Norwegian University
of Life Sciences

Master's Thesis 2022 30 ECTS
Faculty of Science and Technology

Semantic Segmentation of Roof Materials in Urban Environment by Utilizing Hyperspectral and LiDAR Data

Stian Teien
Data Science

Acknowledgement

This thesis rounds off my five-year master's degree in data science at Norwegian University of Life Sciences (NMBU). I am grateful for the opportunity to write a master's thesis in an interesting, innovative and technological field of science.

A special acknowledgment to my supervisors for close guidance, showing interest and inspiration. Thanks to my main supervisor, Prof. Ingunn Burud, for introducing me to the field of image analysis and hyperspectral imagery. Her ability to explain and perceive the bigger picture helped me set a course for my master's thesis. Thanks to my co-supervisor, Assoc. Prof. Kristian Hovde Liland, for interesting conversations, valuable input and technical help. Thanks to my second co-supervisor, PhD(c) Agnieszka Kuras, for guidance and help with the data set. Her insight and knowledge have been precious for this thesis.

I would like to thank the research project for the opportunity to contribute to their work.

Thanks to my fellow master students for enjoyable conversations, coffee breaks and guidance in geomatics wisdom, where my knowledge was limited.

Finally, I would like to thank my family and my girlfriend for their support and constant motivation during my years of study. They have been the foundation of this achievement, and I am grateful for their influence.

Ås, May, 2022
Stian Teien

Abstract

Mapping areas in an urban environment can be challenging due to various materials and manufactured structures. The urban environment is a mix of natural and artificial materials, and finding the right object of a specific material is a challenge even for the trained eye. Therefore, by applying high spectral resolution hyperspectral imagery it is possible to examine surface materials based on spectral signature. Combined with LiDAR, it is also feasible to detect the geometrical structure of the surface. These data can be exposed to a machine learning algorithm to recognize objects automatically. In this study machine learning algorithms are exposed to airborne images of roof materials.

This thesis presents an application of semantic segmentation for roof materials based on fused hyperspectral (HySpex VNIR-1800 and SWIR-384) and LiDAR (Riegl VQ-560i) data acquired from 2021 over Bærum municipality near Oslo in Norway. The machine learning algorithm is a semantic segmentation model named Res-U-net with a U-net architecture and a ResNet34 backbone. The Res-U-Net is a supervised neural network with high capacity to learn high-dimensional airborne data. The model returns a mask of the urban area that pinpoints the roofs' position and materials. The ground truth is generated with information from field work, a geographical database and the watershed algorithm for object detection. This ground truth consists of nine different roof materials and background.

The semantic segmentation model is optimized by testing different model configurations for this specific problem. The best model scores 0.903, 0.896, and 0.579 in accuracy score, F1 score weighted and Matthews Correlation Coefficient. For the binary problem of detecting roof the model scores 0.948, 0.946, and 0.767 on the same metrics. This study demonstrates that semantic segmentation is viable for localizing and classifying roof materials with fused hyperspectral and LiDAR data. Such an analysis can potentially automate several mapping chores and manual assignments by systemically processing a larger area in a short time to free human capacity.

Table of Contents

List of Abbreviations and Acronyms

ANN	Artificial Neural Network
CE	Categorical CrossEntropy
CI	Confidence Interval
CNN	Convolutional Neural Network
CPU	Central Processing Unit
DSM	Digital Surface Model
DTM	Digital Terrain Model
FKB	Felles KartDatabase
FL	Focal Loss
FN	False Negative
FOV	Field Of View
FP	False Positive
GIS	Geographic Information System
GNSS	Global Navigation Satellite System
GPU	Graphics Processing Unit
HS	Hyperspectral (image)
IMU	Inertial Measurement Unit
LiDAR	Light Detection And Ranging
MCC	Matthews Correlation Coefficient
MLP	Multilayer Perceptron
nDSM	Normalized Digital Surface Model
NEO	Norsk Elektro Optikk
nm	Nanometer
PCA	Principal Component Analysis
QTM	Quick Terrain Modeler
ReLU	Rectified Linear Unit
RGB	Red, Green, Blue (image)
SWIR	Short wave infrared
TN	True Negative
TP	True Positive
VIS	Visible Light Range
VNIR	Near Infrared

List of Figures

List of Tables

1 Introduction

1.1 Background and Purpose of Study

Urban environments consist of a mix of natural and artificial surface materials that reflect and influence the energy consummation, climate, and ecological condition of cities and suburbs [1]. These surfaces can be materials ranging from wood, concrete, tiles, ceramic, metal, sand, stone and plastic. One of the main components in urban environment is buildings, manufactured objects built in various materials directly exposed to nature and precipitation. The variety of building materials can dramatically change the characteristics of a building. Therefore, it can be beneficial for authorities and businesses to map areas with roof buildings that can be used in various amounts of analysis, reports, maintenance or conventional chores.

There are several ways of mapping the buildings' roofs. For instance, the owner of the building can report it, someone can collect the information manually or use a sensor far away to photograph the roofs. The last, remote sensing, is by far the most effective for covering a larger area in a short amount of time. Converting this remotely sensed information is challenging due to the complexity, spatial and spectral diversity in the urban area [2, 3].

Hyperspectral imagery has high spectral resolution and can be utilized in finding spectral differences in materials. It has high potential in material-oriented mapping by recognition spectral characteristics in surface materials [1]. Combining this with a laser rangefinder that measures the geometry of objects can lead to both spectral and spatial information of an area. The challenge still stands to find valuable information in this dense fog of data.

Machine learning has the later years excelled in image analysis [4]. By exposing data to particular kinds of machine learning algorithms, valuable information can be derived from the dense fog. In previous years different techniques have been used to extract such information from hyperspectral images and a combination of hyperspectral and laser data. In 2005 Lemp and Weider [5] used hyperspectral and laser scanning data to categorize roof surfaces in segments. They manually extracted the roof with a digital surface model and a vegetation filter. These roofs were manually differentiated based on the spectral information. In 2007 S. van der Linden et al. [6] used a support vector machine to classify different objects in the urban environment. Fast forward to 2019, A. Rangnekar et al. [7] introduced a data set on airborne hyperspectral images. They used several convolutional neural networks to localize and classify different objects in that data set. Talented scientists, like F. Trevisiol et al., [8] have classified roof materials from high spectral resolution satellite images, but did prework to isolate the roof using laser data before classi-

fication. In 2020 R. Senchuri [9] wrote the predecessor of this master's thesis. He analyzed and compared shallow machine learning to recognize objects in the urban environment. This master's thesis will take his work to the next step and use deep learning semantic segmentation in the same location. R. Senchuri's work was mainly focused on pixel-based classification that often results in a salt-pepper effect of different classes in an area. I want to eliminate that issue of mixed pixels, and consider the roofs like cohesion objects.

