NTNU
Norwegian University of
Science and Technology

Image Processing of 2-D Snow Images for Cross-Country Skiing

Master Thesis TFY4900 – Applied Physics

Fredrik Rosenberg

June 2020

Supervisors: Astrid S. de Wijn and Antonius Van Helvoort

Norwegian University of Science and Technology
Faculty of Engineering / Faculty of Natural Sciences
Department of Mechanical and Industrial Engineering / Department of Physics

Abstract

Snow is a porous and complex material influencing a great variety of scientific, technological and societal fields, making snow analysis an important but demanding task. Although three-dimensional snow analysis techniques provide comprehensive and accurate characterization of the snow's microstructure, these methods are often time consuming and require less accessible equipment. To obtain a better impression of the temporal and spatial variations in the snow's structure, more facile probing methods are needed. For cross-country skiing, the snow's structure has direct impact on friction and thus the final performance of the skis. Hence, analysing the current and locally varying snow conditions will give valuable information on how to best prepare the skis in order to achieve better overall results. The presented methods in this study are based on image analysis of two-dimensional images. These are acquired on an instrument called GelSight, which is a handheld and portable imaging system enabling fast and effective imaging of the snow's surface. The first image processing method involves the use of digitized contours to approximate the area and perimeter of snow grains. This allows determining the Optical Equivalent Diameter (OED) and dendricity, two metrics characterizing the size and shape of the grains. A second processing method involves using the machine learning algorithms Support Vector Machine (SVM) and Convolutional Neural Network (CNN) to classify snow images into categories as old and new snow. The contouring method is shown to successfully separate different snow conditions through the calculated OED and dendricity, and the SVM and CNN algorithms achieve an accuracy on the classification of 95.8% and 97.9%, respectively. As the structural parameters and classification relate directly to the observed property of friction, the presented snow analysis is indeed providing valuable information regarding snow conditions relevant to the performance of cross-country skis.

Sammendrag

Snø er et porøst og komplekst materiale med innflytelse på både vitenskapelige, teknologiske og sosiale arenaer. Dette gjør analyse av snø til en viktig, men krevende oppgave. Tredimensjonal snø analyse gir nøyaktig karakterisering av snøens mikrostruktur, men disse metodene er ofte tidkrevende i tillegg til å avhenge av mindre tilgjengelig utstyr. For å få et bedre bilde av hvordan snøens struktur endrer seg i rom og tid, er det derfor behov for mindre omfattende metoder for karakterisering av snø. For langrennsski vil snøens struktur ha direkte innflytelse på friksjonen og dermed ytelsen til skiene. Analyse av de gjeldende og lokalt varierende snøforholdene vil dermed gi verdifull informasjon om hvordan skiene bør prepareres for å oppnå bedre resultater. I denne rapporten er de presenterte metodene basert på analyse av todimensjonale bilder av snø. Bildene er tatt med et instrument ved navn GelSight, et håndholdt og bærbart instrument som raskt og effektivt fanger bilder av snøens overflate. Den første metoden innebærer å bruke digitale konturer for å approksimere areal og omkrets av snøkornene. Fra dette kan man bestemme den Optisk Ekvivalente Diameteren (OED) og 'dendricity', to parametere som karakteriserer størrelsen og formen på snøkornene. Den andre metoden bruker maskinlæringsalgoritmene 'Support Vector Machine' (SVM) og 'Convolutional Neural Network' (CNN) for å klassifisere snø som ny eller gammel. Kontur-metoden lykkes i å skille forskjellige snøforhold gjennom OED og 'dendricity' verdiene, og SVM og CNN algoritmene oppnår en treffsikkerhet på henholdsvis 95.8% og 97.9% på klassifiseringen. De strukturelle parameterne og klassifiseringen relaterer direkte til friksjon, og den presenterte snø analysen gir derfor verdifull informasjon om snøforholdene relevant for ytelsen til langrennsski.

Preface

This project originates from a cooperation between the Department of Mechanical and Industrial Engineering at NTNU and Olympiatoppen with the underlying goal of analysing snow to improve performance of cross-country skis. The master thesis is written and published for the Department of Physics and marks the end of a five year long study program in physics and mathematics. It is a mandatory report for master degree study programs in engineering at NTNU, and for the Department of Physics weighted to a total of 30 ETCS credits. The study was carried out over the spring semester 2020, and all work presented in this thesis is done by the author unless stated otherwise.

Acknowledgements

First and foremost, I would like to thank my supervisor Astrid S. de Wijn for letting me have this project and for the help and guidance throughout the semester. I would also like to thank Bassma Al-Jubouri at the department of Mechanical and Industrial Engineering for advice regarding the classification and characterization of images. My thanks to my formal supervisor, Antonius Van Helvoort, for taking me on and giving tips to the report structure despite his full schedule.

I would also like to extend my gratitude to Felix Breitschädel and Olympiatoppen for providing me with the GelSight images and letting me try the instrument for myself, in addition to giving me a thorough introduction to the preparation of cross country skis.

Trondheim, 05.06.2020

Fredrik Rosenberg

Table of Contents

Chapter 1

Introduction

In the northern parts of the world, snow has interested and affected people for generations. Whether this interest originates from building the perfect snowman, skiing as fast as possible, sending radio-frequent signals through glaciers, researching avalanches or calculating the absorbed solar radiation on the earth's surface, the properties of snow have an important say. Considering its variety of applications, investigation and characterization of snow is of great interest.

Snow is a sintered material consisting of mono-crystalline ice grains bonded together (Colbeck, 1998). The ice crystals form in the atmosphere before falling to the ground where they immediately start sintering together to the foam-like structure known as snow. Here, the snow is exposed to environmental conditions like temperature, wind and humidity which continuously induce changes in the snow's microstructure. These rapid changes and dependence on multiple parameters makes snow a complex material, and there is a need for effective and practical methods to analyse the snow's structure.

Someone with great interest in snow analysis is Olympiatoppen, a Norwegian organization in charge of development of elite sports in Norway. Among the sports they work with is cross-country skiing, where having the better skis on a race day can differentiate between a place on the podium and walking home empty handed. The skis' performance are dependent on ski preparation that can be adapted to the given snow conditions. This rises the question of which skis to prepare on which snow conditions. In order to answer this question, Olympiatoppen have acquired a portable and handheld imaging system called GelSight (GelSight, 2019), which provides a simple and effective way of imaging snow grains. These images can be further analysed to extract properties of snow grains and distinguish between different types of snow.

The motivation of this study is to characterize and classify images of snow to deduce the snow's structure in terms relevant to the performance of skis. By creating robust image processing methods to effectively distinguish between different snow conditions, one can compare previous performances of ski preparation and snow combinations to find the optimal skis for snow with given structure. In this report two novel image processing methods are developed and evaluated. The first involves using a contour approach to calculate the Optical Equivalent Diameter (OED) and dendricity of snow grains, two metrics shown to impact the friction of skis (Böttcher and Scherge, 2017). The second method involves using the machine learning algorithms Support Vector Machine (SVM) and Convolutional Neural Network (CNN) to classify the snow's structure into categories, here old and new snow.

This report consists of seven chapters. Chapter 2 gives an overview of different parameters and methods used in both snow characterization and image analysis. In addition, the underlying theory behind the gliding resistance of skis and applied machine learning algorithms is presented. Chapter 3 describes the two applied methods for snow image analysis, which results are presented in chapter 4 and discussed in chapter 5. The conclusions can be found in chapter 6, before suggestions for future work are presented in chapter 7.

Chapter 2

Theory

2.1 Characterization of Snow

Given the variety of scientific fields and practical problems where snow plays an important role, its microstructure that determines the snow's macroscopic properties has undergone extensive research. However, finding the best metrics in establishing a common ground for snow characterization is not considered an easy task. Snow is a porous material consisting of continuous ice structures and pore spaces, which together form the microstructure. As the snow is laying on the ground, it is exposed to many factors affecting this microstructure. The metamorphism of snow, i.e. the physical change of snow grains due to pressure and temperature changes, is the main factor. As the temperature of snow is close to its melting point, liquid water can occupy the pore spaces, essentially leading to liquid, solid and gaseous phases of water coexisting on the ground (Fierz et al. (2009)). This continually ongoing metamorphism, along with the wind and intermittent nature of precipitation, makes each stratigraphic layer of a snowpack different from each other. Snow is therefore one of the most complex materials on the Earth's surface (Pomeroy and Brun (2001)) and the need for dynamic snow characterization is crucial. Fierz et al. (2009) addressed the problems with inconsistency in snow characterization and deduced a common terminology from which the following parameters are taken from.

2.1.1 Parameters in Snow Characterization

Snow Grain Size

Grain size is an intuitive way of characterizing snow. It is simply the average size of the grains for a given snow layer, ranging form very fine ($<0.2\,\mathrm{mm}$) to very coarse ($2.0\,\mathrm{mm}$-$5.0\,\mathrm{mm}$) and also extreme ($>5.0\,\mathrm{mm}$). However, grain size does not always give the most relevant description of the electromagnetic and mechanical properties of snow. Also, traditional measurements of snow grain size using magnified lens and grid cards are biased as the results differ for different observers (Painter et al. (2007)). Some field techniques therefore use the term Optical Equivalent Diameter (OED), which specifies the diameter of ice spheres that best approximate the scattering and absorption of the actual grains, i.e. spheres exhibiting the same surface to volume ratio (section 3.2.2). Thus, OED in many ways is a objective quantification of grain size, and the gliding resistance at the snow-to-ski interface is shown to depend on the OED of snow grains (Bartlett et al. (2008)).

Snow Grain Shape

Grain shape is a frequently used term in snow characterization, and like grain size it will have impact on the microstructure of the snow, affecting its physical properties (including snow-ski friction). There are many classes of grain shape, as labelled by Fierz et al. (2009), and one of the most used shape descriptors is sphericity. Sphericity, in snow classification, is a measure of how close a snow grain is to being a sphere. This descriptor is used to distinguish between rounded grains which have curvature and faceted grains which have reduced curvature.

Another shape descriptor frequently used in snow analysis is dendricity. It is defined as the ratio of the square of the perimeter of a grain to its area (section 3.2.2). Thus, like sphericity, it says something about the complexity of the grain outline, as seen in **Fig. 2.1**. Freshly fallen snow tends to have more complex structures compared to old snow, therefore dendricity can also be used to differ between new and old snow (Bartlett et al., 2008; Lesaffre et al., 1998).

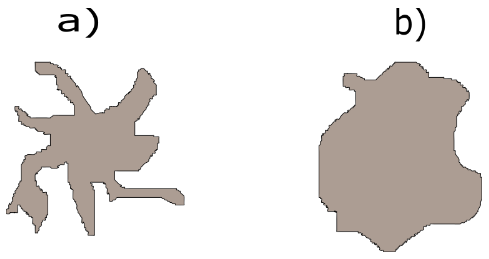


Figure 2.1: Parametric shapes: a) is a complex shape with higher dendricity, while b) is a rounder shape with lower dendricity.

Snow Density

Even though the density of a porous material is a bulk property, its importance on the microstructure can be severe. The mechanical, optical and thermal properties of snow are heavily dependent on the configuration of ice structure and pore spaces, which is embedded in the snows density. The density is also related to other properties of snow, e.g. temperature, as warmer temperatures will lead to more liquid water replacing air in the pore spaces, increasing the weight of the snow. Thus, density is an important property of snow and a frequently used term in snow characterization.

A normal way to determine the snow's density is weighing snow of a known volume. As there is a large discrepancy between the density of moist and dry snow, the total snow density and dry snow density are often measured separately. In addition to weighing snow of a fixed volume, there are other methods to determine the density of snow, including X-ray Computed Tomography (micro-CT) (Lundy et al., 2002) and taking advantage of the dielectric properties (Denoth, 1989).

Snow Hardness

The hardness of snow is a term describing the resistance to an object penetrating the snow. The hardness is a criteria to establish snow stratigraphy, i.e. the different properties between different layers in the snowpack, and is therefore an important parameter in snow characterization. Measurements of snow hardness especially play an important role in avalanche research, where the strength of the snowpack is directly linked to the formation of avalanches (Tyagi et al., 2013). Also, when it comes to ski friction, the deformation of snow when pressure is applied is a relevant factor that depends on the snow's hardness (section 2.2).

Measuring hardness by hand has been, and still is, a common method for scientific measurements of snow hardness. By applying a force and using either the fist, four fingers or one finger to penetrate the snow layer, the observer measures the hardness. Knife blades and pencils can also be used. However, as this method relies on the force used by the observer, the method is subjective. The Swiss ramsonde (Bader et al., 1939), where a ramsonde is driven into the snow by a mechanical hammer blowing on a probe, has also been frequently used. Here, the hardness resolution is limited by the probe itself and the weight of the hammer. Nowadays, the SnowMicroPen is shown to give quasi-continuous hardness readings with better resolution and accuracy (Pielmeier and Schneebeli, 2002), making this the favourable choice for hardness measurements.

2.1.2 Image Analysis and Methods in Snow Characterization

3-D

The most present methods in snow characterization today involves using imaging technologies like micro-CT and X-ray. These are techniques that give an accurate 3-D analysis of the microstructure. However, these techniques require instruments that are not always easy to access and may reside in a laboratory far away. As mentioned, snow is fragile and changes fast due to its metamorphism, making it necessary to preserve the microstructure of the snow sample during transfer. This can be achieved by casting the snow samples with solidifying liquid, as described by Heggli et al. (2011).

After obtaining the images, there are several approaches to further analysis. Krol and Löwe (2016) use the two point correlation function and chord length distribution in order to describe the microstructure of snow from micro-CT images. As the OED alone is not sufficient in describing the complex physical properties of snow, they investigate additional size and shape metrics. The chord length distribution is defined as the lengths of intersection of random rays through the sample with the ice phase (**Fig. 2.2**), while the two-point correlation function is a statistical distribution containing relevant sizes in the microstructure. In the end, the authors were able to define grain shape via size dispersity. Although size dispersity is not exhaustive in characterizing the influence of grain shape, this intersection provides a quantitative starting point for further analysis.

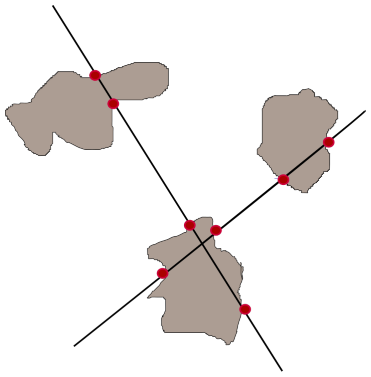


Figure 2.2: The mean chord length is defined as the mean length of lines between the red points in the snow grains.

2-D

Despite the detailed and accurate analysis techniques from 3-D instruments, there is a need for simpler and less elaborate methods in snow characterization. 2-D analysis provides such methods, where one can use digitized images of snow grains to investigate the 2-D surface properties of snow. This is especially useful when the main interest lies in the snow's surface, as for the glide of skis, where the interactions at the snow-ski interface play the major part (section 2.2).

For 2-D analysis to compete with the well established 3-D analysis methods, the used instruments must be easier accessible and provide faster and more effective imaging of the snow. As the metamorphism give rise to rapid changes in the snow's structure, portable instruments providing on-site imaging of snow conditions are favorable. An instrument possessing all these qualities is the GelSight (section 3.1), which is a portable and handheld imaging system using an elastomeric sensor and silicone gel to conform the surface topography of the snow. This results in fast and effective imaging of the snow's surface.

After obtaining the images, they are often preprocessed by noise-removal and segmentation, i.e. turned into black-and-white by using a threshold value on the grayscale images. It is worth mentioning that this is the same process as for 3-D methods like micro-CT. Here one takes a stack of 2-D images, applies noise removal and segmentation, before using multiple images from different angles to reconstruct a 3-D image. For pure 2-D analysis, on the other hand, the image characterization process continues in two dimensions. After the noise removal and segmentation, one can find the contours (outlines) of the snow grains, and therefore a way of interpreting the microstructure. Bartlett et al. (2008) uses both a parabola and a cubic spline approach in curvature measurements of the grain outlines. The goal is to estimate the dendricity and sphericity of the grains using these two methods.

The cubic spline method is a form of piecewise polynomial interpolation. After the preprocessing of the images, the cubic spline method can be applied as a smoothing interpolator of the discrete points in the grain outlines, producing smooth grain shapes. Using the arc length s, a discrete grain outline can be parameterized in Cartesian coordinates as $C(s_n) = (x(s_n), y(s_n))$, where n is the vector index. Further, the smoothing spline function $f(s)$ then minimizes the expression

$$p \sum_{n-1}^{N} |C(s_n) - f(s_n)| + (1-p)\lambda \int |\frac{d^2 f(s)}{ds^2}|^2 \, ds. \tag{2.1}$$

Here p is the so-called smoothing parameter, N is the number of points in the outline and λ is a weight factor to eliminate scale-induced changes between different grains. Equation (2.1) consists of two terms, where the first term is a standard measure of squared distances and the second term is a roughness measurement of the spline. Note that $p = 0$ will make the first term vanish, only restricting the spline to minimize the roughness measure. If $p = 1$ the second term will vanish, and the spline is now only constrained to minimize the squared error term. Bartlett et al. (2008) uses the No-gradient method to find the optimal smoothing coefficient p for each grain.

The parabola approach is the standard for curvature calculation of snow grain outlines. The method consists of evaluating the curvature of every pixel in the grain outline. First, one selects a chosen amount of neighbouring pixels. Thereafter, one rotates this section of pixels into the most horizontal direction before fitting the parabola. The easiest way to ensure the most horizontal orientation of the outline section is found, is simply applying angular increments and measuring the distance in x-direction. For parametric equations $x = x(t)$ and $y = y(t)$, the curvature of a plane curve is defined as $\kappa = \frac{d\phi}{ds}$, where s is the aforementioned arc length and ϕ is the angle between the tangent curve and the x-axis. From this it can be derived that

$$\kappa(t) = \frac{\dot{x}\ddot{y} - \ddot{x}\dot{y}}{(\dot{x}^2 + \dot{y}^2)^{3/2}}. \tag{2.2}$$

The parabola fitting of the rotated outline section can then be found by evaluating the curvature at the origin for $x(t) = t$ and $y(t) = at^2 + bt$. From equation (2.2) one gets

$$\kappa \equiv \kappa(0) = \frac{2a}{(1+b^2)^{3/2}}. \tag{2.3}$$

Using the parabola approach and the spline method, Bartlett et al. estimate the sphericity and dendricity of the contours. Although both methods showed considerable error in curvature measurement depending on the complexity of the grain shapes, they still enabled a broad characterization of different snow grains.

