelines," *European Environment Agency: Copenhagen, Denmark*, vol. 25, 2017.

[13] Copernicus-Programme, "Land use cases." [online], 2021. [cit. 2021-03-05] Available at: `https://land.copernicus.eu/user-corner/land-use-cases`.

[14] C. Champagne, A. Berg, J. Belanger, H. McNairn, and R. de Jeu, "Evaluation of soil moisture derived from passive microwave remote sensing over agricultural sites in canada using ground-based soil moisture monitoring networks," *International Journal of Remote Sensing*, vol. 31, no. 14, pp. 7–11, 2010.

[15] X. Wu, V. Kumar, J. R. Quinlan, J. Ghosh, Q. Yang, H. Motoda, G. J. McLachlan, A. Ng, B. Liu, P. S. Yu, Z.-H. Zhou, M. Steinbach, D. J. Hand, and D. Steinberg, "Top 10 algorithms in data mining," *Knowledge and Information Systems*, vol. 14, 2008.

[16] A. Vinnikov and S. Shalev-Shwartz, "K-means recovers ica filters when independent components are sparse," in *Proceedings of the 31st International Conference on Machine Learning*, vol. 32, pp. 712–720, PMLR, Bejing, China, 2014.

[17] G. Mclachlan, S. K. A. Ng, and D. Peel, "On clustering by mixture models," 2003.

[18] T. Cover and P. Hart, "Nearest neighbor pattern classification," *IEEE Trans. Inf. Theory*, vol. 13, pp. 21–27, 1967.

[19] J. Friedman, J. Bentley, and R. Finkel, "An algorithm for finding best matches in logarithmic expected time," *ACM Trans. Math. Softw.*, vol. 3, pp. 209–226, 09 1977.

[20] Y. Chen, L. Zhou, Y. Tang, J. P. Singh, N. Bouguila, C. Wang, H. Wang, and J. Du, "Fast neighbor search by using revised k-d tree," *Information Sciences*, vol. 472, pp. 145–162, 2019.

[21] A. Khan, A. Sohail, U. Zahoora, and A. S. Qureshi, "A survey of the recent architectures of deep convolutional neural networks," *CoRR*, vol. abs/1901.06032, 2019.

[22] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, pp. 436–444, 05 2015.

[23] M. Ranzato, F. J. Huang, Y. Boureau, and Y. LeCun, "Unsupervised learning of invariant feature hierarchies with applications to object recognition," pp. 1–8, 2007.

[24] C. S. Sergey Ioffe, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *CoRR*, vol. abs/1502.03167, 2015.

[25] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, no. 56, pp. 1929–1958, 2014.

[26] S. Hochreiter, "The vanishing gradient problem during learning recurrent neural nets and problem solutions," *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, vol. 6, pp. 107–116, 1998.

[27] European Space Agency, "Copernicus open access hub." [online], 2014. [cit. 2021-02-01] Available at: `https://scihub.copernicus.eu/`.

[28] SentinelSat contributors, "sentinelsat." [online], 2020.

[29] MapBox, "rasterio." [online], 2016. [cit. 2021-03-05] Available at: `https://github.com/mapbox/rasterio`.

[30] M. Main-Knorn, B. Pflug, J. Louis, V. Debaecker, U. Müller-Wilm, and F. Gascon, "Sen2cor for sentinel-2," p. 3, 10 2017.

[31] K. Tarrio, X. Tang, J. G. Masek, M. Claverie, J. Ju, S. Qiu, Z. Zhu, and C. E. Woodcock, "Comparison of cloud detection algorithms for sentinel-2 imagery," *Science of Remote Sensing*, vol. 2, p. 100010, 2020.

[32] Z. Zhu and C. E. Woodcock, "Object-based cloud and cloud shadow detection in landsat imagery," *Remote Sensing of Environment*, vol. 118, pp. 83–94, 2012.

[33] Sentinel Hub, "custom-scripts." [online], 2021. [cit. 2020-12-02] Available at: `https://github.com/sentinel-hub/custom-scripts`.

[34] J. Braaten, W. Cohen, and Z. Yang, "Automated cloud and cloud shadow identification in landsat mss imagery for temperate ecosystems," *Remote Sensing of Environment*, vol. 169, pp. 128–138, 11 2015.

[35] A. Hollstein, K. Segl, L. Guanter, M. Brell, and M. Enesco, "Ready-to-use methods for the detection of clouds, cirrus, snow, shadow, water and clear sky pixels in sentinel-2 msi images," *Remote Sensing*, vol. 8, no. 8, 2016.

[36] Scikit-Learn-contributors, “scikit-learn.” [online], 2021. [cit. 2020-12-02] Available at: `https://github.com/scikit-learn/scikit-learn`.

[37] QGIS contributors, “Qgis.” [online], 2021. [cit. 2021-02-05] Available at: `https://github.com/qgis/QGIS`.

[38] European Space Agency, “Snap.” [online], 2016. [cit. 2021-01-15] Available at: `https://github.com/senbox-org/snap-desktop`.

[39] Toblerity contributors, “Fiona.” [online], 2020. [cit. 2020-11-28] Available at: `https://github.com/Toblerity/Fiona`.

[40] Google, “Tensorflow.” [online], 2021. [cit. 2020-12-18] Available at: `https://github.com/tensorflow/tensorflow`.

[41] M. Bureš, M. Renda, M. Doležel, *et al.*, *Efektivní testování softwaru: klíčové otázky pro efektivitu testovacího procesu.* Grada Publishing as, 2016.

[42] OpenStreetMap contributors, “Planet dump retrieved from https://planet.osm.org .” [online], 2017. [cit. 2021-03-20] Available at: `https://www.openstreetmap.org`.

[43] C. D. Manning, P. Raghavan, and H. Schütze, *Introduction to Information Retrieval.* Cambridge University Press, 2008.