Using neural network model architectures from image segmentation and exposing these models to a fused hyperspectral and laser data set makes it interesting to see the possibility of localizing and classifying roofs. This is done by feeding the network the raw data without the aid of object extraction before classification. The neural networks job is to find and recognize the roof materials all by itself. The thesis objectives can be summarized in the forms of two research questions:

- Can semantic segmentation models detect roofs in urban environments with fused hyperspectral images and LiDAR data?
- Is semantic segmentation viable for localizing and classifying roof materials in urban environments by utilizing fused hyperspectral images and LiDAR data?

1.2 Thesis Structure

This thesis is divided into six chapters. It begins with background and purpose for the study. Chapter two introduces the theoretical background. Followed by the third chapter that explains the method of data processing and model optimization. The fourth chapter explores the results, and the fifth chapter is a discussion about data quality, model performance, comparison to similar work and future suggestions. The thesis finishes with a conclusion. The appendix contains optimization, experiments and results that have a minor impact on the outcome of the thesis. The source code is a Github repository `https://github.com/stianteien/M_DV_V2022`, where the last commit for the thesis is *f73b658*. Additionally, two demonstration jupyter notebooks in Table 1.1 show the processes of generating the ground truth and semantic segmentation. These two processes are the essential part of this master's thesis, and they give practical examples of how the research questions are solved.

Table 1.1: Demonstrative jupyter notebooks for essential parts of generating the ground truth and training a semantic segmentation model.

Filename	**Link**	**Commit hash**
demo_generate_ground_truth.ipynb	Github-Link	5900512
demo_semantic_segmantion.ipynb	Github-Link	f73b658

2 Theory

2.1 Remote Sensing

The main objective of remote sensing is to gather information at a distance. In airborne remote sensing, the Earth's land and water surfaces are generally the point of interest. These surfaces get observed for the purpose of measuring the reflected or emitted electromagnetic energy with sensors [10].

The history of remote sensing origins from the early practice and technology. In 1839 Loius Daguerre (1789-1851) reported the results of his experiment with photographic chemicals. This date forms the milestone of the very birth of successful photography. Later on in 1858, Gaspard-Félix Tournachon (1829-1910) took the first airborne photo from a tethered air balloon in France. In the following years, airplanes were the primary platform for airborne images. World War I and II excelled the necessity of aerial reconnaissance. As a result of the increasing demand for remote sensing the momentum in technological innovation sped up, and resulted in technologies and equipment such as radars, weather satellites and digital image processing. In the 1980s and 1990s hyperspectral sensors, LiDAR (light detection and ranging) and global remote sensing systems were developed. Nowadays, remote sensing is used in various applications by the military [11], governments [12], agriculture [13], environmental monitoring [14], scientist [15] and even as a hobby [16].

2.1.1 Electromagnetic Radiation

Electromagnetic radiation is electromagnetic waves traveling through space carrying electromagnetic radiant energy [17]. Electromagnetic waves consist of an electric and a magnetic field. The waves' surface normal vectors are oriented orthogonal to one another and travel as vectors in the same direction. The length of the waves tells the energy of a wave and formulates as

$$Energy = \frac{hc}{\lambda}, \tag{1}$$

where h is Planck's constant ($6.62607015 \cdot 10^{34} Js$), c is the speed of light ($2.99792458 \cdot 10^8 \frac{m}{s}$) and λ is the wavelength. Wavelength is the mean distance between wave crests and is measured in nano- (nm) or micrometers (μm). Figure 2.1 illustrates an electromagnetic wave with the wavelength λ, electric and magnetic field.

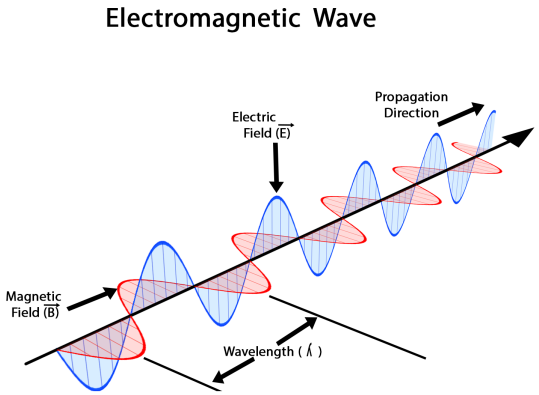


Figure 2.1: Electromagnetic waves consist of electric and magnetic fields perpendicular to one another and the direction of propagating. λ is the wavelength between wave crests. Figure used under a CC BY-SA 4.0 International license.

Electromagnetic radiation is categorized by its wavelength, and is called the electromagnetic spectrum that ranges from low frequency radio waves to high frequency X-rays and gamma waves. Humans can see some parts of the electromagnetic spectrum, and this is what we call visible light. Visible light contains electromagnetic radiation with wavelengths from 400nm to 750nm, and in this range, every visible color appears. In the field of remote sensing, electromagnetic waves are grouped based on their wavelengths. For instance, the lower frequency waves from 400nm to 1000nm are grouped as visible and near infrared (VNIR), waves in the range of 1000nm to 2500nm are called short-wavelength infrared (SWIR), mid and long wavelength infrared are waves with λ between 3000nm to 15000nm. The last group is far infrared and contains wavelength from 15000nm to 1,000,000nm [18].

2.1.2 Electromagnetic Radiation Interference with Surfaces

When electromagnetic radiation emits from the sun or an external light source, it travels through space and interacts with particles and gases in the atmosphere. This causes the radiation to scatter, reflect, transmit and to be absorbed by the elements and particles. The radiation that gets reflected by the surface travels through the atmosphere once again and then hits the sensor that records the energy. Figure 2.2 illustrates this interaction.

When the transmitted radiance reaches the surface of the earth and interacts with surfaces, the radiance is reflected, absorbed and transmitted based on the energy and surface material. For instance, Vegetation absorbs primarily red wavelengths and reflects the green, and we observe the vegetation as green. The material surfaces and density have a great impact on electromagnetic radiation.