2.2 Ski-Snow Interaction

When considering the interaction on the snow-to-ski interface, there are two main questions to consider. The first is the deformation mode of snow, i.e. how the microstructure of snow changes when a force is applied. Theile et al. (2009) use a special linear friction tester and measure four types of deformations; brittle fracture of grain bonds, plastic deformation of ice at the ski-snow contact spots and elastic and delayed elastic deformation of the snow. The term elastic indicates that the snow's structure goes back to its previous state after the applied force is removed, while for plastic deformation the changes in the structure are permanent. They found that delayed elastic deformation of snow is the most dominant.

The second question is finding the real contact area between the snow and the skis. This is the area where friction takes place and is therefore of great importance (Kragelsky and Demkin, 1960). By neglecting shear force from snow grains rubbing against the skis and frictional melting at the top of the snow, one can estimate the real contact area to be the ratio between the normal force and the strength of ice (penetration hardness). Theile et al. measure the real contact area to be $0.4\,\%$ and the average contact spot size to be $110\,\mu\text{m}$ using this approach.

The gliding resistance, i.e. friction of the ski-snow interface, is a complex process difficult to understand. Now that the adhesion theory is mostly related to very low velocities (Maeno and Arakawa, 2004), the melt-water lubrication theory (e.g. Lüthi et al. (2018)) describes the current idea of ski-snow friction in the terms mentioned above. As the ski glides over the snow surface, there will be a deformation of the snow surface and ski base both, before a (real) contact area between the two is formed. Assuming initial dry snow, the dry friction will lead to dissipation of heat into the snow's surface. The dissipation rate of heat into the snow's surface, Q, is given as

$$Q = \mu v N, \tag{2.4}$$

where μ is the friction coefficient, v is the velocity of the skis and N is the normal force on the snow. The friction coefficient μ is directly linked to the surface roughness of snow, as higher surface roughness will increase the friction coefficient (e.g. Kietzig et al. (2009)). Further, the dissipated heat will again lead to melt-water being formed, lubricating the contact area and lowering the friction. However, if too much water is melted, capillary bridges between the snow and the ski base can be formed, increasing the friction. Thus, the friction is evidently minimal when the contact area between ski and snow is minimal and there is just enough water to have lubricated friction. This highly depends on environmental factors and ski preparation both, although Budde and Himes (2017) found the environmental conditions (snow conditions) to affect friction more. Especially around the melting point of snow, Böttcher and Scherge (2017) found that parameters like grain size and grain shape have larger impact on the friction, encouraging investigation of these parameters.

2.3 Machine Learning for Image Classification

2.3.1 Introduction to Machine Learning

Artificial intelligence (AI) is an umbrella term containing systems where a machine is capable of imitating intelligent human behaviour. Tasks normally requiring human intelligence, like speech recognition, visual perception, decision making and language translation are all examples of AI. Machine learning is an approach of AI concerned with machines' ability to learn from data on their own. Mitchell (1997) defines machine learning to include any computer program that improves its performance at some tasks through experience. More precisely, "a computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E." In other words, machine learning describes systems with the ability to automatically learn and improve from experience without being explicitly programmed, but relying on patterns and inference instead.

Supervised and Unsupervised Learning

There are two main categories of machine learning algorithms, namely supervised and unsupervised learners. Which category to choose algorithms from is dependent on the problem at hand and the data available.

In supervised learning, the data consists of both input X and output y. The goal is to use an algorithm to learn the mapping function $f(X) = y$. The dataset the algorithm learns from is called training data. By iteratively predicting the output values of the training data and comparing with the true output, one can calculate the error and adjust the internal parameters of the learning algorithm to minimize it. Hence, the training makes the learning algorithm approximate the mapping function. In order to investigate the validity of the approximated mapping function, test data is used to measure its accuracy. The only difference between training data and test data is that where training data is used to fit and optimize the learning algorithm, the test data is unseen to the model. Therefore, the test data can be used to make predictions and calculate the accuracy of the model by comparing the predicted labels to the actual labels (**Fig. 2.3**). High accuracy for the test data indicates that the model is able to generalize well and can be used in prediction of new, unlabeled (unknown output y) data. Supervised learners are often used in classification (for which the output variable is a category) and regression problems (for which the output variable is a real value). An example of a supervised learning problem can be to distinguish between handwritten digits (Kaensar, 2013).

For unsupervised learners, the available data consists of input X without corresponding output variables. Since the training data only consists of inputs, the goal here is to learn a function to find hidden structures, like groups and clusters in the data (**Fig. 2.4**). This comes in handy when it is difficult to label the data or the variations in data are large, as the unsupervised learning models can be trained to make its own separation by extracting the underlying patterns. Applications of unsupervised learning is target marketing and spam email filtering among others. (Diale et al., 2019).

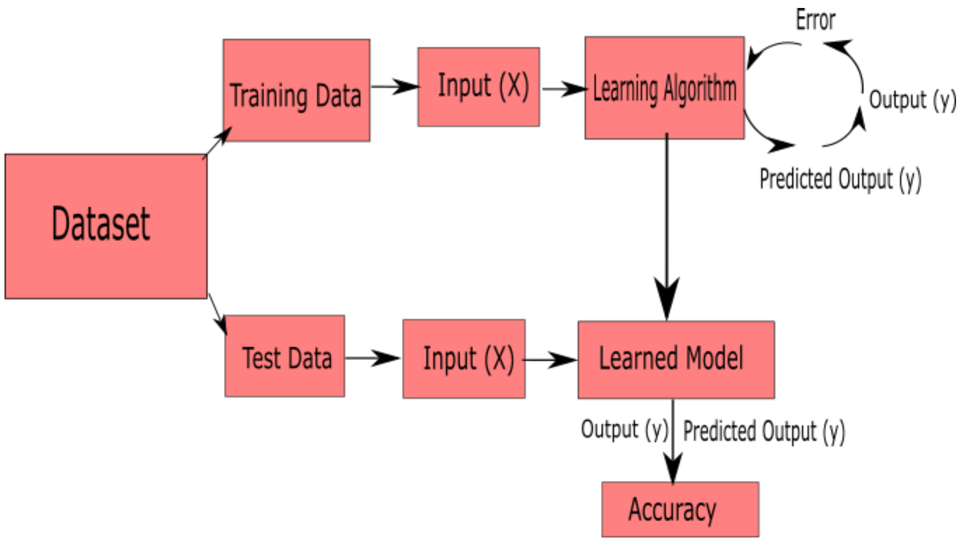


Figure 2.3: A schematic flow of training and testing in supervised learning. First, both the training and test set are acquired from the same pool of data. The algorithm is then trained by comparing the predicted output to the actual output. Calculating the error function and adjusting the internal parameters of the learning algorithm to minimize this function will lead to the learned model. Further, the test data is used to validate the learned model by predicting the output from the input data. Comparing the predicted output to the actual output, the accuracy of the learning algorithm is assessed.

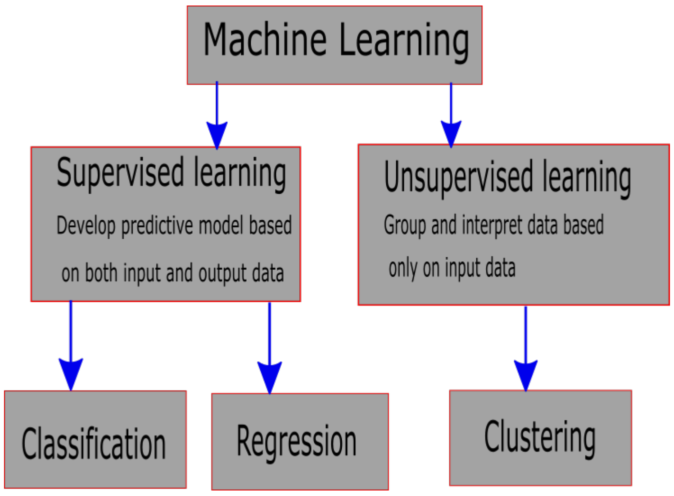

Machine Learning
Supervised learning
Develop predictive model based on both input and output data
Unsupervised learning
Group and interpret data based only on input data
Classification
Regression
Clustering

Figure 2.4: A schematic flow showing the differences between supervised and unsupervised learning. Supervised learning develop predictive models based on input and output data, while unsupervised learning find similarities in the input of the data.

2.3.2 Image Classification

Image recognition is the ability of AI to detect, classify and recognize objects in images. This technology is for example used when unblocking a smartphone by facial recognition. The AI system detects the face, classifies it as a human face and recognizes it as the owner of the smartphone. The classification part of this process is a machine learning problem, more specifically a supervised learning problem.

Image classification uses labelled images as training data, in order to classify new, unlabelled images. The inputs, often called features, consist of arrays of pixel values which together form a representation of the image. The process of choosing the optimal features plays a key part in image classification as it largely determines the performance of the classification system. However, this is by no means a trivial task, as Dollar et al. (2007) point out when describing the characteristics of a good feature. The authors state that a good feature should be (1) informative, (2) invariant to noise or a given set of transformations, and (3) fast to compute. Using all the pixels in an image will lead to informative features, but given that images might contain thousands of pixels in several color channels, this will lead to unreasonable running times. Hence, finding the optimal set of features is challenging.

For image classification, a good learner should be insensitive to irrelevant variations of the input such as position, illumination and orientation, while being sensitive to other minute variations. Images of dogs and cats with the same color, background and orientation could give rise to similar pixel values, while two images of dogs with different colors, backgrounds and orientations would look different at pixel level. A classifier operating on raw pixel values cannot distinguish between the former two images while correctly classifying the latter two, making feature extraction techniques necessary. Common techniques include using mean, variance, color features, gradients etc. However, as these feature extraction techniques need to be pre-selected, it can be hard to choose which techniques to use, especially for complex images. Some learning algorithms therefore have the ability to extract features themselves, as described in section 2.3.4.

2.3.3 Support Vector Machine

Support Vector Machine (SVM) is a supervised learning algorithm. Although it can be used in classification and regression both, it is most frequently used in classification as introduced by Cortes and Vapnik (1995). Especially in classification of images, SVM is commonly used, achieving good accuracy on benchmark image datasets compared to other state of the art learning algorithms (Kaensar, 2013; Le et al., 2012). While initially designed for binary classification, SVM can be extended to be used for multi-class problems as well, as described by Weston and Watkins (1999). Here, however, the focus will be on the binary version of the algorithm.

In binary classification, SVM is designed to find the best classification function distinguishing between two different classes. The algorithm uses geometrical optimization to find the hyperplane in a high-dimensional feature space that maximizes the margin, i.e. the distance between the hyperplane and the closest datapoints of the two classes. These datapoints are called the support vectors, and the support vectors fully determine the placement of the hyperplane. The hyperplane is used as a decision boundary where all datapoints, of which the feature vector lies on one side of the hyperplane, are separated into their respective classes, as demonstrated in **Fig. 2.5**.

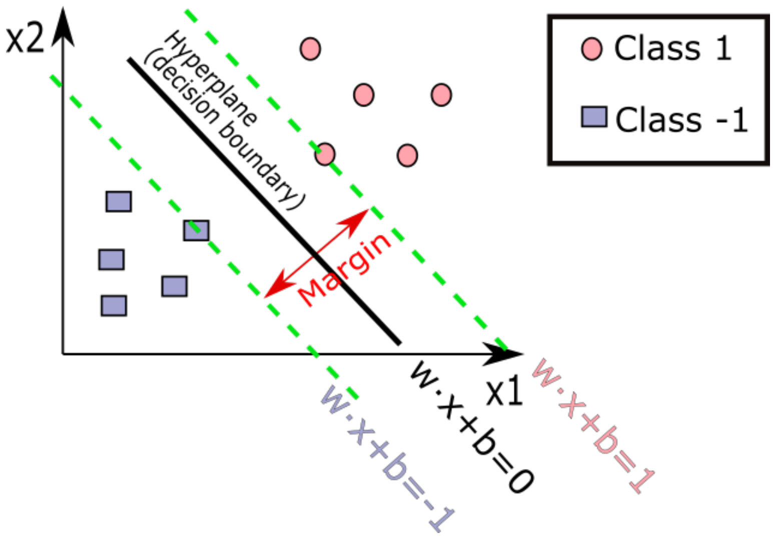


Figure 2.5: A graphical representation of the SVM algortihm for a two-dimensional feature vector $\vec{x} = (x_1, x_2)$ and linear separable data. The algorithm maximizes the margin to find the optimal hyperplane for classification of the two classes.

Linear SVM

The training data consists of n points on the form $(\vec{x_1}, y_1), ..., (\vec{x_n}, y_n)$, where $\vec{x_i}$ represents the feature vector, and y_i the label taking values $y_i = 1$ and $y_1 = -1$ for the respective classes. Assuming a two-dimensional feature space and linearly separable data, there exists a line $x_2 = ax_1 + b$ separating the classes. This gives $ax_1 + b - x_2 = 0$. Rewriting to vector form the equation for a hyperplane can be written as

$$\vec{w} \cdot \vec{x} + b = 0, \tag{2.5}$$

where $\vec{w} = (a, -1)$ is a normal vector to the hyperplane and $\vec{x} = (x_1, x_2)$ is the already stated feature vector of the training data. Although derived here from two-dimensional vectors, equation (2.5) holds for any number of dimensions. The goal in SVM is as mentioned to maximize the margin, and this can be achieved by constructing two parallel hyperplanes that separate the two classes of data, namely

$$\vec{w} \cdot \vec{x_i} + b = -1 \quad \text{for } y_i = -1 \text{ and} \tag{2.6}$$

$$\vec{w} \cdot \vec{x_i} + b = 1 \quad \text{for } y_i = 1. \tag{2.7}$$

The hyperplane represented by equation (2.6) classifies all points on or below this boundary to the class represented by $y_i = -1$, while equation (2.7) classifies all points on or above this boundary to the class represented by $y_i = 1$. This can be summarized as

$$\vec{w} \cdot \vec{x_i} + b \leq -1 \quad \text{for } y_i = -1 \text{ and} \tag{2.8}$$
$$\vec{w} \cdot \vec{x_i} + b \geq 1 \quad \text{for } y_i = 1. \tag{2.9}$$

Combining equation (2.8) and equation (2.9) gives the constraint

$$y_i(\vec{w} \cdot \vec{x_i} + b) \geq 1, \quad \text{for } y_i = \{-1, 1\} \text{ and } 1 \leq i \leq n. \tag{2.10}$$

The distance between the two hyperplanes can be shown to be $\frac{2}{||\vec{w}||}$ (Adankon and Cheriet, 2009). The maximum margin hyperplane (equation (2.5)) is the hyperplane that lies halfway between these two hyperplanes, with a margin of $M = \frac{1}{||\vec{w}||}$. Maximizing this margin equals minimizing $||\vec{w}||$. Therefore, the optimization problem becomes minimizing $||\vec{w}||$ subjected to the constraints of equation (2.10).

Non-Linear SVM

The complexity of real world problems give rise to datasets that are characterized by their non-linearity. For SVM to classify non-linear data, one applies a technique called the kernel trick (Schölkopf, 2000). The idea is mapping the non-linear dataset into a high-dimensional feature space where one can find a hyperplane that can separate the datapoints (**Fig. 2.6**).

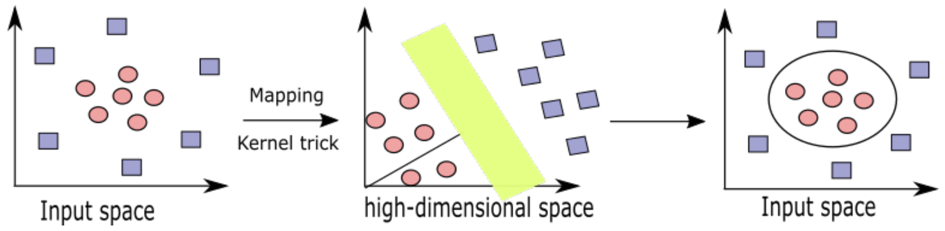


Figure 2.6: Schematic representation of the kernel trick. Data cannot be separated by linear SVM in input space. By applying the kernel trick, the data is mapped to a high-dimensional feature space where a hyperplane can be used as a decision border for the now linearly separable data. Thus, the SVM algorithm is able to classify the non-linear data.

Let Φ be the mapping of $\mathbb{R}^n \rightarrow \mathbb{R}^m$, that maps the vectors in $\mathbb{R}^n$ to some feature space $\mathbb{R}^m$. A kernel is defined as a function $K(\vec{x_i}, \vec{x_j}) = \Phi(\vec{x_i}) \cdot \Phi(\vec{x_j})$, corresponding to the the inner product of $\vec{x_i}$ and $\vec{x_j}$ in the feature space. The inner products of feature vectors are calculated when solving the optimization problem presented in the previous subsection (Asraf et al., 2012). In many cases, the computation of inner products of feature vectors in high-dimensional spaces can be complex and computational demanding tasks (Huang et al., 2018). Hence, the kernels are useful as they give a way to compute inner products in some feature space without knowing the characteristics of the space or explicitly calculating the mapping Φ.

2.3.4 Convolutional Neural Networks

Convolutional Neural Networks are a class of deep learning, which is a subset of machine learning that imitates the working of the human brain in order to be used in decision making. As previously mentioned in section 2.3.2, conventional machine learning requires feature extracting as a prerequisite (Lecun et al., 1998), which can be challenging in cases where prerequisite knowledge of the data is limited. Deep learning overcomes this problem by not requiring pre-selected features, but extracting significant features from raw input automatically. This is achieved by using a collection of processing layers that learns features through multiple levels of abstraction, as described by Lecun et al. (2015). Given their advantages, deep learning has emerged as the leading architecture in many problems, including speech recognition, image classification and language translation among others (Indolia et al., 2018).