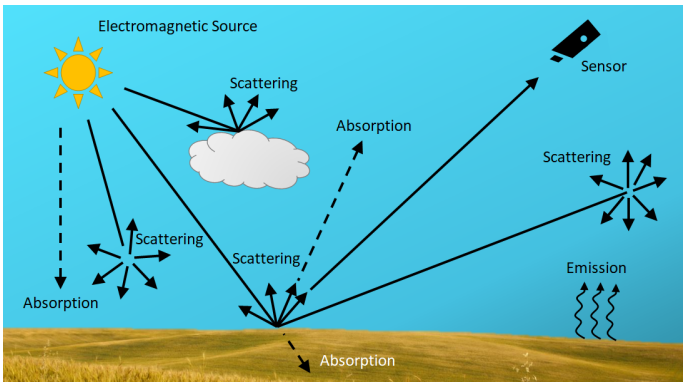


Figure 2.2: Electromagnetic radiation interference with different kinds of obstacles in their paths.

2.1.3 Radiance and Reflectance

Radiance is the value of reflected light in each wavelength [19]. Radiance includes all radiation reflected from the surface, including interfering reflection from nearby objects. Radiance is derived from what the sensor detects. The rawest data is a digital number that corresponds to the intensity of the electromagnetic radiation for the recorded wavelength. These digital numbers are converted to radiance based on the physical properties of sensor. Even though the values are transformed with respect to the sensor, the data still contains a lot of noise from the atmosphere. Radiance is heavily influenced by the condition of the atmosphere and light. The atmosphere and light condition are constantly changing, and the data needs to be corrected to get stable repetitive values before an analysis.

Reflectance is the ratio between reflected and incident radiation as a function of wavelength [20]. In other words, how much of the light gets reflected. Radiance consists of atmospheric effects, and estimating the reflectance spectrum from the radiance spectrum is a crucial step in aerial image analysis applications. To obtain a reflectance spectrum, an atmospheric correction needs to be performed [21].

2.1.4 Corrections

Atmospheric correction is a process for removing the atmospheric effect of the radiance values from images taken by satellite or airborne sensors [22]. The correction processes radiance data to reflectance data. Essential factors when correcting are aerosol thickness, water vapor, airplane altitude, the terrain, time and sun angle.

Ortorectification is one correction in the process of making ortophoto [23]. Ortophotos are aerial photos seen directly above. When capturing aerial photos the photos are often taken from an angle. This angle causes the photo's perspective to see the object slightly from the side. Therefore, images over the same area are processed together to create results that look like they are captured directly above

the areas. Figure 2.3 illustrates the angle effect.

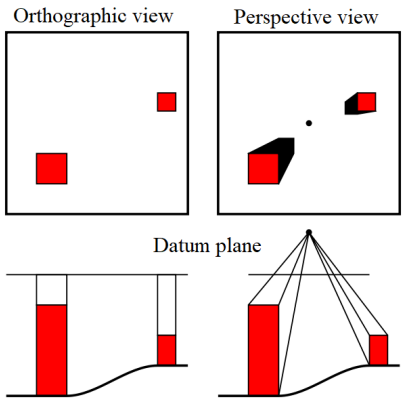


Figure 2.3: Illustration of how angle from an airborne image can affect the photo. Ortorectification is used to correct the photo. Illustration under CC0 1.0 licence.

2.1.5 Georeferencing

X. A. Yao, author of the book *Georeferencing and Geocoding* [24], explains georeferencing as the process of giving the internal coordinate system of a map or aerial scan relative to a geographic coordinate system. It is one of the fundamentals of geographic information system (GIS). Every location on the earth's surface can be specified by a set of values in a coordinate system [24]. When georeferencing aerial photos, the operators use the position of the airplane, altitude, airspeed and known locations on the ground. In combination with a Global Navigation Satellite System (GNSS) they pinpoint the exact location of the photographed area.

2.1.6 Capture Methods

In aerial photography there are two major scan methods: push broom and whisk broom. Push broom uses a line, often called line-scan, of detectors that are perpendicular to the flight direction and ground [25]. Each detector focuses on their points of the scan line. This method gives a robust, instantaneous and clear signal from the scanned area, but requires calibration on all detectors. Whisk broom scanners move back and forth like a whisk broom, and capture a whole strip at once [26]. Figure 2.4 illustrates a flight where both methods are used. Field of view (FOV) explains the width of the capture angle.

Remote sensing sensors are categorized into two types: passive remote sensing sensors and active remote sensing sensors. Passive remote sensing, such as hyperspectral images, requires radiation to be emitted or reflected from a surface. This means that an external source of electromagnetic radiation, like the sun, must be emitting radiation while using a passive remote sensor. Passive remote sensors are

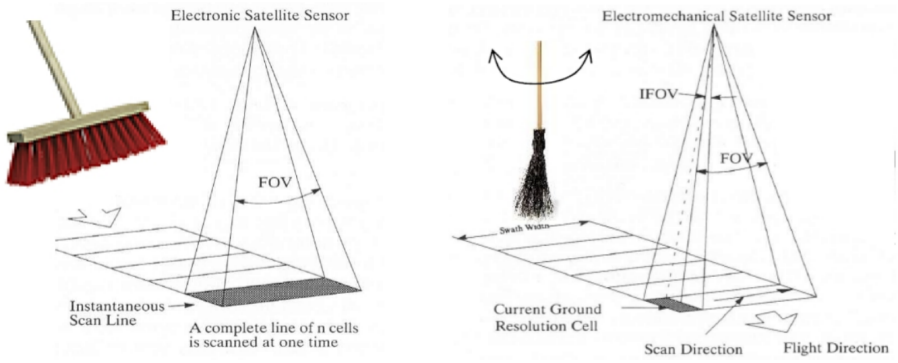


Figure 2.4: Illustration of push broom to the left and whisk broom methods to the right in aerial imagery. FOV explains field of view, the width of the capture angle. Illustration modified from [27].

sensitive to atmospheric conditions and illumination. Active remote sensors, like LiDAR, emit radiation like a laser beam, and are less responsive to external factors [28]. An active sensor does not need an external light source, and can for instance use the intensity measurement of the beam to map shaded areas in urban environments [29]. The sensors are complementary and are often used in combination.