To mimic the behaviour of the human brain, deep learning uses artificial neural networks consisting of neurons connected in a web (**Fig. 2.7**). A neuron contains a set of inputs, weights and an activation function. The inputs can either be raw features coming directly from the input values or be the output of neurons from the previous layer. As seen from **Fig. 2.7**, a neuron have many connections (arrows) to other units in the network. The neuron processes this information as a weighted sum of all inputs. That is, for a set of inputs $X = [x_1, x_2, x_3, ..., x_n]$ and weights $W = [w_1, w_2, w_3, ..., w_n]$ one calculates the sum $E(x_1, x_2, x_3, ..., x_n) = w_1x_1 + w_2x_2 + w_3x_3 + ... + w_nx_n$. This value is then fed to the activation function of the neuron. The purpose of the activation function is to add non-linearity to the neural network. As the weighted sum of the inputs only consists of linear operations, the activation function is needed to perform non-linear mappings from input to output. An example of an activation function is the sigmoid function, $\sigma(z) = \frac{1}{1+e^{-z}}$. As the sigmoid function is restrained to take values on the interval $< 0, 1 >$, it can be viewed as a probability measure and is especially useful for models where the output takes probabilistic form, e.g. in classification of images. The output from the activation function is used as input for neurons in the next layer, where the same operations are repeated. This process, called forward propagation, continues until the output layer of the neural network is reached. Here, for classification, the output values are normalized into a probability distribution. The input is then mapped to the output with the highest probability value.

Training of the neural network is achieved by predicting the output, comparing it to the desired (labeled) output and adjusting the weights to minimize the error. To adjust the weight vector, the learning algorithm computes a gradient vector that, for each weight, calculates what the increase or decrease in the error would be if the weight were slightly increased. The direction of the negative gradient vector indicates the steepest descent, where the error gets smaller and eventually converges to a minimum. The weight vector is then adjusted in the opposite direction to the gradient vector, decreasing the loss of the neural network. The process of training the network is called back-propagation as the calculation of the gradient proceeds backwards through the network, starting at the final layer of weights and ending at the first layer. Back-propagation is thus a way of propagating the total loss back into the neural network to find out how much of the loss each neuron is responsible for, adjusting the weights in such a way that it minimizes the error by giving neurons with

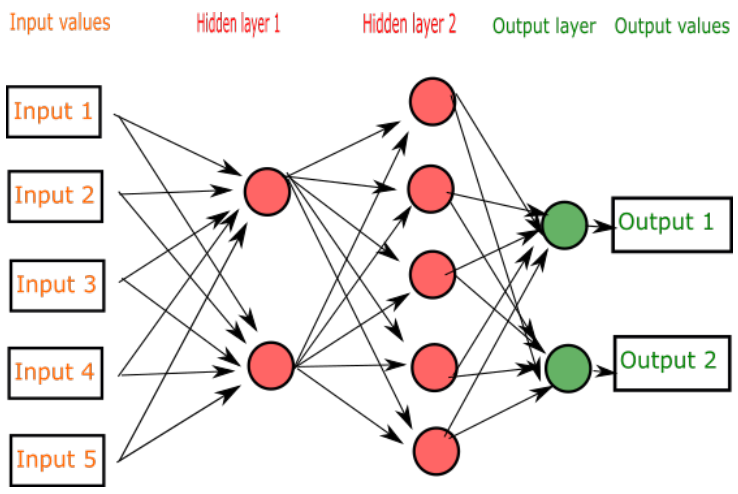


Figure 2.7: A simplistic overview of a neural network with two hidden layers of neurons. The arrows show how all neurons are connected and how data travels from the input to the output layer.

higher error rates lower weights and vice versa. Since partial computations of the gradient are reused from one layer to the next, it allows for more efficient computation of the gradient at each layer compared to calculating the gradient at each layer separately. The training process increases the accuracy of the learning algorithm, enabling prediction of new data.

Ordinary neural networks consist of fully connected layers where all neurons are in contact with each other (**Fig. 2.7**). Convolutional Neural Networks (CNN), however, are characterized by convolutional layers where each neuron is only connected to a few neurons in the previous layer, giving convolution of the signal. The convolution leads to a reduction of parameters and re-usability of weights, making the network better at capturing spatial and temporal dependencies. This has made CNN the leading image classification method, outperforming other methods in both accuracy and computational cost (Diale et al., 2019; Sharma et al., 2018). A typical CNN architecture consists of the following layers:

- **Input layer:** The input layer is the first layer of the CNN, which passes the images onto further layers for feature extraction. All preprocessing of images (resizing, removing colors etc.) must have taken place beforehand.

- **Convolutional layer:** The next few layers are convolutional layers that act like feature extractors with the objective of extracting high-level features (edges, lines, curves etc.) from the input images at a low computational cost. In a convolutional layer, a filter sweeps over the input features, enclosing a local region. This local region, often called receptive field, is then convoluted (dot product) with the filter, resulting in a single scalar (**Fig. 2.8**). By iteratively moving the filter, the convolution process is repeated for all input features. The number of features (pixels) the filter moves in each iteration is determined by the stride. A larger stride will decrease the overlapping of receptive fields, that is decrease the number of times each feature participates in convolutions. This can help prevent overfitting of the training images, i.e. help the CNN generalize beyond training data. Another important technique when performing convolution of the images, is padding. Padding consists of adding zeroes to the input matrix symmetrically. Without padding, the spatial resolution of the output of the convolutional layers is reduced compared to the input. By applying padding, however, the spatial resolution of the images stays the same. This is helpful, as preserving the dimension makes the design of the network easier and also allows for deeper networks without the reduction in resolution happening too quickly. Also, as the input matrix is extended, the pixels on the edges will appear in more convolutions, making the CNN benefit more from the information located in the outer regions of the images.

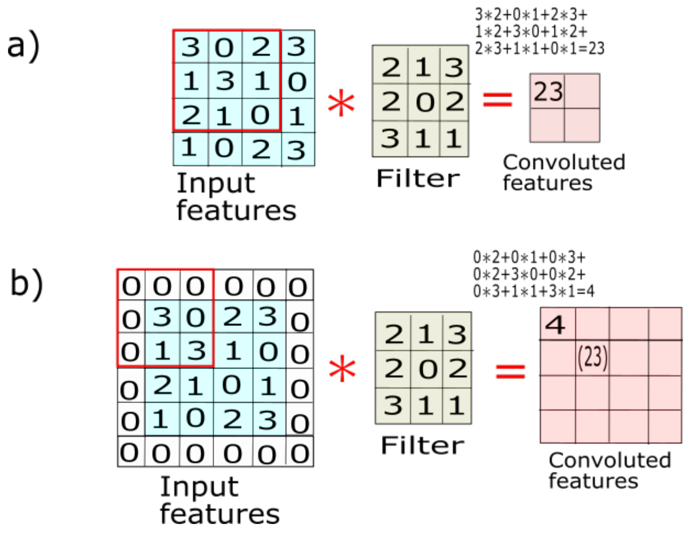


Figure 2.8: An example of a convolutional layer in a CNN. Input features are convoluted with a filter to create a single convoluted feature. This process is repeated by moving the filter (red square in figure) until all input features take part in at least one convolution. a) represents a convolution with stride one and no padding. The spatial resolution of the convoluted features is reduced compared to the input features. b) represents a convolution with stride one and padding. In this case, the spatial resolution is conserved in the convolution process. Note how the outer edges of the input features in a) will appear in more convolutions with the use of padding in b). Also, the convoluted feature values of a) will all be present in the convoluted feature values of b).

- **Pooling layer:** The pooling layer takes the extracted features from the convolutional layer, sweeps over them with a window (filter) of a given size, and chooses the most dominant one for each window (**Fig. 2.9**). As for the convolutional layer, this filter is moved by a given stride before the process is repeated for the whole set of input features. In determining the most dominant feature, max pooling is often used, where only the feature with the maximum value is extracted for each window. The function of the pooling layer is reducing the spatial size of the convoluted features while preserving the most valuable information.

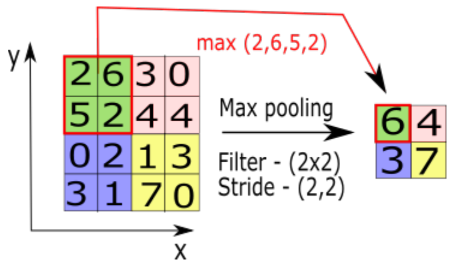


Figure 2.9: An example of max pooling in a CNN. A filter of size 2×2 sweeps over the features with a stride of 2 in both x and y direction. The max feature values are extracted and preserved for further processing.

- **Fully connected layer:** The fully connected layer is the same as the hidden layers for the ordinary neural network seen in **Fig. 2.7**. The objective of the fully connected layer is, as described above, to take the high-level filtered images and translate them into labeled categories by adjusting the weights of the neurons to minimize the training error. The fully connected layer are often followed by a technique called dropout. Dropout is a regularization technique that consists of dropping a neuron, i.e. temporarily removing it from the network along with all incoming and outgoing connections. By randomly dropping neurons with a given probability, the training process becomes noisy by effectively creating a slightly different configuration for the network. The idea is that dropout will break up situations where layers co-adapt to correct mistakes from prior layers. This will prevent overfitting the model to the training data, and hence lead to a more robust model.
- **Output layer:** The final layer of the CNN is the output layer. The output layer follows the last fully connected layer, with the objective of transforming the output of the fully connected layer to a probability distribution of the input belonging to a specific class.

An example of a complete Convolutional Neural Network can bee seen in **Fig. 2.10**. Convolutional layers and pooling layers represent the feature extraction in the CNN, while fully connected layers and the output layer represent the classification part of the model.

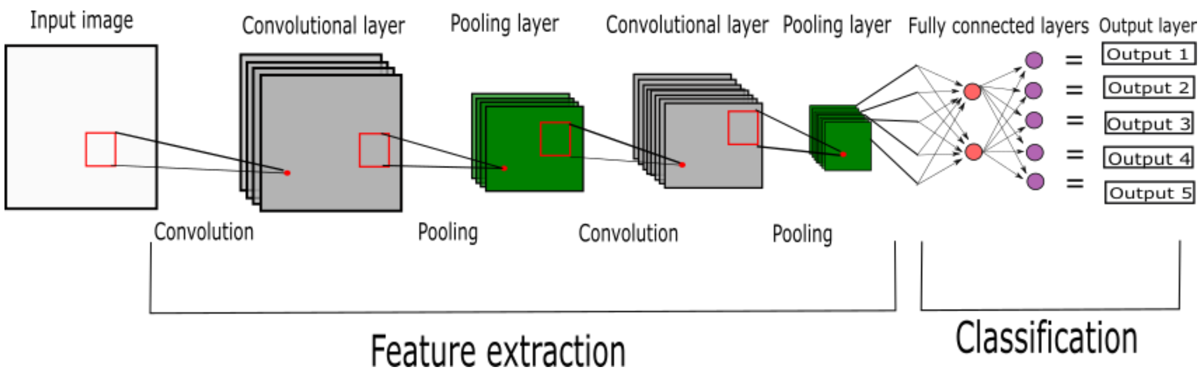


Figure 2.10: An example of a complete Convolutional Neural Network. An input image is fed into a sequence of convolutional layers and pooling layers to extract features from the image. Both convolution and pooling create a single feature from multiple feature values in the prior layer. The spatial resolution shrinks in the pooling process, while the use of padding preserves the spatial resolution for the convolution process. The extracted features are passed onto fully connected layers where they are translated into categorical labels. Finally, the output layer gives the probabilities that an input image belongs to the respective categories.

Chapter 3

Method

As seen in chapter 2, there are many ways to both characterize and classify snow. The focus of this report is characterizing and classifying 2-D images of snow in terms relevant to the performance of skis. The following presents a way of characterizing snow in terms of grain shape and grain size through the calculation of OED and dendricity of snow grains. This is achieved using an image analysis method involving contours. Additionally, the machine learning algorithms SVM and CNN are used to classify snow images as old or new snow, providing a separation between the two classes.

3.1 GelSight

The snow analysis in this report is based on the images from the GelSight Mobile™ equipment (GelSight, 2019). The GelSight is a handheld and portable instrument giving detailed surface analysis of any material. Such an instrument has been acquired by Olympiatoppen to be used in analysing both snow and ski surfaces. The instrument can bee seen in **Fig. 3.1**. The setup consists of a handheld GelSight instrument connected to a tablet/PC. An elastomeric sensor combined with a silicone gel conforms the surface topography, revealing small changes in the microstructure which are further captured in high-resolution images. These images are instantly captured and uploaded to the computational device. The images taken with the GelSight have an image size of 2464×2056 pixels, corresponding to 16.9×14.1 mm.

Figure 3.1: The GelSight Mobile acquired by Olympiatoppen. It consists of the GelSight instrument connected to a tablet/PC (GelSight, 2019)

3.2 Characterization of Snow Images

The GelSight instrument produce clear high-resolution images as seen in **Fig. 3.2**. In order to characterize these images, grain shape and grain size were chosen as the describing characteristics through the calculation of dendricity and OED, respectively. These parameters were calculated by finding the contours of snow grains in the images. Contours can be thought of as a curve joining all continuous points along a boundary having the same intensity (section 2.1.2), and the following preprocessing of the images plays an important role in the contour finding process.

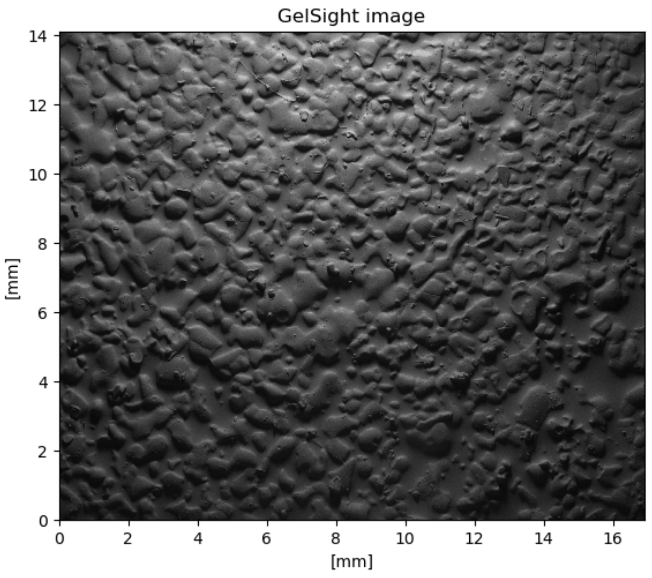


Figure 3.2: Example recording of snow by the GelSight instrument.

3.2.1 Preprocessing of Snow Images

First, the images are segmented by choosing an appropriate threshold value. That is turning all pixels below the threshold white and all pixels above the threshold black, giving a binary image. From **Fig. 3.2** one can see that the brightness of the image is uneven. There are more lighting at the top middle of the image and it gets darker when moving toward the other edges. A global threshold value for the whole image is therefore inadequate, as it would have to be chosen to either perform well on the bright or dark parts of the image. However, using an adaptive threshold method where one uses separate threshold values for separate parts of the image will account for the difference in brightness. The adaptive threshold algorithm was carried out by creating a window of a chosen area and calculating the average pixel value of the enclosed pixels. This value was then used as the threshold value of the specific window, separating the pixels into black and white. Repeating this process over the whole image will give the segmented image.

3.2.2 Contouring and Calculation of Parameters

The contouring is done in *Python*, using the *OpenCV*-library (Bradski, 2000) of *Python*-bindings, frequently used to solve computer vision problems. Finding the contours is achieved by using the *findContours()* function, which is based on the topological structural analysis by Suzuki and Abe (1985).

When the contours are found, one can calculate the enclosed area using Green's theorem. The theorem states that for a positively oriented, piecewise smooth and simple closed curve C enclosing a region D, one has the relation

$$\oint_C (A\,dx + B\,dy) = \iint_D (\frac{\partial B}{\partial x} - \frac{\partial A}{\partial y})\,dx\,dy \tag{3.1}$$

where A and B are functions of (x, y) having continuous partial derivatives on D. By choosing A and B such that $\frac{\partial B}{\partial x} - \frac{\partial A}{\partial y} = 1$, the right side of equation (3.1) will simply give the area of the enclosed region. Thus, the contour area (grain area) can be calculated from the line integral, as implemented in *contourArea()* function from the *OpenCV*-library.

As mentioned in section 2.1.1, OED of a non-spherical snow grain is equal to the diameter of a spherical snow grain exhibiting the same properties, and is a term often used for characterization of grain size. Hence, in two dimensions, the calculation of OED becomes

$$OED = \sqrt{\frac{4 \times A}{\pi}}, \tag{3.2}$$

where A is the area of a single contour. The calculation is then repeated for all contours in the image. For characterizing the shape of the images, the perimeter of the grains were found by counting the number of pixels making up the contour. Having the contour area, A, and the perimeter, p, the dendricity of a single snow grain can be calculated as (section 2.1.1)

$$D = \frac{p^2}{4\pi \times A}. \tag{3.3}$$

3.3 Classification of Snow Images

To classify the snow images, SVM and CNN are used as learning algorithms. As the feature extraction process differs for the two models, it is convenient to apply each method to investigate if one is better at capturing the characteristics of the images, as this will lead to a better performance for the classification. Also, where the SVM is shown to perform well on relatively small datasets, neural networks like CNN have shown to scale better on larger, more complex datasets with respect to accuracy and computational time (Kaensar, 2013). Hence, there lies interest in applying SVM on the relatively small dataset presented in this report, but also to build and apply a CNN to form a foundation for larger datasets and more complex snow image classification tasks in future work.

3.3.1 Dataset and Preprocessing of Snow Images

The snow images used for image classification are acquired from different ski arenas around Europe by Olympiatoppen. In addition to the images, Olympiatoppen also collects data of the snow conditions, including labeling the snow as new/old. The time it takes before new snow lying on the ground is characterized as old snow varies, as this is dependent on the metamorphism of snow, which again is highly dependent on environmental factors like temperature, wind etc. Hence, there exists no absolute limit where the snow goes from new to old, and the process can be seen as a smooth transition. However, the waxing team at Olympiatoppen have years of experience working with different snow conditions, making them qualified to make a distinction between new and old snow for ski purposes, which again is the aim of this report.

The used training data consists of a total of 276 images, where 140 of them are labeled as new snow and 136 are labeled as old snow. To improve the performance of the classifiers, this dataset has been expanded by increasing the number of images. The reasons for this is (1) to give the learning model more images to learn from, and (2) to even out the number of images in the two categories. Both will help the learning algorithms prevent overfitting, i.e. help the models capabilities of generalizing beyond training data. Increasing the number of images has been done by including images with different illuminations. The GelSight instrument creates a total of six versions of the captured image, where the difference lies in where the illumination takes place. There are four images with illuminations in corners, while there are two images with illumination at the top and bottom of the images. As there were fewer original (illumination at the top) images of new snow, the number of images with different illumination is higher for this category, although images with different illuminations have also been included for the category of old snow in order to minimize the discrepancies between the two classes. In addition to the use of images with different illuminations, the dataset is expanded by rotating images as well. For every image in the dataset, a duplicate image rotated 90 degrees is included. As the snow conditions on the ground should be found independent of the rotations and illuminations of the images, the two data augmentation methods will test the classification algorithms' abilities to concentrate on the relevant information in the images, creating more robust classification models.

The test data consists of 96 images, where 48 of them are labeled as new snow and 48 are labeled as old snow. This makes the amount of images being for testing and training roughly 25% and 75%, respectively. As for the training set, the number of images in the test set have been expanded by including the same image with different illumination and creating rotated images. Here, however, the number of images with different illuminations is the same for both categories. The test and training data are distinct from each other in the way that the same images, no matter rotation or illumination, do not appear in both the training set and the test set.

3.3.2 Support Vector Machine

Before being fed into the learning algorithm, the grayscale images in the dataset (**Fig. 3.2**) are loaded into feature vectors where each pixel represent a feature. To reduce the computational cost of the SVM, original images of size 2464×2056 are reduced to 492×410 pixels, that is reduced to 20% of their original size. Further, the features are standardized to have unit variance and zero mean, as this is shown to benefit both accuracy and computational cost (Juszczak et al., 2002).

The SVM algorithm is implemented in *Python* using the machine learning library *scikit-learn* (Pedregosa et al., 2011) and the *Support Vector Classifier* function. The kernel used in the SVM algorithm is a radial basis function (RBF). The RBF kernel is given as

$$K(\vec{x_i}, \vec{x_j}) = e^{-\frac{||\vec{x_i} - \vec{x_j}||^2}{2}}. \tag{3.4}$$

Hence, the RBF is a monotonically decreasing function of $||\vec{x_i} - \vec{x_j}||^2$, the Euclidean distance between feature vectors. The interpretation of the RBF kernel is therefore as a similarity measure, where similar feature vectors will give rise to higher values of the kernel function. The *Python* code for the implementation of the SVM algorithm can be seen in appendix A.

3.3.3 Convolutional Neural Network

As for the SVM case, the images are reduced to 20% of their original size before standardized to have unit variance and zero mean. Further, the CNN is implemented in *Python* via the deep learning library *Tensorflow* (Abadi et al., 2015) and the application program interface (API) *Keras* (Chollet et al., 2015).

The architecture of the CNN model can be seen in **Fig. 3.3**. Here, both the convolutional layers and pooling layers use a stride of $(1, 1)$. The convolutional layers utilizes padding, and the pooling layers apply max pooling. Further, the first and second fully connected layer consist of 30 and 15 neurons, respectively, with the second fully connected layer followed by the dropout technique. For more details on the specific parameters of the CNN model, see appendix A.

The model is trained with a batch size of ten. This means that the model will take the first ten images in the training data, process them, before updating the internal weights of the model. For the 276 training images this gives a total of 28 batches, 27 of them with ten images and one with six images. When all the training images are processed once (all

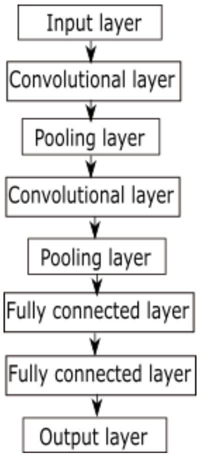


Figure 3.3: The architecture of the CNN model used for the image classification of snow. All convolutional layers and pooling layers use a stride of $(1, 1)$. The convolutional layers use padding and the pooling layers apply max pooling. The first and second fully connected layer consist of 15 and 30 neurons, respectively. The second fully connected layer is followed by the dropout technique.

28 batches), the model has trained one epoch. For this classification problem, 75 epochs are used in training. The chosen optimization algorithm for updating the weights in the network is a gradient descent procedure called ”Adam” (Kingma and Ba, 2014).

The loss function used for the CNN model is categorical cross entropy, which is defined as

$$E(y_{\mathrm{t}}, y_{\mathrm{p}}) = -\sum_{i} y_{\mathrm{t}} \log(y_{\mathrm{p}}). \tag{3.5}$$

Here y_{t} and y_{p} are the true and predicted labels, respectively. The summation is over all images, and the minus sign is included to get positive loss values. For the two-class classification problem in this report, the image is either old snow, $y_{\mathrm{t}} = [0, 1]$, or new snow, $y_{\mathrm{t}} = [1, 0]$. The predicted probabilities of an image belonging to either old or new snow can for example take values like $y_{\mathrm{p}} = [0.2, 0.8]$ or $y_{\mathrm{p}} = [0.8, 0.2]$. The accuracy of the model is calculated by comparing the index of the maximum value in y_{t} with the index of the maximum value in y_{p}. If they are similar, the labels are correctly predicted, if not, they are wrongly predicted. Hence, dividing the number of correctly predicted samples on the total number of images gives the accuracy of the CNN model.

Chapter 4

Results

4.1 Characterization of Snow Images

4.1.1 Preprocessing of Snow Images

For snow characterization the images are preprocessed by applying the adaptive threshold method, giving results as seen in **Fig. 4.1**.

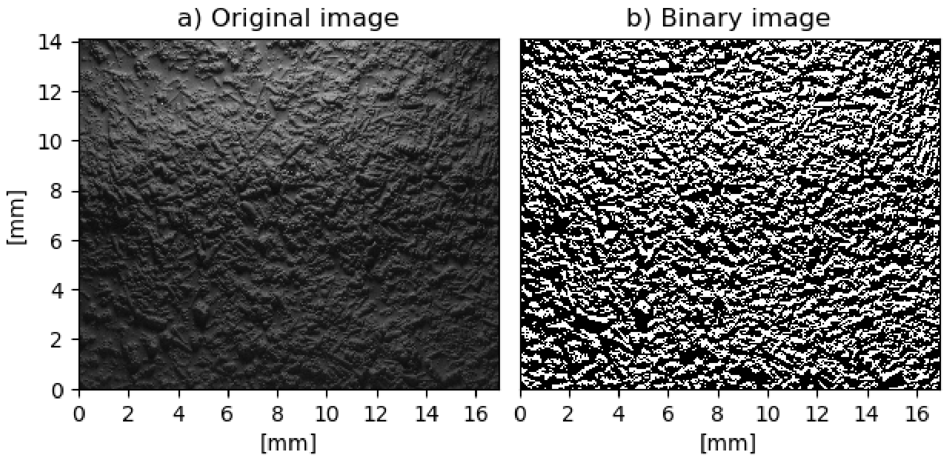


Figure 4.1: The figure shows a) the original image and b) the binary image after the adaptive threshold method is applied.

The white and black pixel values represent the grains and grain outlines, respectively. Since the thresholding algorithm uses the mean pixel value among a chosen amount of neighbouring pixels as threshold, the grains in the original images must have higher intensity compared to the grain outlines for the algorithm to work properly, as seems to be the case in the figure. However, images with different illuminations will cast different shadows that will alter the pixel values in the original images, which will in turn lead to changes in the binary images as well. **Fig. 4.2** shows the same image as in **Fig. 4.1** but with different illumination. From the figures it is seen that lighting conditions affects which parts of the original images that are highlighted and which parts that are not. These differences create distortions in the binary images.

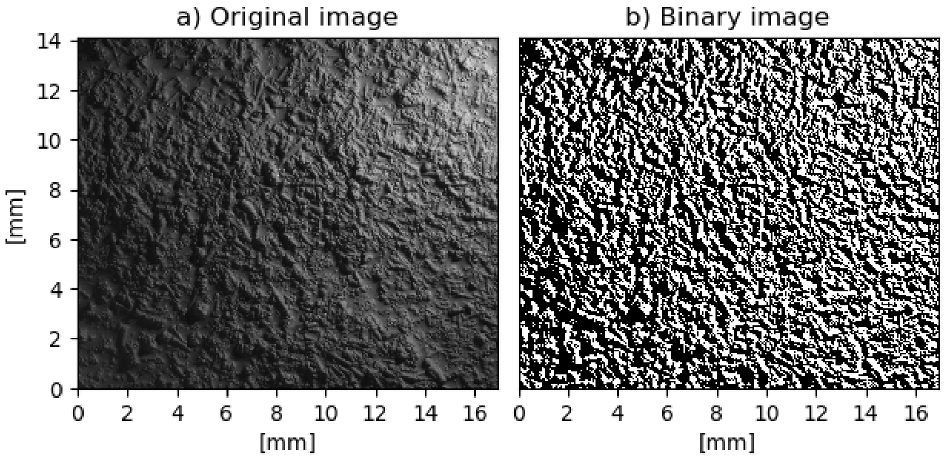


Figure 4.2: The figure shows a) the original image and b) the binary image after the adaptive threshold method is applied. The original image used is the same as in **Fig. 4.1** but with different illumination.

As the binary images lay the foundation for further contouring and calculation of parameters, the segmentation of images is crucial for the accuracy of the characterization. Since the size and shape of snow grains is independent of lighting conditions, so should the calculated OED and dendricity. Thus, the aforementioned distortions for binary images preprocessed from original images with different illuminations should be as small as possible. **Fig. 4.3** shows the average dendricity and OED for the two different illuminated snow images presented above. The calculated parameters for the two images are similar, with a relative difference below 3 % for both the dendricity and OED.

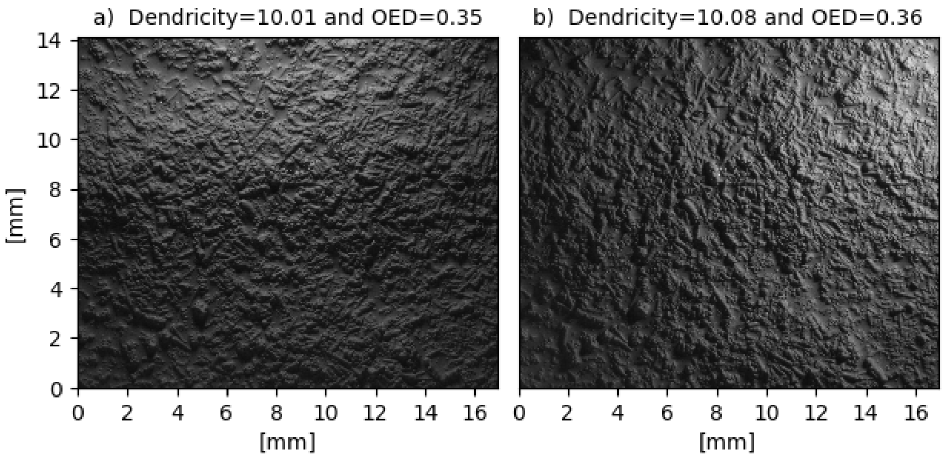

a) Dendricity=10.01 and OED=0.35
b) Dendricity=10.08 and OED=0.36
[mm]
[mm]
[mm]

Figure 4.3: The two original images in **Fig. 4.1** (a)) and **Fig. 4.2** (b)) with the average OED and dendricity shown. The two images show the same snow grains with different illumination. Both the dendricity values and OED values are similar for the images.

4.1.2 Contouring and Calculation of Parameters

Following the image segmentation is the contouring of the binary images, where curves joining all continuous points having the same intensity are found and drawn. **Fig. 4.4** shows the whole contouring process from the original image to the contoured image. From the figure one can observe the complexity of the images in form of the dense distribution of grains, resulting in the dense distribution of drawn contours.

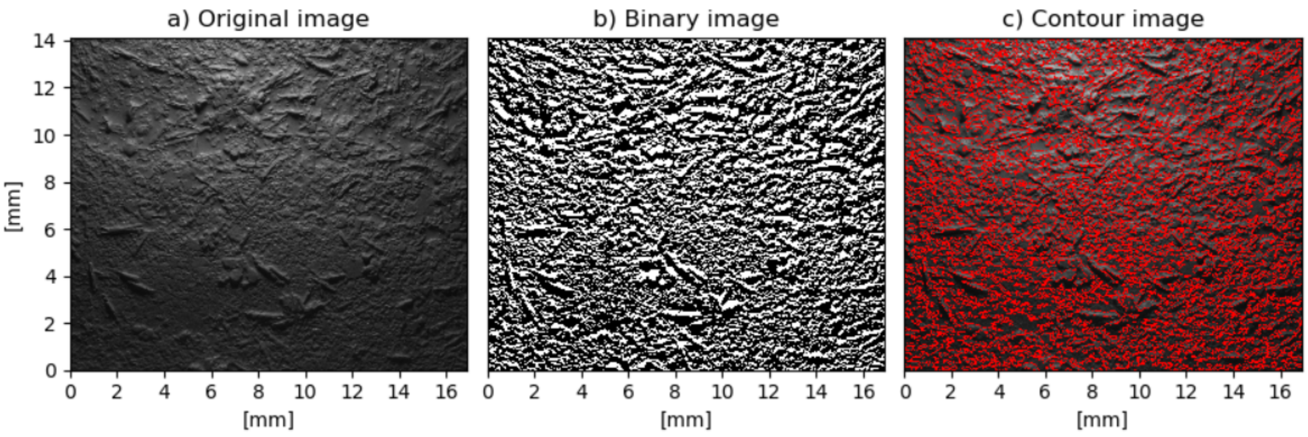


Figure 4.4: The contouring process of a snow image. The figure shows a) the original image, b) the binary/segmented image and c) the original image with drawn contours.

From the contours one can approximate the area of the snow grains and calculate the OED and dendricity of snow images, as described in section 3.2.2. **Fig. 4.5** shows the average OED and dendricity of two different snow images. Image a) has sharper, more dendritic grains compared to the more rounded grains of image b). This leads to a higher calculated dendricity for image a). For the OED, the rounder grains of image b) appear larger compared to the dendritic grains of a), giving a larger OED for image b).

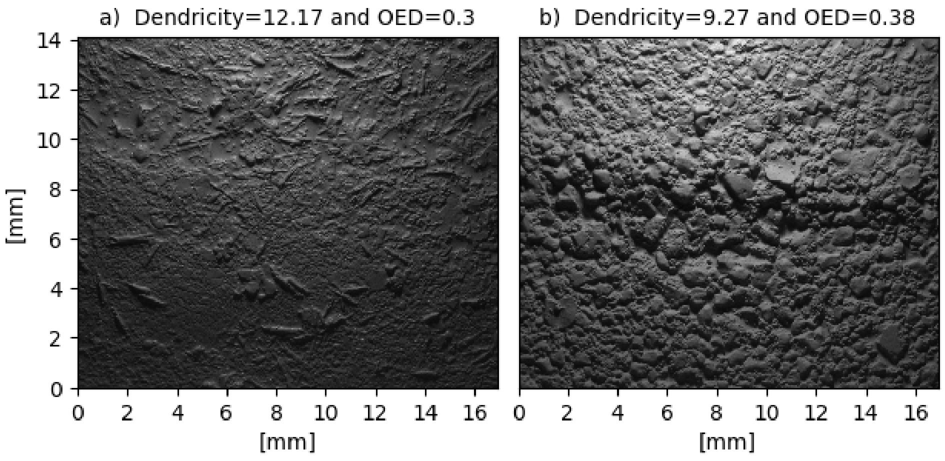

a) Dendricity=12.17 and OED=0.3
b) Dendricity=9.27 and OED=0.38
[mm]
[mm]
[mm]
0
2
4
6
8
10
12
14
16

Figure 4.5: Two snow images of different snow conditions. The sharper grains of a) have a higher average dendricity compared to the more rounded grains of b), while the larger grains in b) have a higher OED compared to a).

From the figure one can also observe that the snow grains in the respective images are not uniform, but vary in both shape and size. These local changes are due to factors like weather, terrain, physical disturbances etc. The average value of the calculated parameters do therefore not give a complete overview on the distribution of the shape and size of snow grains as different distributions can give similar averages, and also images with different averages can share more or less similarities in their distributions. Hence, beside the average values of the OED and dendricity it is also interesting to look at their distributions. **Fig. 4.6** shows the images in **Fig. 4.5** and their distribution of OED and dendricity. For the OED, the distributions look pretty similar with most of the grains being in the range of $0.10 - 0.20$ mm for both images. The difference between the two comes from image b) having a higher percentage of larger grains compared to a), leading to the higher average OED of b). Looking at the dendricity distributions of the grains, there are larger differences between the images. Image a) has a more flat distribution indicating that the the number of grains in each dendricity range is more similar compared to image b) where the major part of grains have lower dendricity values. This leads to the lower average dendricity value of image b). From the distributions one can also observe that the largest fractions of grains for the two images have both lower OED and dendricity than the average values, leveled out with a fewer number of larger and more dendritic grains.