2.1.7 Felles Kartdatabase

Felles KartdataBase (FKB) is a central part of the collection in the Norwegian base map [30]. It is fundamental geographical information to enact statutory and regulatory causes. The maps can be used in projects, administrative processes, analysis or creating new maps. The database is owned by the "Geovekst"-collaboration [1] in close association with the responsible municipalities. FKB contains information about area resources, railways, buildings, building projects, contours, cables, airports, nature information, tracks, water and roads. FKB accommodates such an amount of information, and gets used regularly in governance, the information gets updated regularly.

2.2 Hyperspectral Imaging

Hyperspectral imaging is a technique that combines conventional images with optical spectroscopy [31]. The images have the advantage of combining spatial information of an area, and spectral data from a wide range of electromagnetic wavelengths. Hyperspectral images originate from remote sensing and have been used in various

[1] A collaboration between The Norwegian Mapping Authority, municipalities, Norwegian Public Roads Administration, the county municipalities, Energy Norway, The Norwegian Ministry of Agriculture and Food, Bane NOR, Telenor and The Norwegian Water Resources and Energy Directorate

applications like vegetation and water resources control [32, 33], food safety and control [34] and biomedicine [35]. NASA has through the years been one of the leading developer of the hyperspectral cameras for mapping the earth while orbiting it [36].

A hyperspectral image can be seen as a cube. The first two dimensions are the spatial dimensions representing the shape of the hyperspectral image, while the third, spectral dimension represents the number of spectral bands [37]. These bands contain optical information recorded at an exact wavelength. The recordings are obtained so each pixel in the image has an approximate continuous range of spectral information [38]. An example of a hyperspectral cube is shown in Figure 2.5. This cube has spatial dimensions x and y. Y is obtained with the photographically technique push broom scan. The spectral axis λ contains optical information from a range of wavelengths. In this case the bands span from 400nm to 700nm called the visible light range (VIS). A hyperspectral cube can give a vast amount of information about objects with the same spectral capability as conventional images.

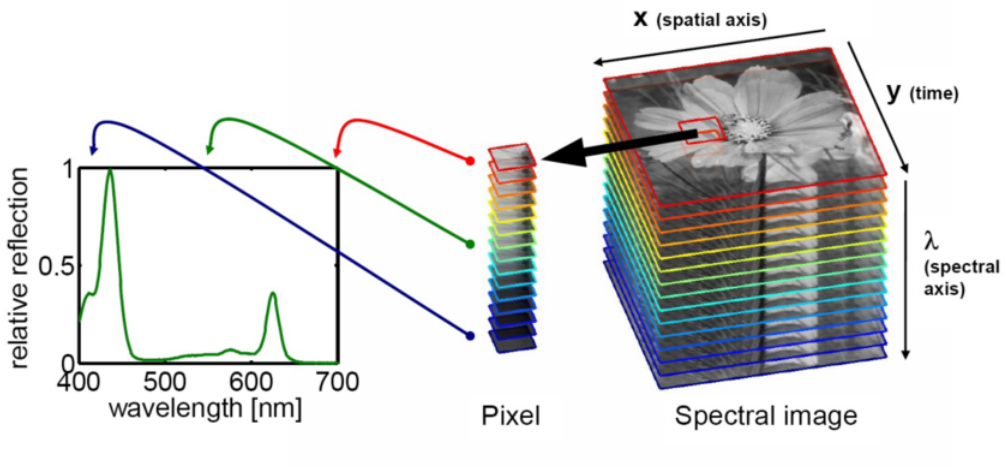


Figure 2.5: An example of a hyperspectral cube. X and Y are the spatial dimensions and λ is the spectral axis. One pixel is extracted and the spectral information is visualised. In this example, the bands cover the spectrum of visible light [39]. Illustration under CC BY 4.0 International licence and personal approval by the author.

2.2.1 Unfolding a Hyperspectral Cube

The hyperspectral cube has three dimensions, but most algorithms only have a two dimensional input. Therefore it is beneficial to unfold the cube before applying statistical modeling. The most common approach to unfolding the cube is to use the pixels as data points and the spectral information as features. This will reduce the dimensions of the data from three to two dimensional, and in the process the

spatial information is lost. However, this shape is favorable for exploiting spectral information and variance from all the pixels. In Figure 2.6, a hyperspectral cube is unfolded. The wavelengths in the spectral dimension now become the features dimension of a table-like data matrix.

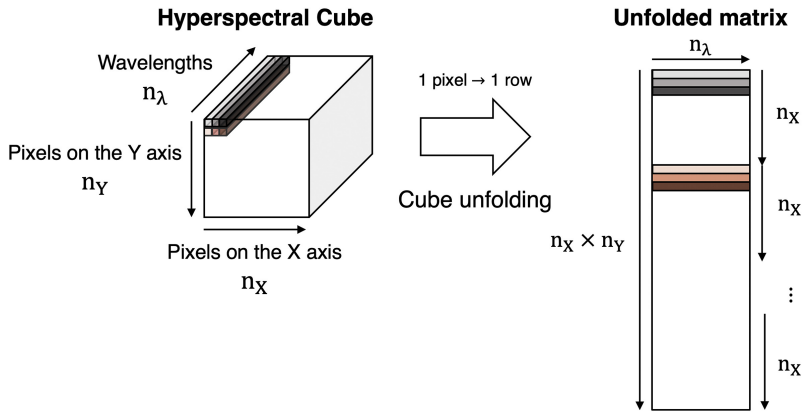


Figure 2.6: Hyperspectral cube unfolded. The cube is transformed into a two dimensional matrix. Reprinted from [40] with permission from Elsevier.

2.2.2 Spectral Signature

Spatial resolution from airborne remote sensing is often so coarse it is difficult to identify objects. Therefore, it is interesting to inspect the spectral signatures of the surface of materials. The spectral signature of an object can be defined in the solar-reflective region by its reflectance as a function of wavelength, measured at an appropriate spectral resolution [10]. Physical properties and chemical composition determinate absorption and reflection for a material [41]. These characteristic properties give the basis for unique spectral signatures for different materials. Conventional images contain three wavelengths: red, blue and green. The spectral signature of these three wavelengths is often not enough to identify objects. This motivates that hyperspectral images can distinguish materials based on differences in their spectral signatures. In Figure 2.7 are some typical spectral signatures of land cover types. The reflectance of a material can vary in strength reliant on the incoming solar reflections. Therefore, the spectral signature is often the same shape independent of the light conditions. For instance in Figure 2.7, green vegetation will have a similar shape in different light settings, but the reflectance amplitude may vary. Even though there is much information to extract from the spectral signature, the spatial resolution still limits some of the analysis. For instance, if an urban area with a car is scanned and one pixel shows the whole car, it is challenging to see that pixel as a car. Using the spectral signature for the metal of the car, it is possible to assume that the pixel is an automobile.