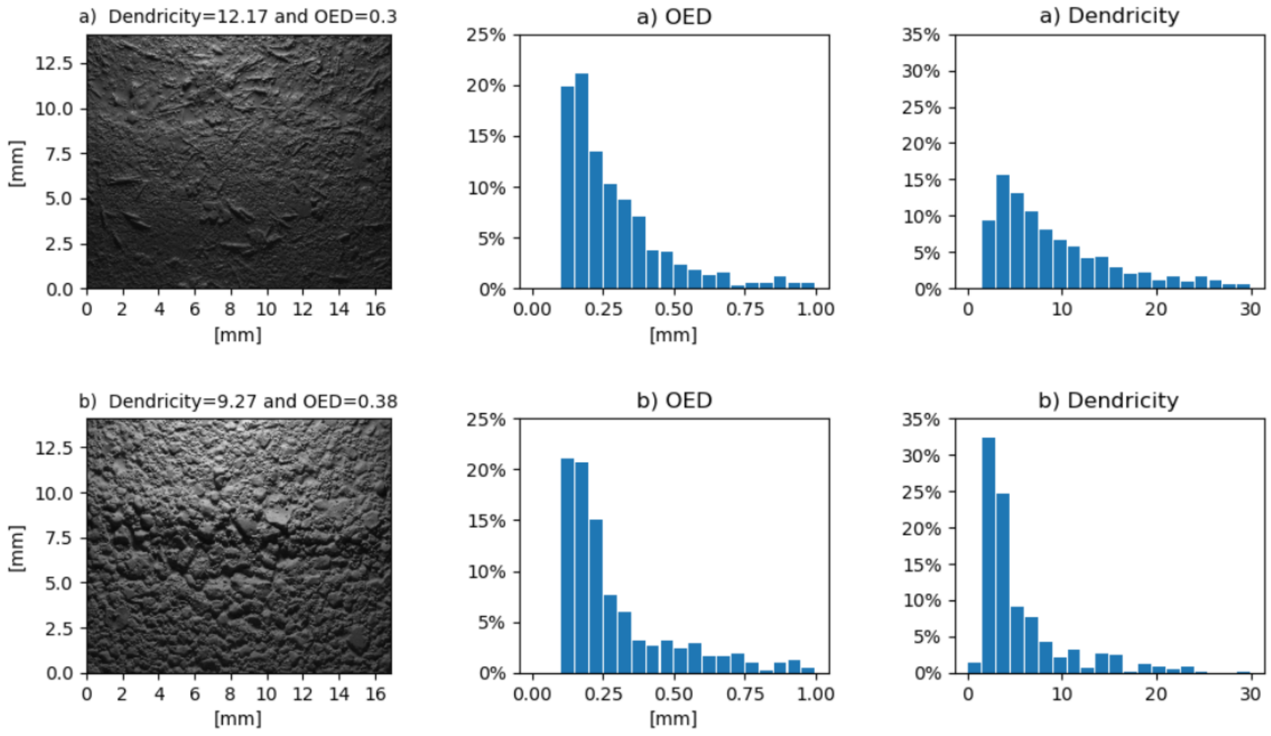


Figure 4.6: The two different snow images in **Fig. 4.5** with the distribution of OED and dendricity for the snow grains. The OED distributions are quite similar except b) having some higher fraction of larger grains compared to a). For the dendricity, a) has a flatter distribution compared to b) where most of the snow grains have smaller dendricity values.

For the presented characterization method to work properly, one should expect similar looking snow conditions to have similar calculated values of grain size and grain shape. **Fig. 4.7** show two images exhibiting similar average values of OED and dendricity. Comparing the two images one can observe that the snow grains look similar as well, with pointy, sharp grains appearing in both of the images. Further, their distribution of the OED and dendricity can be seen in **Fig. 4.8**. These distributions also look much alike, but image b) has a higher fraction of grains with both smaller dendricity and OED compared to image a). As for the two previously shown images in **Fig. 4.6**, one can observe that the largest fractions of the dendricity and OED are in the lower part of the scale, and that the average values are increased by a smaller amount of larger and more dendritic grains.

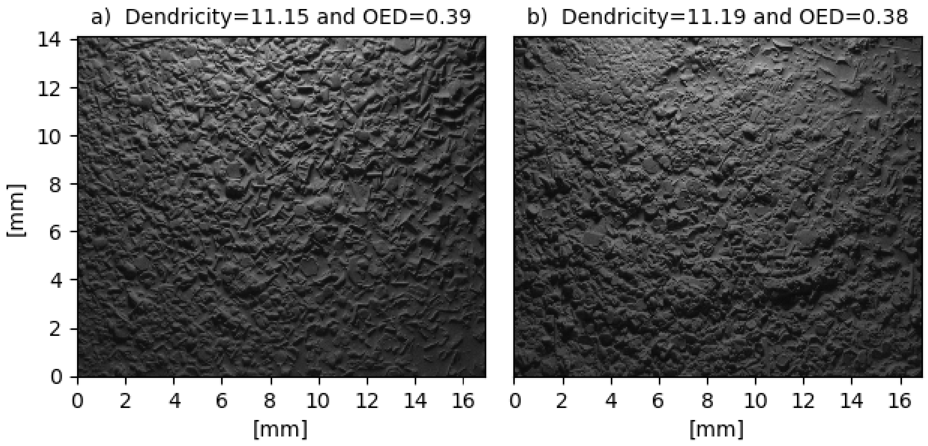

a) Dendricity=11.15 and OED=0.39
b) Dendricity=11.19 and OED=0.38
[mm]
[mm]
[mm]

Figure 4.7: Two snow images of similar snow conditions. The snow grains in the two images look similar and also have similar values for OED and dendricity.

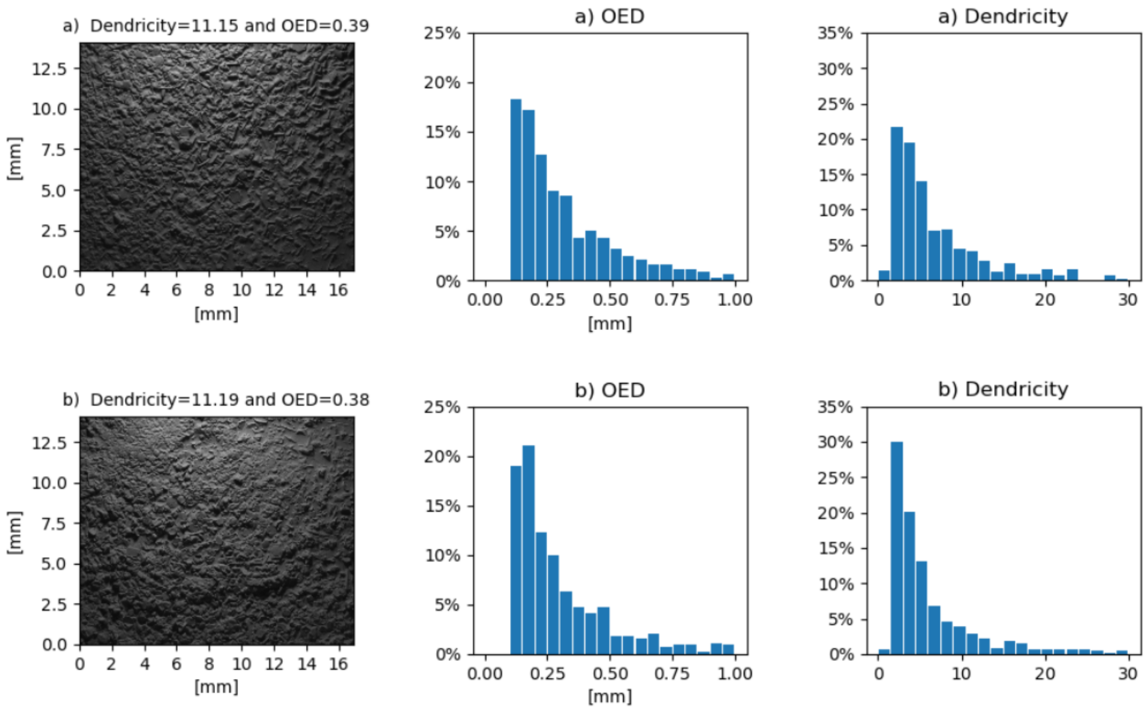


Figure 4.8: The two similar snow images in **Fig. 4.7** with the distribution of OED and dendricity for the snow grains. The distributions are quite similar but with image b) having a larger fraction of grains with both smaller dendricity and OED compared to a).

Thus far, the presented results investigate the abilities of the contouring method to distinguish between different snow conditions through the calculation of OED and dendricity, which is indeed one aim of this report. However, as these parameters represent actual physical values, it is interesting to look at their accuracy as well. This will indicate the validity of using calculated parameters from the contouring as stand-alone results, or if there are shortages in the contouring making the accuracy of these values inadequate. In order to validate the accuracy of the contouring, sandpaper is used as reference. The Federation of European Producers of Abrasives (FEPA) has compiled a standard for the grit sizes of different types of sandpaper. By sieving the grits they have estimated the mean diameter, which can be compared with the calculated OED from the contouring. Here, three sandpapers with different sized grits are used, as seen in **Fig. 4.9**.

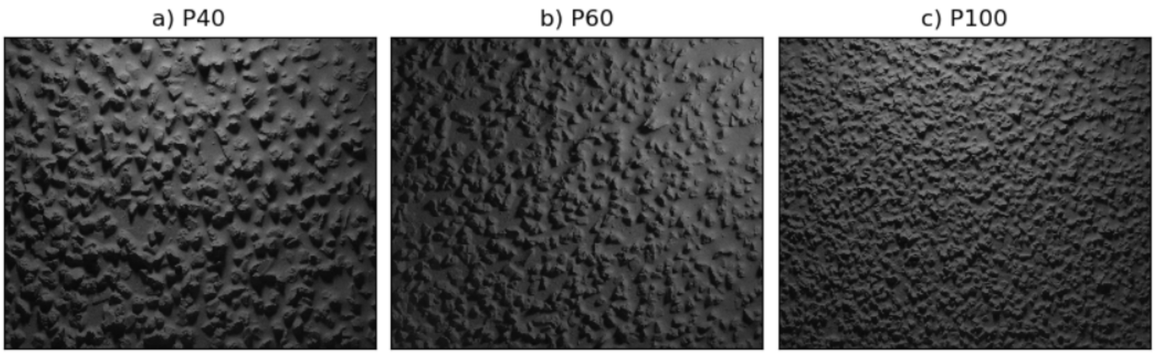


Figure 4.9: The three different types of sandpaper used to validate the contouring method. The lower the P-number, the larger the size of the grits.

The sandpapers are named P40, P60 and P100, and from the figure one can see that the grit size decreases as P-number increases. The same can be said for the density of grits, where the image of P100 clearly inhabits more grits compared to the other two images. The calculated OED along with estimated diameter from FEPA can be seen in **Table. 4.1**. The trend for both the OED and the FEPA diameter is the same, where moving up in P-number gives smaller grits, as expected from the figure. However, the FEPA values shrink considerably more compared to the calculated OED. For P40 the results are similar, but when moving to P60 and P100 there are large discrepancies between the values. This overestimation indicates that the contouring algorithm have trouble separating grits when the grits are small and the density of grits is high, as discussed in section 5.1.2.

Table 4.1: The calculated OED and estimated diameter from FEPA for different types of sandpaper. The decreasing trend of OED and FEPA diameter for increased P-number is the same, but much larger for the FEPA diameter.

Type	OED [mm]	Diameter (FEPA) [mm]
P40	0.424	0.426
P60	0.357	0.269
P100	0.30	0.162

4.2 Classification of Snow Images

4.2.1 Dataset and Preprocessing of Snow Images

The object of the image classification is to classify snow images into categories of new and old snow. The dataset used to train this classifier is labelled by Olympiatoppen, and some images with their respective labels can be seen in **Fig. 4.10**. As can be observed from the images, there are differences within each category, where images with the same label differ in appearance of both the size and shape of grains. Between the categories, the main difference is the more rounded grains of old snow compared to the more elongated grains of new snow.

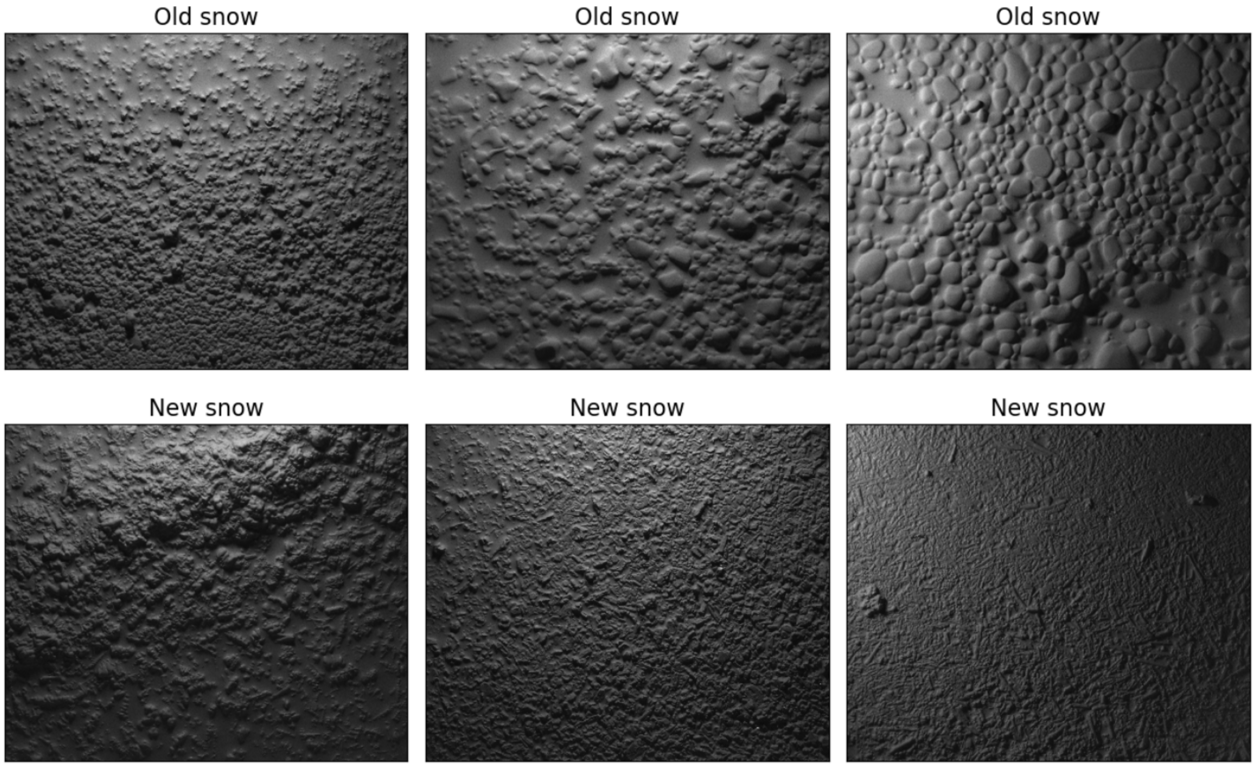


Figure 4.10: The figure shows images labelled as both new and old snow. The two categories are represented by a variety of images, where both the size and shape of grains differ for the same category. The main difference between the categories is the more rounded grains of old snow compared to the more elongated grains of new snow.

The dataset containing the images is first resized to lower the computational cost of the learning algorithms, as described in section 3.3.1. Secondly, the images are augmented by including images with different illuminations and creating rotated images. The results of the data augmentation and the preprocessing can be seen in **Fig. 4.11**. The original images of size 2464×2056 are resized to 492×410 pixels, that is reduced to 20% of their original size. From the figure one can see that the perceived changes between the original image and resized image are small and hard to detect. Hence, the loss of information when downscaling the images should not have any large impact on the performance of the classification algorithms. The figure also shows the same image with a different illumination and a rotation of 90 degrees, which are the used techniques to expand the dataset.

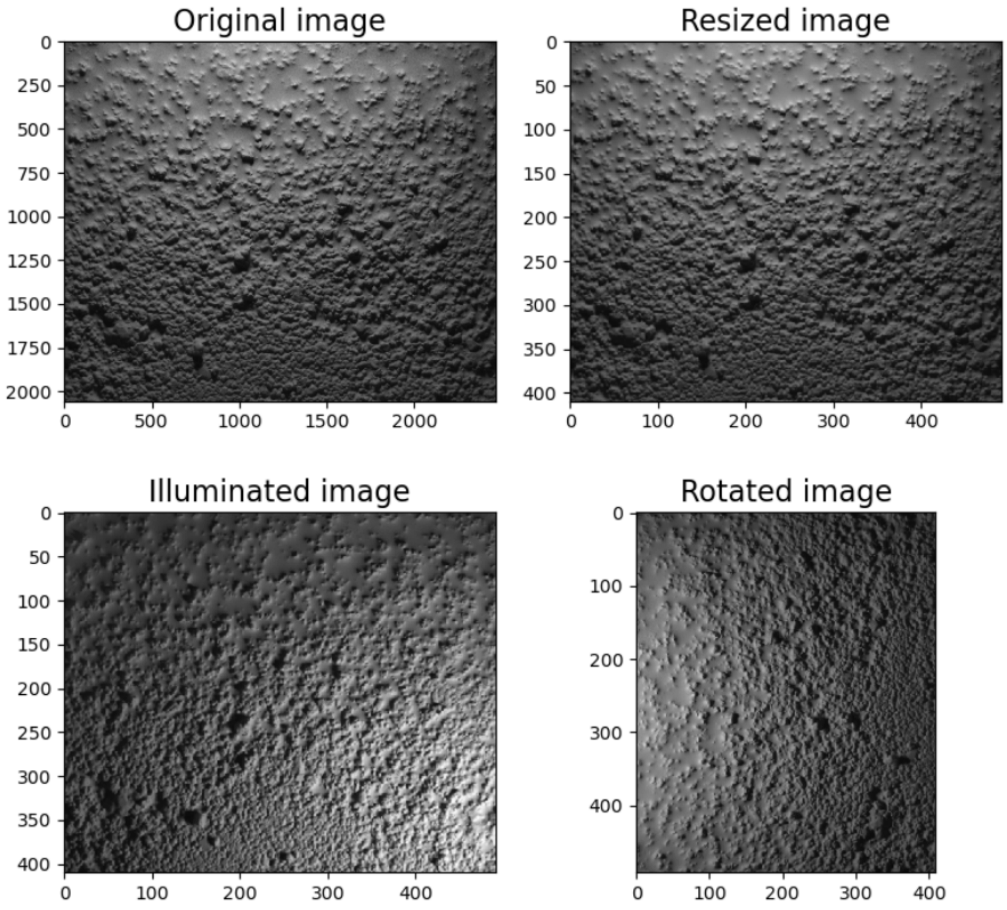
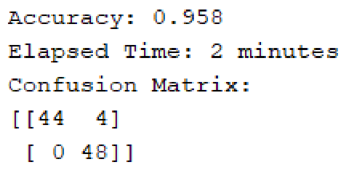


Figure 4.11: The figure shows the results of preprocessing and augmentation of an image in the dataset. The rescaling of the original image consisting of 2464×2056 pixels to 492×410 pixels leads only to tiny changes between the images. Further, the same image is included with different illumination and a 90 degree rotation, which techniques are used to expand the dataset.

4.2.2 Support Vector Machine

The performance of the SVM algorithm is assessed by using the trained learning algorithm to predict the labels of the images in the test set. The calculated accuracy, computation time and confusion matrix as retrieved from the *Python*-script can be seen in **Fig. 4.12**. The accuracy of the SVM algorithm is 95.8%. Hence, of all the 96 images in the test set, 92 of them are predicted correctly and 4 of them are predicted wrongly. The same conclusion can be drawn from the confusion matrix, where the first row indicates that out of the 48 images labelled as new snow, 44 of them are classified as new snow and 4 of them are classified as old snow. The second row indicates that all 48 images labelled as old snow are indeed classified as old snow. Thus, the four misclassified images are all images of new snow falsely predicted to be old snow.



```
Accuracy: 0.958
Elapsed Time: 2 minutes
Confusion Matrix:
[[44  4]
 [ 0 48]]
```

Figure 4.12: The accuracy, computation time and confusion matrix for the SVM algorithm. The accuracy is 95.8%, corresponding to 92 out of 96 images correctly classified. The confusion matrix shows that all misclassifications come from images of new snow predicted to be old, while all images of old snow are predicted correctly.

The wrongly classified images can be seen in **Fig. 4.13**. The four images consist of two non-rotated and two rotated images. The non-rotated images can be seen to picture the same snow but with slightly different quality and lighting, and the other two images are their respective rotations. Hence, the four misclassified images originates from the same snow conditions.

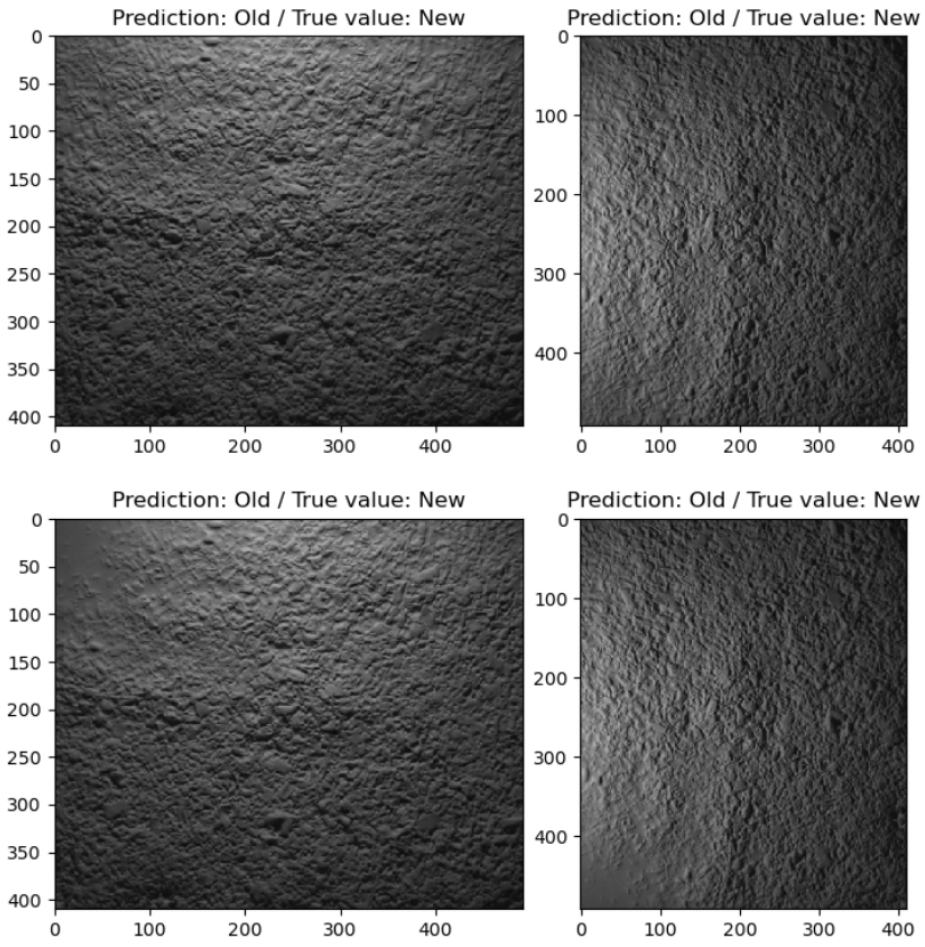
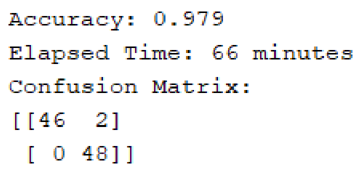

Prediction: Old / True value: New
Prediction: Old / True value: New
Prediction: Old / True value: New
Prediction: Old / True value: New

Figure 4.13: The figure shows the four images misclassified by the SVM algorithm. The two non-rotated images differ only in lighting and quality, and the two rotated images are their respective rotations.

4.2.3 Convolutional Neural Network

As for the SVM algorithm, the performance of the CNN is assessed by using the trained learning algorithm to predict the labels of the images in the test set. The calculated accuracy, computation time and confusion matrix as retrieved from the *Python*-script can be seen in **Fig. 4.14**. The accuracy of the CNN algorithm is 97.9%. Hence, of all the 96 images in the test set, 94 of them are predicted correctly and 2 of them are predicted wrongly. The confusion matrix shows that the two misclassified images are images of new snow falsely predicted to be old snow.



```
Accuracy: 0.979
Elapsed Time: 66 minutes
Confusion Matrix:
[[46  2]
 [ 0 48]]
```

Figure 4.14: The accuracy, computation time and confusion matrix for the CNN algorithm. The accuracy is 97.9%, corresponding to 94 out of 96 images correctly classified. The confusion matrix shows that all misclassifications come from images of new snow predicted to be old, while all images of old snow are predicted correctly.

The wrongly classified images can be seen in **Fig. 4.15**. These images are the same images misclassified by the SVM algorithm as well, and are identical but for a 90 degree rotation. Hence, the two misclassified images originate from the same snow conditions.

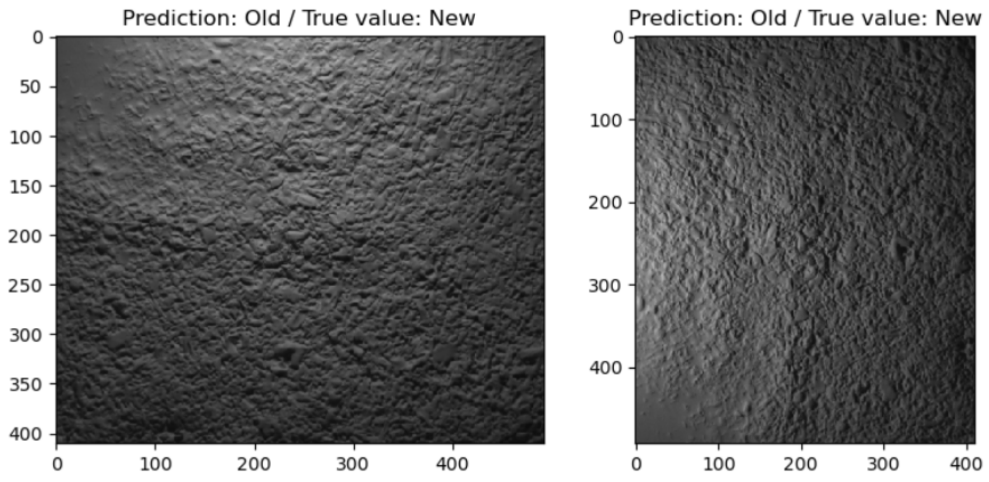

Prediction: Old / True value: New
Prediction: Old / True value: New
0
50
100
150
200
250
300
350
400
0
100
200
300
400
0
100
200
300
400
0
100
200
300
400

Figure 4.15: The figure shows the two images misclassified by the CNN algorithm. The images are identical but for a 90 degree rotation.

The CNN model trains by adjusting its weights to minimize the loss function and hence increase the accuracy of the classification. This training process can be seen in **Fig. 4.16**, where the loss and accuracy are plotted for each trained epoch. The figure also shows how the accuracy and loss on the test set evolves during the training process. It is worth repeating that the test and training set are independent of each other, and that the test set has nothing to do with the tuning and optimization of the CNN model, but is merely included to give insights to the model's performance. From plot a) one can see that the training accuracy rapidly increases after just a few epochs, while the accuracy of the test set is above 90% after just one trained epoch. Further, the accuracy of the training and test set increases before stabilizing above 95% with a slightly higher accuracy on the training set. From the model loss in plot b), one can observe a rapid decrease for both the training and test set after just a few number of trained epochs, followed by a slightly decrease before the curves eventually flatten out. As for the model accuracy, the loss on the training and test set behave similarly during the training process.

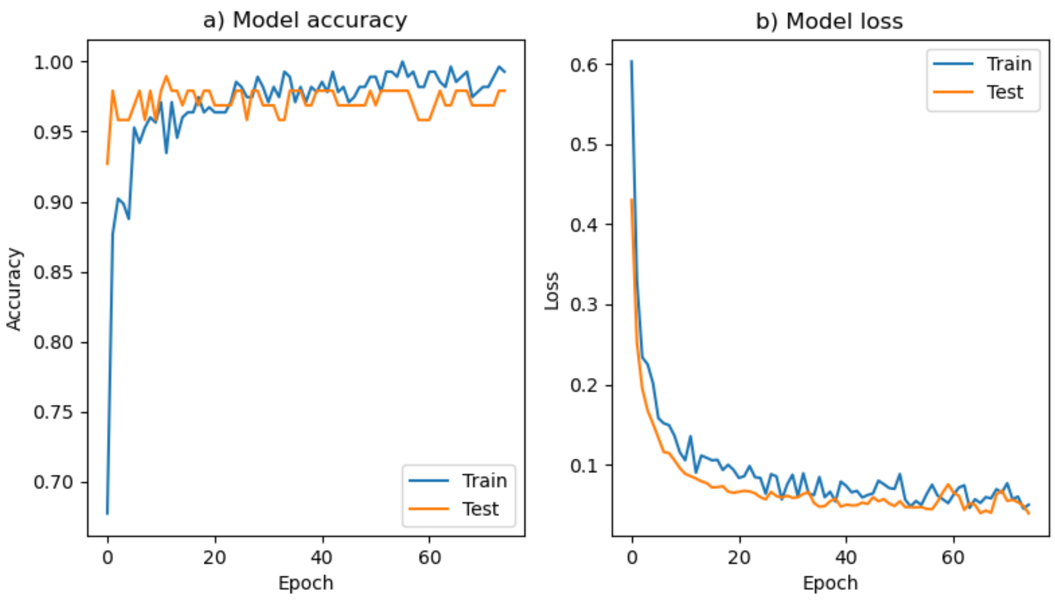


Figure 4.16: The figure shows a) the model accuracy and b) the model loss for the training and test set as a function of trained epochs.

Chapter 5

Discussion

5.1 Characterization of Snow Images

5.1.1 Preprocessing of Snow Images

The preprocessing of digitized images forms the basis for further analysis, and good results in this initial step are therefore crucial to achieving good results on the characterization process as a whole. However, the preprocessing of images is no straightforward task, and from **Fig. 4.1** one can see the complexity of the snow images in form of the dense distribution of grains. As a consequence of this, some grains can be seen to overlap with others. Hence, locating where a grain starts and stops is not trivial, not for the human eye nor the threshold algorithm. As described in section 3.2.1, the adaptive threshold algorithm creates local thresholds based on the intensity of surrounding pixel values. Thus, the number of neighbouring pixels used to calculate the average thresholds will impact the results of segmentation. If too many pixels are chosen, the algorithm will be unable to detect small grains and capture small details in the images. On the other hand, if too few pixels are chosen, the algorithm will be overly sensitive to tiny changes in pixel values and also noise, making it unable to capture larger grains. The optimal number of neighbouring pixels will vary for different regions in an single image, and also between different snow images, due to the difference in size, density and complexity of the snow grains. Hence, the optimal number of neighbouring pixels to best capture the grains and grain boundaries for all images is difficult to find.

As the threshold algorithm operates on raw pixel values, it is sensitive to the lighting of the images. The lighting will affect which parts of the images become highlighted and appear brighter, and which parts become shaded and appear darker. A global threshold for the whole image will not be able to separate the grains and grain boundaries for both the darker and brighter parts of the image. However, as the shape and size of snow grains stay indifferent to the lighting, so should the calculated OED and dendricity. Here the adaptive threshold method comes in handy, as this method should capture nuances in local regions and thus work for both the brighter and darker parts of the images. **Fig. 4.3** shows two images of the same grains but with different illuminations. Both the dendricity and OED are similar for the two images with a relative difference below 3%. This indicates that the adaptive threshold method indeed provides a segmentation of images enabling characterization of grains independent of illumination.

5.1.2 Contouring and Calculation of Parameters

Finding the contours is equivalent with finding a curve along a boundary joining points with the same intensity, and is completely dependent on the quality of both the original images and the segmentation process. Thus, noise in the original images or inaccurate segmentation of the binary images will have negative effects on the contouring as well. One effect can be clustering multiple grains into a larger grain. As there is a high density of grains in the images, separating them from each other can be hard, and clustering of grains can therefore occur. In addition, one can have the other way around, where larger grains are divided into smaller grains. In both cases, there are small inaccuracies in the pixel values that can be decisive, leading to detection of a smaller or larger grain area. In order to account for that, the contour areas are filtered based on their size, setting a lower and a higher limit. This filters out the largest and smallest grains, which greatly benefited the consistency of the contouring and calculated parameter values.

For the contouring method to be a useful tool in ski preparation it should be able to distinguish between different and similar snow conditions. The two different snow images in **Fig. 4.5** show different values of OED and dendricity, where the sharper and smaller grains of image a) give both a higher average dendrictiy and lower OED compared to the more rounded and larger grains of image b). As for the similar snow conditions in **Fig. 4.7**, one has similar values of OED and dendricity. This indicates that the contouring method indeed provides a separation of snow conditions. Looking at their respective grain distributions in **Fig. 4.6** and **Fig. 4.8** one can observe that the snow grains in an image are not uniform, but vary in terms of size and shape, adding to the complexity of the characterization. From the same figures one can also observe that the largest fractions of grains inhabit smaller OED and dendricity compared to the average values, evened out by a fewer number of larger and more dendritic grains. Hence, the average values do not necessary give the best representation of the snow conditions. For the friction of skis, it is not certainly correct that all grains in the images should be given equal weight. Colbeck (1996) among others have found smaller grains to give a higher friction coefficient μ leading to higher overall friction (section 2.2) compared to larger grains. As most of the grains in the presented images have a smaller OED than the average values, it could be that the smaller grains should be weighted more to give an OED value in compliance with the actual friction of the snow. However, the friction between snow and skis is determined by the real contact area of the ski-snow interface and is a complex process depending on factors like shearing of meltwater film, deformation and fractures, plowing of snow in front of skis and so on. Hence, further processing of the OED and dendricity values accounting for all these factors is a complex task beyond the aim of this report. Thus, the values of OED and dendricity presented here are unprocessed and weighted equally. The range of sizes and shapes in each snow image indicates that both the average value and distributions should be used to thoroughly characterize the properties of the snow.

As previously mentioned, it is interesting to validate the accuracy of the calculated OED and dendricity values. This will indicate the validity of using calculated parameters from contouring as stand-alone results, or if there are shortages in the contouring making the accuracy of these values inadequate. The calculated OED and estimated diameter of different types of sandpaper in **Table. 4.1** show clear differences between the values. For P40 the values are similar, but as the grits becomes smaller the FEPA values shrink much more compared to the calculated OED. From **Fig. 4.9** one can observe how the density of grits increases when moving up in P-number, making it harder to distinguish between where different grits starts and stops. This can lead to clustering of multiple grits into larger grits, effectively overestimating the size of the grits. One can see from the table that this effect is more prominent for P100 where the density of grits is largest, making the separation of grits more difficult. As discussed for the preprocessing, the adaptive threshold algorithm can be made more or less sensitive by adjusting the number of neighbouring pixels used to create the local threshold. Making this number smaller makes the algorithm more sensitive to smaller changes and thus better at capturing smaller grains, but at the cost of underestimating the larger grains. As the size of snow grains in the images are more like the grits for P40 rather than P100, the contouring algorithm has been tuned to better fit the relatively larger grains of P40, as seen in the table.

The presented results show that the contouring method has flaws. The complexity of the snow images in terms of the density of grains and variations in shape and size makes it hard to accurately capture all grains, as seen from the comparison of the OED and FEDA diameter. Hence, as the contouring algorithm fails to capture the complexity of the GelSight images, the values of dendricity and OED should not be used as stand-alone results. For this purpose, more elaborate image analysis techniques like the previously mentioned micro-CT would be a better choice. However, the aim here is not to outperform such imaging techniques, but rather to investigate if a simple and fast analysis using the captured images from the GelSight instrument can say something about the current snow conditions and thus help the ski-technicians prepare better skis. As previously discussed, the image analysis is consistent as both similar snow conditions give similar values of dendricity and OED and different snow conditions give different values. Also, for the sandpaper, the contouring still correctly separates the images based on the size of the grains, even though the values are not accurate. This consistency of the contouring algorithm makes it useful for comparing different snow conditions. Finding snow conditions similar to previously analysed snow conditions will give the ski-technicians the opportunity to use their previous experience on these conditions to prepare the optimal skis. Hence, the presented contouring can be used as a tool in ski preparation.