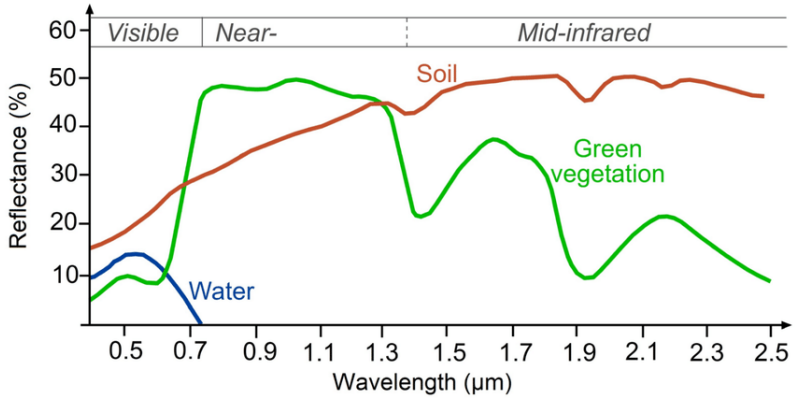


Figure 2.7: Typical spectral signatures of different land covers in visible and infrared region of the electromagnetic spectrum. Illustration with modification from [42].

2.3 LiDAR

Light Detection and Ranging (LiDAR) is an optical sensor used to measure points of physical objects [43]. LiDAR is an active sensor that uses a laser beam aimed at an object, then with a very precise measurement counts the time of the returning laser beam. Thus, it calculates the distance to the object. LiDAR is both fast and accurate, therefore it is suitable for mapping larger complex areas. LiDAR is a viable solution for ongoing tasks for mapping infrastructure and terrain for modelling, preliminary maintenance and analysis [44]. LiDAR scans often produce a point cloud, where each point is the distance from the device. In Figure 2.8 is a typical point cloud from a LiDAR scan. The density of the clouds varies from type of scanner, distance from the scanner and texture of an object. Scans with a high density of points enable objects to be recognized easier. In airborne LiDAR scans variation of point density from scans is minimal.

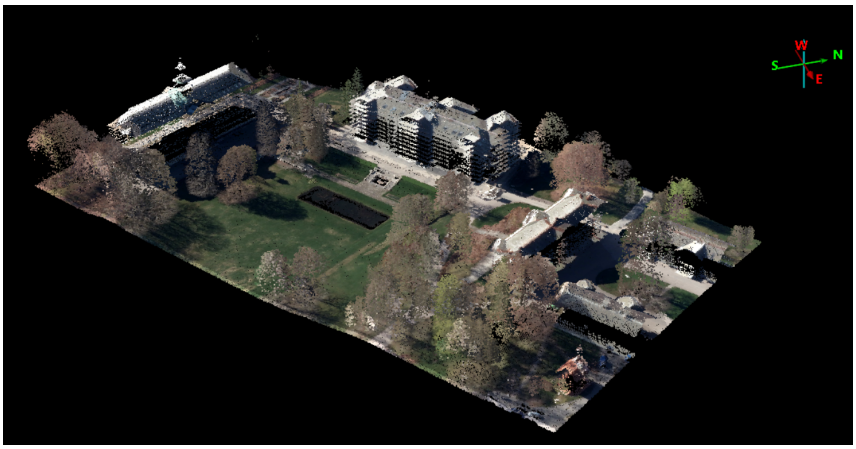


Figure 2.8: Point cloud of NMBU from Kartverket's open-access database. The point cloud has RGB colors and the data points are collected with an airborne sensor.

2.3.1 LiDAR Returns and Intensity

When the LiDAR system fires a laser beam it hits various surface materials. These materials will have different influences on the beam. The intensity of a return is a result of the material's reflection, transmission and absorption of light. This can tell something about what kind of material the surface contains [45]. Moreover, if the laser beam transmits an object, the system can receive multiple returns. Multiple returns are typical when scanning vegetation because beams pass the leaves or pine needles. For instance, if the beam hits a tree, the first return will have information about the top cone. The next returns will give information about the middle part, and the last return will be something impenetrable like the ground. As seen in Figure 2.9 the first return is the top of the tree, then intermediate returns inside the tree, and the last return of the impenetrable ground. Some modern LiDAR scanners also get the intensity of each return as seen on the right side of the Figure 2.9.

The first return is usually the most significant when combining LiDAR with hyperspectral images. Because hyperspectral cameras are passive sensor and will often only see the top surface of an object. Combining these two sensors can give powerful insight when performing surface analysis.

2.3.2 Digital Surface Model

Digital Surface Model (DSM) represents all the natural and built features on the surface of an area [46]. DSM is usually the first return of a LiDAR scan, and gives a representation of the scanned area. The DSM is extracted from the point cloud of an area, and generated by using the height of the first return [47]. DSM has many successful applications, and the most profitable is analyzing the surface layer of an area. For instance, monitoring forest regeneration [48], wildlife fire risk assessment

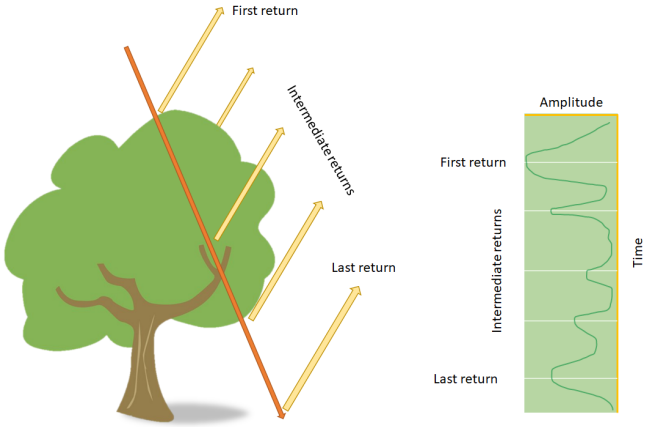


Figure 2.9: Different return of LiDAR with the intensity of the different returns.

[49] and extracting urban features [50]. In Figure 2.10 is an example of a DSM where the colors indicate the height of the surface. Note that the trees and buildings on the hill are more colorful than similar trees and buildings in the lower areas.

2.3.3 Digital Terrain Model

Digital Terrain Model (DTM) represents the terrain surface of an area. To achieve a DTM every object that is not terrain needs to be filtered out from the DSM. Natural vegetation above the terrain and human-made objects like buildings and power lines are some of the objects that need to be filtered out to make a smooth DTM that represents the terrain. To generate a DTM, points in the point cloud must be classified as terrain or not-terrain. For vegetation the last return is often the ground, but for objects like buildings, external information needs to be supplemental or an interpolation from the nearest points. Figure 2.10 shows DTM where color represents the heights. Notice how the middle part of the area has a colorful hill.

2.3.4 Normalized Digital Surface Model

Normalized Digital Surface Model (nDSM) combines DSM and DTM. The nDSM is the relative elevation of an object from the surrounding terrain. By subtracting the DTM from the DSM,

$$nDSM = DSM - DTM, \tag{2}$$

the remaining part is the nDSM, the heights of the objects. This gives a basis for comparison of objects on different terrain elevations. Objects like buildings, trees, cars and power lines now stand out with their relative height from the surrounding ground. Figure 2.10 represents an example of a nDSM. Now see how the hill in the middle of the area is lowered. The colors represent the relative height in the area.

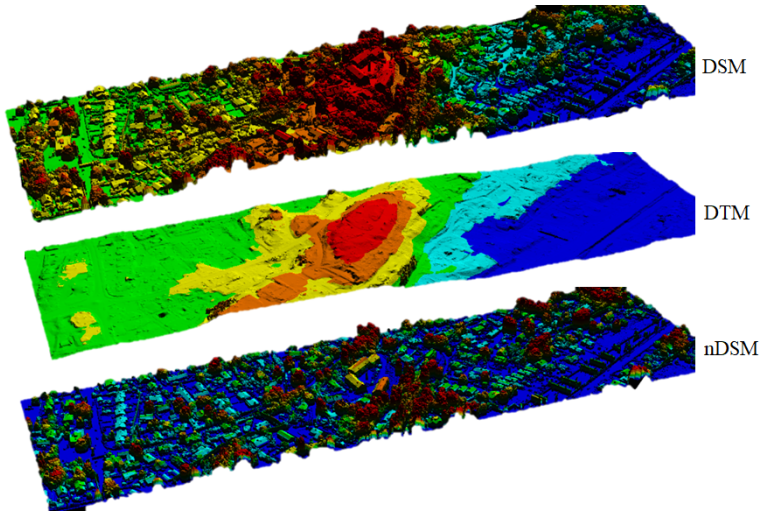


Figure 2.10: DSM, DTM and nDSM.

2.4 Machine Learning Algorithms

Machine learning is a study of algorithms that can derive knowledge from data in order to make predictions [51]. These algorithms run on computers to significantly increase computation speed and increase application. Instead of requiring humans to manually derive rules and build models by analyzing large amounts of data, machine learning offers an effective alternative to learning patterns and knowledge in the data [51] resulting in data-driven processes and freeing human capacity.

Machine learning has a direct impact on almost everyone's life nowadays. From robust email filters, voice recognition software on smartphones to computer vision in medicine or automobiles [52, 53, 54, 55].

The jungle of machine learning can be quite obscure. However, it is possible to categorize in three main types: supervised learning, unsupervised learning and reinforcement learning [51]. Supervised learning refers to an algorithm that trains on training data with known labels. The labels are called the ground truth, and the algorithm makes calculations to imitate this ground truth based on the input data. Training on known data makes it possible to estimate the outcome on unseen data. Unsupervised machine learning trains on data with no known ground truth. The algorithm tries to distinguish based on meaningful structure within the data. These algorithms often group the data into clusters or perform a dimension reduction. The idea behind reinforcement learning is to develop a model that improves its performance on interaction with an environment [51].

The general workflow of a machine learning algorithm seen in Figure 2.11 uses training data, with or without labels. Then creates a model that can give a prediction on new, unseen data.

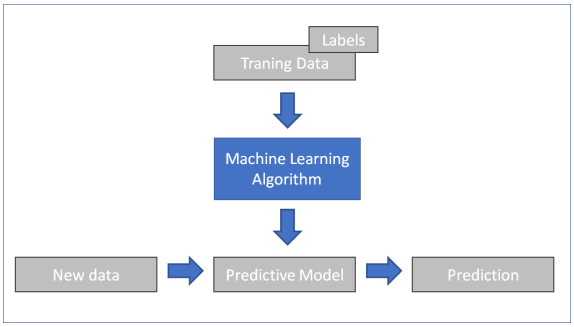


Figure 2.11: Typical workflow of a machine learning algorithm. Training data is used to fit a model to make a prediction on new data. The Figure is inspired by [51].

Even though machine learning has excelled in many scientific fields, there are some limitations. Bias in the data set is the foremost issue when applying machine learning. In man-made data sets there are unconscious and institutional biases already present in society [56]. M. Garcia writes that without careful consideration, our technology will be just as racist, sexist, and xenophobic as we are. For instance, when M. Riberio et al. used one of Google's pre-trained networks to classify husky or wolf in an image. The prediction model discovered that wolf images have snow, so once a husky image with snow is presented to the model, it classifies the image as a wolf [57]. Survivor bias is an example of selecting the desirable bias that can lead to overly optimistic beliefs because multiple failures are ignored. This happened in world war II when the officers examined the surviving airplanes' bullet holes [58]. They thought that reinforcing these areas would make the aircraft withstand longer in combat. The misjudgment by the officers was that they reinforced based on the surviving airplanes, and the casualties that did not return home probably were hit in other areas. Figure 2.12 shows most hit areas on the plane. There is a great chance that the casualties might be hit in the engines or cockpit. Machine learning models might find hidden patterns in the model, but it is important to understand what kinds of patterns this might be.

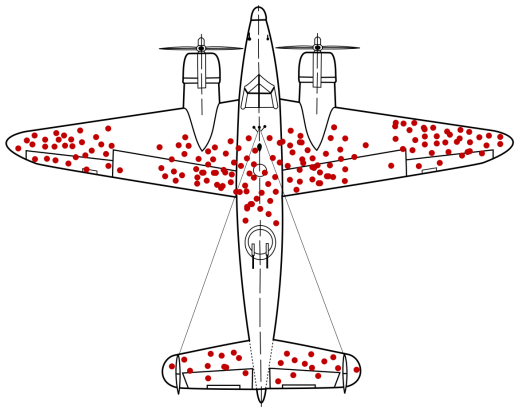


Figure 2.12: Examination of most frequent bullet hits of returning flights in WWII. The officers reinforced these areas without considering the causality airplanes might be hit elsewhere. Figure under CC BY-SA 4.0 International Licence

Another challenge when applying machine learning is overfitting the model. In general, the goal is to maximize the model's predictive accuracy on new data. Not necessarily its accuracy on the training data [59]. It is usually better to make the model solve the general problem with some errors. Instead of making the model fit perfectly into the training data. Figure 2.13 shows a good compromise between under- and overfitting that will make a general model for a problem.