5.2 Classification of Snow Images

Since the metamorphism of snow is a continuous process, one will have a continuous transformation of snow grains as well. This leads to the variety of snow conditions seen in **Fig. 4.10**. As seen from the figure, there is a wide range of images both within and between the two categories. Within each category there will also be an age difference between the images, ranging from more old to less old (old snow) and more new to less new (new snow). Hence, images between categories will be more similar if they are images of less old snow and less new snow, compared to if they are images of very new and very old snow. As there is no absolute limit to where new snow transforms into old snow, images from each category can end up quite similar, adding to the complexity of the classification. Another factor contributing to the complexity of the classification is, as seen from the contouring, the lack of uniformity in terms of the size and shape of snow grains. This could be due to differences in temperature, terrain, pressure etc. However, as one should expect neighbouring snow grains to experience similar impact from the surroundings, a more likely explanation is that new snow falling from the sky blends in with the older snow on the ground. This creates a mixed composition of older and newer snow. Hence, the age of different snow grains within each image is not necessarily the same, and some images can contain grains that have the characteristics of both old and new snow.

In order to reduce the computational cost of the learning algorithms, the images are rescaled from 2464×2056 to 492×410 pixels, transferring to a 80% reduction in image size. From **Fig. 4.11** one can see that this relatively large reduction does not impact the characteristics of the image, and as the snow conditions stay the same, the resizing should not have significant effect on the performance of the classification other than reducing its computational cost. Further, the same figure also shows the two augmentation techniques of including illuminated and rotated images. Again, the snow conditions do not change upon either one of these techniques, and they still represent the class label of the original image. An image classification model should be able to make classifications based on prominent characteristics and at the same time ignore insignificant changes. Hence, since the rotation and illumination of images do not alter the snow conditions, this should not negatively affect the performance of the classifiers, but rather lead to more robust models.

An accuracy of the SVM algorithm of 95.8%, corresponding to correct predictions of 92 out of 96 images, shows that the learning algorithm indeed has been able to learn the classification problem. The four misclassified images are shown in **Fig. 4.13**, and as previously mentioned they all originate from the same snow conditions. Although they are labeled as new snow, a comparison with the snow images in **Fig. 4.10** does not make it clear that these images belong to this category. The wrongly classified images contain both rounder grains characterized by old snow and more elongated grains characterized by new snow. Hence, the four images may place themselves near the previously discussed borderline of what is considered new and old snow, making them harder to classify correctly.

The CNN algorithm achieves an accuracy of 97.9%, corresponding to correct predictions of 94 out of 96 images. The two wrongly classified images seen in **Fig. 4.15** are also wrongly classified by the SVM algorithm. These two images have poorer quality, as seen in the top left corner of the non-rotated image, and may thus be more difficult to classify compared to the other two images misclassified by SVM, which the CNN classifies correctly. The training process of the CNN algorithm seen in **Fig. 4.16** shows only small differences for the model accuracy and loss between the test and training set. This indicates the the CNN model is able to generalize beyond training and is therefore a good fit for the data. From the plot of the model loss one can also see that the curves flatten out, indicating convergence of the loss function and thus sufficient training of the model.

Comparing the two learning algorithms it is clear that both of them enable separation of images into categories of new and old snow. The wrongly predicted images do all have characteristics of both old and new snow, making them hard to classify. The CNN achieves a slightly higher accuracy compared to the SVM, but the difference in computational time is huge. Where the SVM algorithm only need two minutes to train and predict the images, the CNN algorithm uses a total of 66 minutes. From the training process of the CNN algorithm in **Fig. 4.16** one can see that the accuracy on the test set does not increase significantly after the first few epochs. In other words, the model trains a large number of epochs without improving its accuracy. The reason so many epochs are included is to find where the loss function converges to the minimum. Although it does not manifest in the accuracy, this will optimize the model and make the CNN more confident in its classifications. Yet, the complexity of the problem is not large enough for the CNN to unleash its full potential. As mentioned in section 3.3, the CNN model have shown to scale better on larger, more complex datasets with respect to both accuracy and computational time. However, for the binary classification problem as presented here, the small improvement in the accuracy for the CNN does not make up for the much longer computation time compared to the SVM. As the metamorphism of snow leads to rapid transformation of snow grains, one wants to minimize the time between the capture and classification of images, making the SVM more suited on the presented classification problem.

Chapter 6

Conclusion

This report investigates how images of snow can be analysed to provide useful information in relevance to the performance of cross-country skis. The analysis is based on images from the GelSight imaging system, which provides an easy and effective way of capturing real-time snow conditions. Two independent methods for characterization and classification of snow images have been developed.

The presented characterization method involves using contours to calculate the OED and dendricty of snow grains. The validity of these results were tested by comparing the calculated OED of sandpaper to the estimated diameter from the FEPA standard. The calculated OED values of sandpaper do not fit with the given diameter values from the standard. As the density of grains becomes larger and grain size becomes smaller, the algorithm fails to separate one grain from another, leading to an systematic overestimation of the grain size. Hence, the parameter values from the contouring method should not be used as stand-alone results. However, the contouring method is consistent in its characterization as larger snow grains indeed give rise to higher OED compared to smaller grains. In addition, more dendritic grains give rise to higher values of dendricity compared to less dendritic grains. This consistency enables separation of different snow conditions through the OED and dendricity values.

The presented classification method involves using the learning algorithms SVM and CNN to classify snow images into categories of new and old snow. With an accuracy of 95.8% for the SVM algorithm and 97.9% for the CNN algorithm, they both enable separation of snow images based on the age of snow grains. The main source of error for the classification is new snow falling from the sky blending in with older snow on the ground, creating a mixed composition of new and old snow. This makes the images both harder to label and classify, resulting in false predictions by the two models. The shorter computation time of the SVM algorithm compared to the CNN algorithm, 2 versus 66 minutes for a training set of 276 images, makes the SVM algorithm better suited on the small dataset presented here.

The results of both the characterization and classification of snow images provide tools for separating different snow conditions. The contouring method is consistent as similar snow conditions give rise to similar values of OED and dendricity, and the learning algorithms achieve high accuracy on the classification of snow images. Thus, comparing snow images through their OED, dendricity and class label will give ski-technicians the opportunity to use previous experiences on similar snow conditions to prepare more optimal skis. Hence, the presented snow analysis can be used to provide useful information in relevance to the performance of cross-country skis.

Chapter 7

Future Work

The work presented in this report gives a way of separating different snow conditions using the OED, dendricity and age of the snow. This enables a broad separation of different snow conditions. However, as snow is a complex material with dependencies on many different factors, there is room for more narrow and accurate characterization of the snow's structure. For cross-country, the glide of skis is directly related to the deformation of both the snow surface and ski base at the contact spots. Evaluation of the hardness and density of snow could therefore benefit the characterization. These parameters can be measured directly as described in section 2.1.1 of this report. However, as the snow's structure can change rapidly, the measurements should be carried out in the field and just before the preparation of skis. Hence, there might be easier to include these parameters indirectly through measurements of air temperature, humidity, snow temperature etc. Combining the extracted parameters from the GelSight images with measured weather parameters can give a more thorough characterization of the present snow conditions.

In this report the classification problem was to distinguish between old and new snow. As seen from the snow images, there can be large variations within the two categories as well. This makes room for further separation of the snow conditions, and Fierz et al. (2009) operate with many different classes of snow grains. Here, the CNN algorithm should thrive as more classes makes the classification more complex, which better utilises the capabilities of the CNN. However, as there are different types snow grains existing within each snow image, it can be difficult give a single label for the whole image. Thus, instead of supervised learning algorithms like the CNN and SVM, unsupervised learning can prove beneficial. Here one can apply clustering to find similarities in the data, as showed in **Fig. 7.1**. The figure shows a connector graph representation on how a database of snow images can be created and exploited to find similar snow conditions. First, the GelSight instrument is used to acquire the snow images. Secondly, one deduces the snow parameters from the images and measures the weather parameters. After gathering a set of all these parameters, one can apply an unsupervised learning algorithm to cluster similar snow conditions based on the parameter values. In the end, one can use previous knowledge regarding ski preparation on similar conditions to prepare more optimal skis.

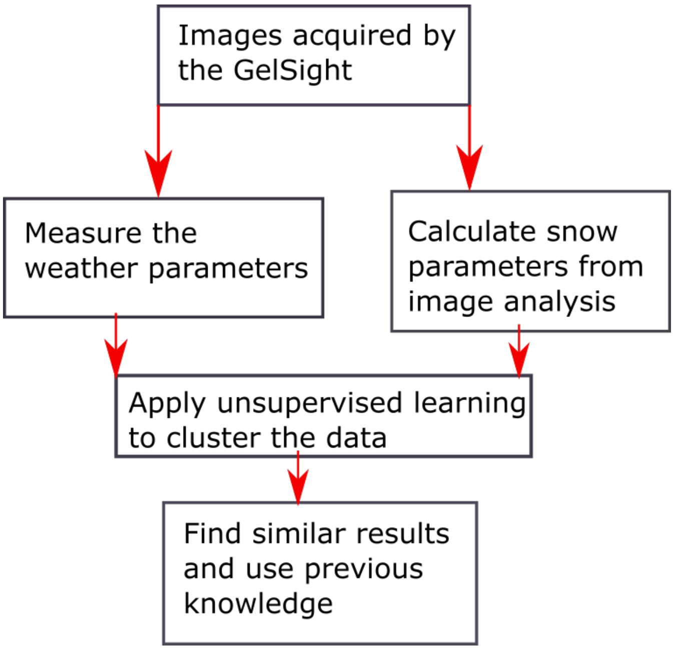

Images acquired by the GelSight
Measure the weather parameters
Calculate snow parameters from image analysis
Apply unsupervised learning to cluster the data
Find similar results and use previous knowledge

Figure 7.1: A connector graph representation on how a database of snow images can be created and exploited to find similar snow conditions.

Bibliography

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G.S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., Zheng, X., 2015. TensorFlow: Large-scale machine learning on heterogeneous systems. URL: `http://tensorflow.org/`. (Accessed 14.05.2020).

Adankon, M.M., Cheriet, M., 2009. Support Vector Machine. Springer US, Boston, MA. pp. 1303–1308. doi:`10.1007/978-0-387-73003-5_299`.

Asraf, H.M., Nooritawati, M., Rizam, M.S., 2012. A comparative study in kernel-based support vector machine of oil palm leaves nutrient disease. Procedia Engineering 41, 1353 – 1359. doi:`https://doi.org/10.1016/j.proeng.2012.07.321`.

Bader, H., Haefeli, R., Bucher, E., Neher, J., Eckel, ., Tharns, C., 1939. Der schnee und seine metamorphose. beitrage zur geologie der schweiz. schweizerische schnee- und lawinenforschungskomrnission. translation: 1954. snow and its metamorphism. SIPRE Translation 14 .

Bartlett, S.J., Rüedi, J.D., Craig, A., Fierz, C., 2008. Assessment of techniques for analyzing snow crystals in two dimensions. Annals of Glaciology 48, 103–112. doi:`10.3189/172756408784700752`.

Bradski, G., 2000. The OpenCV Library. Dr. Dobb's Journal of Software Tools URL: `http://docs.opencv.org/`. (Accessed 14.05.20).

Budde, R., Himes, A., 2017. High-resolution friction measurements of cross-country ski bases on snow. Sports Engineering 20, 299–311. doi:`10.1007/s12283-017-0230-5`.

Böttcher, R., Scherge, M., 2017. Ski preparation as a three-dimensional problem. Snowstorm Publishing - Gliding 1, 19–23.

Chollet, F., et al., 2015. Keras. URL: `https://keras.io`. (Accessed 14.05.2020).

Colbeck, S.C., 1996. A review of the friction of snow. Springer Netherlands, Dordrecht. pp. 275–291. doi:`10.1007/978-94-015-8705-1_18`.

Colbeck, S.C., 1998. Sintering in a dry snow cover. Journal of Applied Physics 84, 4585–4589. doi:`10.1063/1.368684`.

Cortes, C., Vapnik, V., 1995. Support-vector networks. Mach. Learn. 20, 273–297. doi:`10.1023/A:1022627411411`.

Denoth, A., 1989. Snow dielectric measurements. Advances in Space Research 9, 233 – 243. doi:`https://doi.org/10.1016/0273-1177(89)90491-2`.

Diale, M., Celik, T., Walt, C.V.D., 2019. Unsupervised feature learning for spam email filtering. Computers & Electrical Engineering 74, 89 – 104. doi:`https://doi.org/10.1016/j.compeleceng.2019.01.004`.

Dollar, P., Tu, Z., Tao, H., Belongie, S., 2007. Feature mining for image classification, pp. 1–8. doi:`10.1109/CVPR.2007.383046`.

Fierz, C., Armstrong, R., Durand, Y., Etchevers, P., Greene, E., Mcclung, D., Nishimura, K., Satyawali, P., Sokratov, S., 2009. The international classification for seasonal snow on the ground (UNESCO, IHP (International Hydrological Programme)–VII, Technical Documents in Hydrology, No 83; IACS (International Association of Cryospheric Sciences) contribution No 1).

GelSight, 2019. Gelsight. URL: `https://gelsight.com/`. (Accessed 14.05.2020).

Heggli, M., Köchle, B., Matzl, M., Pinzer, B., Riche, F., Steiner, S., Steinfeld, D., Schneebeli, M., 2011. Measuring snow in 3-d using x-ray tomography: assessment of visualization techniques. Annals of Glaciology 52, 231–236. doi:`10.3189/172756411797252202`.

Huang, S., Cai, N., Pacheco, P.P., Narrandes, S., Wang, Y., Xu, W., 2018. Applications of Support Vector Machine (SVM) learning in cancer genomics. Cancer genomics & proteomics 15, 41—51. doi:`10.21873/cgp.20063`.

Indolia, S., Goswami, A.K., Mishra, S., Asopa, P., 2018. Conceptual understanding of convolutional neural network- a deep learning approach. Procedia Computer Science 132, 679 – 688. doi:`https://doi.org/10.1016/j.procs.2018.05.069`.

Juszczak, P., Tax, D., Duin, R., 2002. Feature scaling in support vector data description .

Kaensar, C., 2013. A comparative study on handwriting digit recognition classifier using neural network, support vector machine and k-nearest neighbor, in: Meesad, P., Unger, H., Boonkrong, S. (Eds.), The 9th International Conference on Computing and Information Technology (IC2IT2013), Springer Berlin Heidelberg. pp. 155–163.

Kietzig, A.M., Hatzikiriakos, S., Englezos, P., 2009. Ice friction: The effects of surface roughness, structure, and hydrophobicity. Journal of Applied Physics 106, 024303 – 024303. doi:10.1063/1.3173346.

Kingma, D., Ba, J., 2014. Adam: A method for stochastic optimization. International Conference on Learning Representations .

Kragelsky, I., Demkin, N., 1960. Contact area of rough surfaces. Wear 3, 170 – 187. doi:https://doi.org/10.1016/0043-1648(60)90136-8.

Krol, Q., Löwe, H., 2016. Relating optical and microwave grain metrics of snow: The relevance of grain shape. The Cryosphere 10, 2847–2863. doi:10.5194/tc-10-2847-2016.

Le, H., Le, T., Tran, S., Tran, H., Thuy, N., 2012. Image classification using support vector machine and artificial neural network. International Journal of Information Technology and Computer Science 4. doi:10.5815/ijitcs.2012.05.05.

Lecun, Y., Bengio, Y., Hinton, G., 2015. Deep learning. Nature 521, 436–44. doi:10.1038/nature14539.

Lecun, Y., Bottou, L., Bengio, Y., Haffner, P., 1998. Gradient-based learning applied to document recognition. Proceedings of the IEEE 86, 2278 – 2324. doi:10.1109/5.726791.

Lesaffre, B., Pougatch, E., Martin, E., 1998. Objective determination of snow-grain characteristics from images. Annals of Glaciology 26, 112–118. doi:10.3189/1998AoG26-1-112-118.

Lundy, C.C., Edens, M.Q., Brown, R.L., 2002. Measurement of snow density and microstructure using computed tomography. Journal of Glaciology 48, 312–316. doi:10.3189/172756502781831485.

Lüthi, A., Fauve, M., Rhyner, H., Müller, E., 2018. Investigations of fundamental processes in ski-snow friction. Science and Skiing VII, 376–385.

Maeno, N., Arakawa, M., 2004. Adhesion shear theory of ice friction at low sliding velocities, combined with ice sintering. Journal of Applied Physics 95. doi:10.1063/1.1633654.

Mitchell, T.M., 1997. Machine Learning. McGraw-Hill, New York.

Painter, T., Molotch, N., Cassidy, M., Flanner, M., Steffen, K., 2007. Instruments and methods - contact spectroscopy for determination of stratigraphy of snow optical grain size. Journal of Glaciology 53, 121–127. doi:10.3189/172756507781833947.

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

Pielmeier, C., Schneebeli, M., 2002. Snow stratigraphy measured by snow hardness and compared to surface section images.

Pomeroy, J.W., Brun, E., 2001. Physical properties of snow.

Schölkopf, B., 2000. The kernel trick for distances. Advances in Neural Information Processing Systems 13, 301–307.

Sharma, N., Jain, V., Mishra, A., 2018. An analysis of convolutional neural networks for image classification. Procedia Computer Science 132, 377 – 384. doi:https://doi.org/10.1016/j.procs.2018.05.198.

Suzuki, S., Abe, K., 1985. Topological structural analysis of digitized binary images by border following. Computer Vision, Graphics, and Image Processing 30, 32 – 46. doi:https://doi.org/10.1016/0734-189X(85)90016-7.

Theile, T., Szabo, D., Luthi, A., Rhyner, H., Schneebeli, M., 2009. Mechanics of the ski–snow contact. Tribology Letters 36, 223–231. doi:10.1007/s11249-009-9476-9.

Tyagi, K.D., Bahl, R., Kumar, A., 2013. An overview of methods for snow stratigraphy studies, in: 2013 International Conference On Signal Processing And Communication (ICSC), pp. 230–235. doi:10.1109/ICSPCom.2013.6719788.

Weston, J., Watkins, C., 1999. Support Vector Machines for multi-class pattern recognition. Proceedings of the 7th European Symposium On Artificial Neural Networks , 219–224.