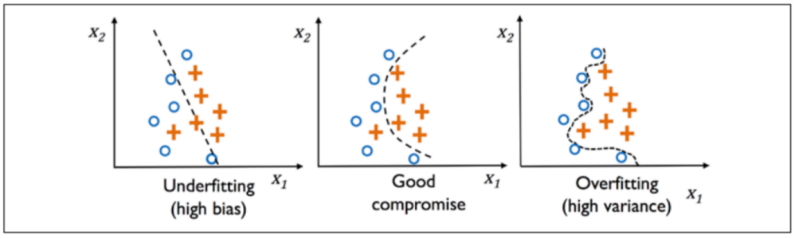


Figure 2.13: An example of under- and overfitting, and how a good compromise can results in a general model [51]. Figure used by permission from Packt publisher. All rights reserved to the author and publisher.

2.5 Artificial Neural Networks

A philosophy in artificial intelligence is that our brain follows the law of physic and chemistry, and as our knowledge expands, it might be possible to reproduce the nervous system with some physical device [60]. The first step towards this was in 1943 when W. McCulloch and W. Pitts wrote a paper on how neurons act [61]. Since that discovery, it went almost two decades before B. Widrow and M. Hoff in 1959 developed ADALINE [62]. An adaptive linear model that is still in

commercial use. In the upcoming decades, neural networks were mostly limited by computational power. In the first years after the turn of the millennium, artificial neural networks raised in popularity. In 2012 AlexNet [63] joined the ImageNet competition and performed considerably better than the runner-ups. This started an era of well-performing neural network. The computer hardware was able to handle an immense amount of calculations, thus commercial usage excelled.

A simple artificial neural network can consist of an input layer, one or more hidden layers and one output layer. Each layer contains a set of nodes, and between the layers are weights connected to nodes in the previous and next layer. These weights are often called parameters and are the editable and adaptive component of a neural network. When the network is fully connected it is often called a multilayer perceptron (MLP) [51]. Figure 2.14 shows a small MLP with two inputs, two hidden layers of three neurons each and output of two neurons. The purple nodes are the bias, they are often set to 1 and has adjustable weights. This last output is the prediction the neural network makes. The orange arrows are the weights from each neuron. Each weight multiplies with the previous neuron inside the neuron and sums up in the current neuron. X_1 and X_2 are the input features, $\hat{y}_1$ and $\hat{y}_2$ are model's predictions.

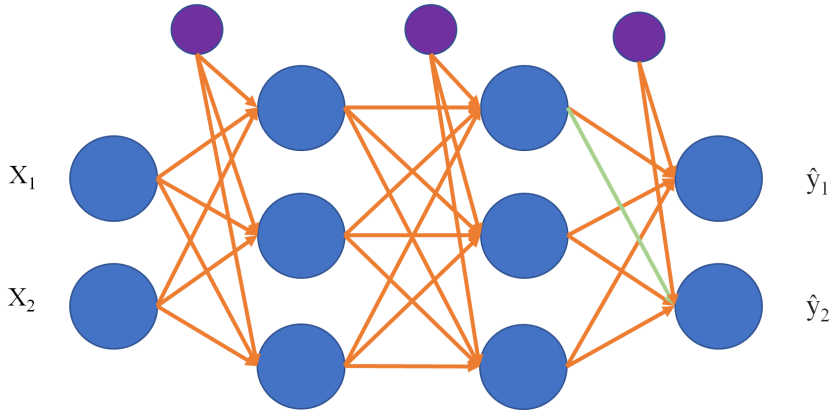


Figure 2.14: Multilayer perceptron with two neurons as input, two hidden layers with three neurons each, and one output. The X in the input data, and ŷ as the prediction. The orange lines are the weights from each neuron. The purple nodes are the bias.

If we look at the upper neuron in the second hidden layer, the multiplication will from neuron will look like

$$a_0^{(2)} = a_0^{(1)} \cdot w_0^{(1)} + a_1^{(1)} \cdot w_1^{(1)} + a_2^{(1)} \cdot w_2^{(1)} + w_{b_0}^{(1)} = \sum_{i=1}^{3} (a_i^{(1)} \cdot w_i^{(1)}) + w_{b_0}^{(1)} \qquad (3)$$

where $a_0^{(2)}$ is the neuron. The 0 stands for the index as the first neuron, the 2 stands for the second layer. w is the corresponding weights from the neurons and w_b is the weight from the bias. Inside each neuron, a nonlinear activation function often controls the sum of neurons and weights. The bias allows us to shift the activation function left or right.

The results of neural networks are often so good that they exceed human capability. Because of the nonlinear characteristics and capacity of some neural networks, the idea is that a neural network can learn anything [64].

2.5.1 Backpropagation

The neural network input values get processed through the network and the network makes a prediction. Based on a loss function and optimizing strategy, the prediction compares to a ground truth, and the parameters adjust to fit the ground truth better. This adjustment is usually done by a process called backpropagation. Backpropagation is an intricate process that solves an optimization problem and adjusts every weight based on that solution. Based on the loss function, backpropagation finds the relative proportion of the change in weights that causes the most rapid decrease in loss. Technically, backpropagation finds the negative gradient of a loss function, the difference between a predicted and a true state. Then takes the partially derived of this function for each weight and multiplies with an often low learning rate. The weight change can be seen as

$$w := w + \Delta w, \text{ where } \Delta w = -\eta J(w), \tag{4}$$

where J is the loss function and η is the learning rate. If we look at the node from earlier and the connecting weight with the green line to the top right, the partial derivative of the weight can be seen as

$$\frac{\partial}{\partial w_{0,1}^{out}} J(W) = a_0^{(2)} \delta_1^{(out)}, \tag{5}$$

where δ_1^{out} is the error term for the second out node. In this case δ_1^{out} will be the difference between the prediction and the value after the activation function. The direction for the optimization often ends up at a local minimum. This minimum describes the least wrong, or the "rightest", a neural network can take.