Appendix

A *Python* Scripts

The *Python* code used in this report is shown here. The code is also available in the 'Python scripts.zip' file submitted with this report.

Contouring and Calculation of Parameters

```
#Importing libraries
import numpy as np
import cv2
from matplotlib import pyplot as plt
from skimage import io, transform as tf
import os
from scipy.ndimage import rotate
import matplotlib.ticker as mtick

#Specifying locations
DIRECTORY=r'D:\Master images\Real snow'
FILENAME=r'D:\Master images\Real snow\Training\Falling snow\191121
                                      Beitostoelen -1 snowing 1900 sk - 002
                                      .png'

FILENAME_2=r'D:\Master images\Real snow\Testing\New Snow\191112
                                      NatrudstilenNew Scans scan009 image02
                                      .png'

#Choice lets the user choose whether to 1: plot original GelSight image
                                      together with binary image and
                                      contours,
#2: Calculate the OED and dendricity of all images in a database and #3
                                      plot two different snow images with
                                      calculations of OED and dendricity
                                      displayed
CHOICE='3'

#Image dimensions
IMAGEAREA=16.9*14.1
PIXELAREA=2464*2056

# Loads image and performs adaptive threshold
def loading_image(filename):
    image = cv2.imread(filename)
    #Preparing image by making it gray and setting threshold for black
                                          white
    imgray = cv2.cvtColor(image,cv2.COLOR_BGR2GRAY)
    thresh=cv2.adaptiveThreshold(imgray,255,cv2.ADAPTIVE_THRESH_MEAN_C,cv2
                                          .THRESH_BINARY,101,-3)
    return thresh,image

# Finds and draw contours and calculates area and perimeter of grains
def drawing_contours(thresh,image):
    contours, hierarchy = cv2.findContours(thresh.copy(), cv2.RETR_LIST,
                                          cv2.CHAIN_APPROX_SIMPLE)
    area_list=[]
    perimeter_list=[]
    count_contours=0
    for i in range(len(contours)):
        area = cv2.contourArea(contours[i]) # area
        if area > 150 and area < 250000:  # Try to remove all small
                                              contours and large contours (
                                              remove errors)
            area_list.append(area)
            contour_image=cv2.drawContours(image,contours[i],-1,(255,0,0),
                                                3)
            perimeter_list.append(cv2.arcLength(contours[i],True))
            count_contours+=1
    area_list=np.asarray(area_list)*IMAGEAREA/PIXELAREA
    perimeter_list=np.asarray(perimeter_list)*15.5/2260

    return contour_image, area_list, perimeter_list

#Plots GelSight image and contour/binary image or all three.
def plot__contour_image(raw_image,thresh_image,contour_image):
    plt.subplot(131),io.imshow(raw_image,extent=(0,16.9,0,14.1))
    plt.title('a) Original image')
    plt.xlabel('[mm]')
    plt.ylabel('[mm]')
    plt.xticks(np.arange(0, 17 + 1, 2))
    plt.yticks(np.arange(0, 14 + 1, 2))
    plt.subplot(132), io.imshow(thresh_image,extent=(0,16.9,0,14.1)) #Go
                                          back to Thresh
    plt.title('b) Binary image')
    plt.xticks(np.arange(0, 17 + 1, 2))
    plt.yticks([])
    plt.xlabel('[mm]')
    #Use the below code for calculating original, binary and contours side
                                          -by-side
    plt.subplot(133), io.imshow(contour_image, extent=(0, 16.9, 0, 14.1))
    plt.title('c) Contour image')
    plt.xticks(np.arange(0, 17 + 1, 2))
    plt.yticks([])
    plt.xlabel('[mm]')
    plt.subplots_adjust(wspace=0.05)#0.05
    #plt.savefig(r'C:\Users\Fredrik\Documents\Skole\Master\Data\Figures\
                                          Gelsight\whole_contouring_process
                                          ',bbox_inches='tight')
    plt.show()

#Plots different original images with either OED or dendricity displayed
def plot_raw_images():
    thresh1,raw_image1=loading_image(FILENAME)
    thresh2,raw_image2=loading_image(FILENAME_2)
    contour_image1, area1, perimeter1=drawing_contours(thresh1,raw_image1.
                                            copy())
    contour_image2, area2, perimeter2 = drawing_contours(thresh2,
                                            raw_image2.copy())
    equi_diameter1 = np.sqrt(4 * area1 / np.pi)
    equi_diameter2 = np.sqrt(4 * area2 / np.pi)
    dendricity1 = perimeter1 ** 2 / (area1*4*np.pi)
    dendricity2 = perimeter2 ** 2 / (area2*4*np.pi)
#HISTOGRAM
    fig = plt.figure()
    ax0=fig.add_subplot(232)
    plt.hist(equi_diameter1,bins=20,range=(0,1),rwidth=0.9, weights=np.
                                            ones(len(equi_diameter1)) / len(
                                            equi_diameter1))
    plt.title('a) OED')
    plt.ylim(ymin=0, ymax=0.25)
    ax0.yaxis.set_major_formatter(mtick.FuncFormatter(lambda y, _: '{:.0%}
                                            '.format(y)))
    plt.xlabel('[mm]')
    ax1 = fig.add_subplot(233)
    plt.title('a) Dendricity')
    plt.ylim(ymin=0,ymax=0.35)
    ax1.yaxis.set_major_formatter(mtick.FuncFormatter(lambda y, _: '{:.0%}
                                            '.format(y)))
    plt.hist(dendricity1, bins=20, range=(0, 30), rwidth=0.9,weights=np.
                                            ones(len(dendricity1)) / len(
                                            dendricity1))
    ax2 = fig.add_subplot(235)
    ax2.yaxis.set_major_formatter(mtick.FuncFormatter(lambda y, _: '{:.0%}
                                            '.format(y)))
    plt.hist(equi_diameter2, bins=20, range=(0, 1), rwidth=0.9,weights=np.
                                            ones(len(equi_diameter2)) / len(
                                            equi_diameter2))
    plt.xlabel('[mm]')
    plt.title('b) OED')
    plt.ylim(ymin=0,ymax=0.25)
    ax3 = fig.add_subplot(236)
    plt.title('b) Dendricity')
    plt.ylim(ymin=0, ymax=0.35)
    ax3.yaxis.set_major_formatter(mtick.FuncFormatter(lambda y, _: '{:.0%}
                                            '.format(y)))
    plt.hist(dendricity2, bins=20, range=(0,30), rwidth=0.9,weights=np.
                                            ones(len(dendricity2)) / len(
                                            dendricity2))
    #plt.show()

    plt.subplot(231),io.imshow(raw_image1,extent=(0,16.9,0,14.1))
    plt.xlabel('[mm]')
    plt.ylabel('[mm]')
    plt.xticks(np.arange(0, 17 + 1, 2))
    plt.title('a)  Dendricity='+str(round(np.average(dendricity1),2))+'
                                            and OED='+str(round(np.average(
                                            equi_diameter1),2)),fontsize=10)
    plt.subplot(234), io.imshow(raw_image2,extent=(0,16.9,0,14.1))
    plt.title('b)  Dendricity=' + str(round(np.average(dendricity2),2))+'
                                            and OED='+str(round(np.average(
                                            equi_diameter2),2)),fontsize=10)
    plt.xlabel('[mm]')
    plt.xticks(np.arange(0, 17 + 1, 2))
    plt.ylabel('[mm]')
    plt.subplots_adjust(wspace=0.05)
    plt.tight_layout()
    #plt.savefig(r'C:\Users\Fredrik\Documents\Skole\Master\Data\Figures\
                                                  Gelsight\Similar_parameters',
                                                  bbox_inches='tight')
    plt.show()


#Loads image, finds and draws contours, and calculates dendricity and OED
                                             for single GelSight image
def plot_single_image(FILENAME):
    thresh,image=loading_image(FILENAME)
    raw_image=image.copy()
    contour_image,area,perimeter=drawing_contours(thresh,image)
    dendricity=perimeter**2/(area*4*np.pi)
    equi_diameter = np.sqrt(4 * area/ np.pi)
    print('Equivalent diameter: ',np.average(equi_diameter))
    print('Dendricity: ',np.average(dendricity))
    print(np.average(np.sqrt(4*area/np.pi)))
    plot__contour_image(raw_image,thresh,contour_image)

def calculate_parameters(): #Calculates the OED and dendricity for images
                                             in the directory.
    f=open(r'D:\Master images\Sandpapir\test_lek.csv','w',)
    f.write('image;\t''OED;\t''Dendricity\n')
    for subdir, dirs, files in os.walk(DIRECTORY):
        for filename in files:
            if filename.endswith(".png"):
                path=os.path.join(subdir, filename)
                print(path)
                thresh, image = loading_image(path)
                contour_image, area, perimeter = drawing_contours(thresh,
                                                     image)
                dendricity = perimeter ** 2 / (area*4*np.pi)
                equi_diameter = np.sqrt(4 * area / np.pi)
                f.write(str(path)+';'+str(np.average(equi_diameter))+';'+
                                                     str(np.average(
                                                     dendricity))+'\n')


            else:
                print('ERROR: filename does not end with .png')
    f.close()


def main():
    if CHOICE=='1':
        plot_single_image(FILENAME)

    elif CHOICE=='2':
        calculate_parameters()
    elif CHOICE=='3':
        plot_raw_images()

main()
```

Preprocessing Data for Machine Learning

```
#Importing libraries
import numpy as np
import os
import cv2
from scipy.ndimage import rotate
WIDTH=492
HEIGHT=410


DATAPATH=r'D:\Master images\Real snow'

def preprocess_images(): #Function returns thresholded images and labels
                                         to the images
    X_train=[]
    X_test=[]
    original_images=[]
    y_train=[]
    y_test=[]
    for subdir, dirs, files in os.walk(DATAPATH):
        for filename in files:
            path = os.path.join(subdir, filename)
            if '.png' in path:
                image = cv2.imread(path)
                original_images.append(image.reshape(2464 * 2056 * 3))
                imgray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
                dim = (WIDTH, HEIGHT)
                imgray = cv2.resize(imgray, dim, interpolation=cv2.
                                                     INTER_AREA)
                random_degree=90
                if 'Old' in path:
                    if 'Training' in path:
                        y_train.append(1)
                        X_train.append(imgray.reshape(WIDTH * HEIGHT))
                        y_train.append(1)
                        r_image=rotate(imgray,random_degree)
                        X_train.append(r_image.reshape(WIDTH*HEIGHT))
                    else:
                        y_test.append(1)
                        X_test.append(imgray.reshape(WIDTH * HEIGHT))
                        y_test.append(1)
                        r_image=rotate(imgray,random_degree)
                        X_test.append(r_image.reshape(WIDTH * HEIGHT))
                else:

                    if 'Training' in path:
                        y_train.append(0)
                        X_train.append(imgray.reshape(WIDTH * HEIGHT))
                        y_train.append(0)
                        r_image=rotate(imgray,random_degree)
                        X_train.append(r_image.reshape(WIDTH * HEIGHT))
                    else:
                        y_test.append(0)
                        X_test.append(imgray.reshape(WIDTH * HEIGHT))
                        y_test.append((0))
                        r_image=rotate(imgray,random_degree)
                        X_test.append(r_image.reshape(WIDTH * HEIGHT))
    return X_train,X_test, y_train, y_test, original_images


def prepare_dataset():
    X_train, X_test, y_train, y_test, original_images=preprocess_images()
    X_test_unscaled=X_test
    # Feature Scaling
    X_train = (X_train - np.mean(X_train)) / np.std(X_train)
    X_test = (X_test - np.mean(X_test)) / np.std(X_test)
    return X_train ,X_test,y_train,y_test, original_images,
                                        X_test_unscaled
```

Support Vector Machine

```
#Importing libraries
from sklearn.metrics import accuracy_score, confusion_matrix
from sklearn.svm import SVC
import Preprocess_and_prepare as prepare
from matplotlib import pyplot as plt
from skimage import io
import numpy as np

def SVM(X_train, X_test, y_train, y_test):
    clf = SVC(kernel='rbf')  # RBF
    clf = clf.fit(X_train, y_train)
    predictions = clf.predict(X_test)
    accuracy = accuracy_score(y_test, predictions)
    print('SVM: ', accuracy)
    print(y_test)
    print('##########')
    print(predictions)
    return accuracy, predictions

def main():
    accuracy_list=[]
    X_train, X_test, y_train, y_test, original_images_test,
                                        X_test_unscaled = prepare.
                                        prepare_dataset()
    accuracy, predictions = SVM(X_train, X_test, y_train, y_test)
    accuracy_list.append(accuracy)
    #print(accuracy_list)
    print(confusion_matrix(y_test,predictions))
    print("Accuracy: ",np.round(np.average(accuracy_list),3))
    print("Confusion Matrix:")
    print(confusion_matrix(y_test, predictions))
    fig=plt.figure()
    counter=0
    for i in range(len(y_test)):
        if y_test[i]!=predictions[i]:
            print(i)
            fig.add_subplot(221+counter)
            if counter%2==0:
                io.imshow(X_test_unscaled[i].reshape(410,492))
                plt.title('Prediction: Old '+'/ True value: New',fontsize=
                                                    12)
            else:
                io.imshow(X_test_unscaled[i].reshape(492, 410))
                plt.title('Prediction: Old ' + '/ True value: New',
                                                     fontsize=12)
            counter+=1

    plt.show()

if __name__=='__main__':
    main()
```

Convolutional Neural Network

```
#Importing libraries
from keras.models import Sequential
from keras.layers import Dense, Conv2D, Flatten, MaxPooling2D, Dropout
from keras.optimizers import Adam
from sklearn.metrics import confusion_matrix
import Preprocess_and_prepare as prepare
from keras.utils import to_categorical
import numpy as np
from matplotlib import pyplot as plt
from timeit import default_timer as timer
from skimage import io


def ConvNeuralNetwork(inputShape,n_classes,lerning_rate,filter_value,
                                    depth_value):
    model = Sequential(name='CNN')
    model.add(Conv2D (filters=filter_value, kernel_size=(3,3), input_shape
                                        =(prepare.HEIGHT,prepare.WIDTH,1)
                                        , activation='elu',padding='same'
                                        ))
    model.add(MaxPooling2D(pool_size=(2, 2)))
    model.add(Conv2D (filters=filter_value*2, kernel_size=(3,3),
                                        activation='elu', padding='same')
                                        )
    model.add(MaxPooling2D(pool_size=(2,2)))
    model.add(Flatten())
    model.add(Dense (depth_value*2, activation='relu'))
    model.add(Dense (depth_value, activation='relu'))
    model.add(Dropout(0.5))
    model.add(Dense(n_classes, activation = 'softmax'))
    model.compile(loss='categorical_crossentropy',metrics=['
                                        categorical_accuracy'], optimizer
                                        =Adam(lr=lerning_rate))
    return model

def trainModel(model,X_train,y_train,X_test,y_test,batchsize):
    result = model.fit(X_train, y_train, batch_size=batchsize, epochs=75,
                                        verbose=2,validation_data=(X_test
                                        ,y_test))
    return result

def calcAccuracy(predictions, targets):
    print('CalcAccuracy')
    correct=0
    actual=0
    predictions_winner=[]
    targets_winner=[]
    for i in range(0,len(predictions)):
        if np.argmax(predictions[i])==np.argmax(targets[i]):
            correct+=1
        else:
            print(i)
        if np.argmax(predictions[i])==0:
            predictions_winner.append(0)
        else:
            predictions_winner.append(1)
        if np.argmax(targets[i])==0:
            targets_winner.append(0)
        else:
            targets_winner.append(1)
    print(np.around(predictions,3))
    return correct/len(predictions),predictions_winner,targets_winner

def main():
    start=timer()
    learning_rate=[0.000001]
    batchsize=[10]
    counter=0
    f=10 #Filters
    d=15 # Depth
    for lr in learning_rate:
        for b in batchsize:
                X_train, X_test, y_train, y_test, original_images,
                                                     X_test_unscaled=
                                                     prepare.
                                                     prepare_dataset()
                X_train=X_train.reshape(-1,prepare.HEIGHT,prepare.WIDTH,1)
                X_test =X_test.reshape(-1, prepare.HEIGHT,prepare.WIDTH, 1
                                                     )

                y_train = to_categorical(y_train)
                y_test = to_categorical(y_test)
                learning_model = ConvNeuralNetwork((X_train.shape),2,lr,f,
                                                     d)
                history = trainModel(learning_model, X_train, y_train,
                                                     X_test,y_test,b)
                preds = learning_model.predict(X_test)
                accuracy,prediction_winner,targets_winner = calcAccuracy(
                                                     preds, y_test)
                print('Accuracy:',np.round(accuracy,3))
                print('Confusion Matrix:')
                print(confusion_matrix(targets_winner, prediction_winner))
                #Accuracy
                counter+=1
                plt.subplot(1,2,counter)
                plt.plot(history.history['categorical_accuracy'])
                plt.plot(history.history['val_categorical_accuracy'])
                plt.title('a) Model accuracy')
                plt.ylabel('Accuracy')
                plt.xlabel('Epoch')
                plt.legend(['Train', 'Test'], loc='lower right')
                #Loss
                plt.subplot(1,2,counter)
                plt.plot(history.history['loss'])
                plt.plot(history.history['val_loss'])
                plt.title('b) Model loss')
                plt.ylabel('Loss')
                plt.xlabel('Epoch')
                plt.legend(['Train', 'Test'], loc='upper right')
    plt.show()
    end=timer()
    print('Elapsed time of program is: ',(end-start)/3600,' hours')
    counter=1
    fig=plt.figure()
    for i in range(len(preds)):
        if prediction_winner[i]!=targets_winner[i]:
            ax=fig.add_subplot(220+counter)
            if counter in [1,3]:
                io.imshow(X_test_unscaled[i].reshape(410,492))
            else:
                io.imshow(X_test_unscaled[i].reshape(492, 410))

            counter+=1
            ax.title.set_text('Prediction: Old ' + '/ True value: New')
    fig.tight_layout()
    plt.show()

if __name__=='__main__':
    main()
```