2.5.2 Activation Function

Each node has an activation function. This function makes the neural network learn nonlinear conditions. Two of the most popular activation functions are sigmoid [65] and rectified linear unit (ReLU) [66]. The sigmoid function is characterized as an S shape function that converts a real value into a value between 0 and 1. One challenge with the sigmoid function is when derivative of larger numbers can be

close to zero. As seen above, when performing backpropagation we find the partial derivative of the loss function. If the derivative is close to zero, the change in the weights is close to zero, and the neural network does not learn. This can cause the vanishing gradients problem [67]. The sigmoid function is given by

$$\phi_{sigmoid}(x) = \frac{1}{1 + e^{-x}}. \tag{6}$$

To the left in Figure 2.15 is the sigmoid function and its derivative.

ReLU is mainly known for solving the computational problem of vanishing gradients. ReLU is defined as

$$\phi_{ReLU}(x) = max(0, x), \tag{7}$$

and takes the max of 0 or the input value. Every value beneath 0 will be filtered by the ReLU function. That means that the derivative of the function will either be 0 or 1. In Figure 2.15 to the right is this function with its derivative illustrated.

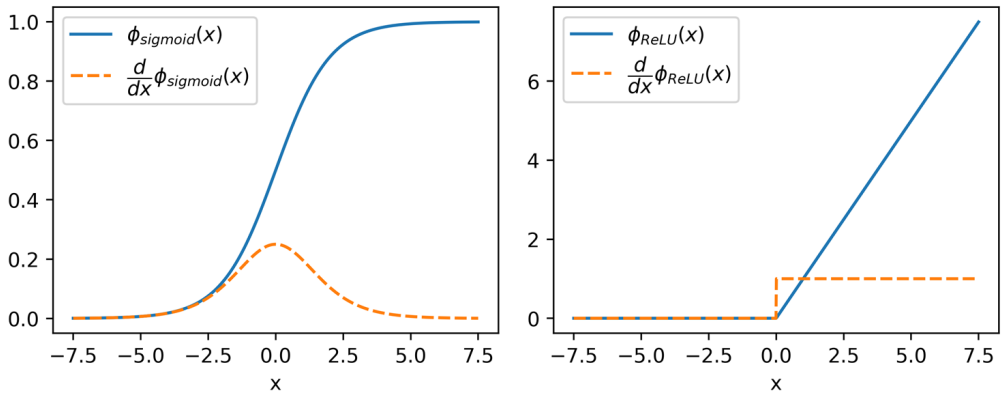


Figure 2.15: Sigmoid and ReLU activation functions with corresponding derivatives.

The last activation function that needs to be mention is the softmax activation function. This function is often used as the last layer of a neural network. It is a normalized exponential function [68] for multiple dimensions, and used to normalize the last output to be a probability distribution of the outputs. With K as number of classes, the softmax function can be described as

$$\phi_{softmax,i}(x) = \frac{e^{x_i}}{\sum_{j=1}^{K} e^{x_j}} \text{ for } i = 1, ..., K \text{ and } x = x_1, ..., x_K). \tag{8}$$

2.5.3 Convolution Neural Networks

A convolution neural network (CNN) is an advancement of the artificial neural network and is especially well-performing in image analysis. CNNs are a family of

models that were originally inspired by how the visual cortex of the human brain recognized objects [51]. It was first introduced when Y. LeCun et al. proposed a neural network architecture to recognize handwritten digits from images [69]. CNN is similar to ANN with an input layer, several hidden layers and an output layer. They both arose from the same concept of adaptable weights. However, there are notable differences in the arrangement of nodes and weights. CNN contains filters that maneuver over the image and do a convolution. In these filters, the same weights get used in different patches of the input image. A convolution is a relation between two vectors. It can be formulated as

$$\mathbf{y} = \mathbf{x} * \mathbf{w}, \tag{9}$$

where $\mathbf{x}$ is the input and $\mathbf{y}$ is the output. $\mathbf{w}$ is the relation between those two vectors. In the convolution, the $\mathbf{w}$ uses a sliding window approach to get the output, dot product, for each element in the $\mathbf{x}$ vector [51]. This can be formulated as

$$\mathbf{y} = \mathbf{x} * \mathbf{w} \longrightarrow y[i] = \sum_{k=0}^{k=m} x[i+m-k]w[k], \tag{10}$$

and i is the index and m is the length of the vector. To solve the problem where the convolution hits the edges of a vector, there is usually a padding of zeros to control the shape of the output.

The idea of CNN is to extract important features from the data. These can be a color combination of pixels, intensity in the color channels or the shape of an object. Since CNN has filters that maneuver over the data, the filters will examine an area including the surrounding pixels. By doing so, CNN will understand both the spatial and spectral context.

Figure 2.16 shows a short CNN with an input image and convolution layers that extract the import features of the image. Lastly the filters are flattened and feed into an artificial neural network that classifies the image.

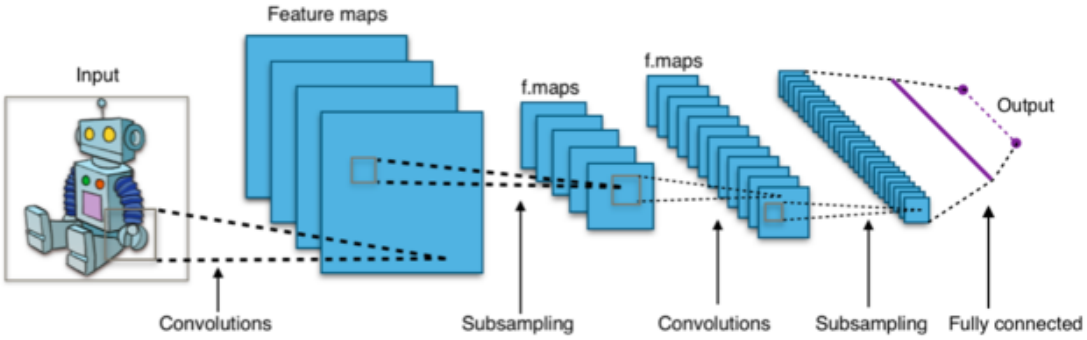


Figure 2.16: CNN with convolution layers that extract the import features of the image. Then fed into an artificial neural network to classify the image. Figure used under a CC BY-SA 4.0 International license.

2.5.4 Layers in CNN

There are many layers that can be used in a CNN. Here is a collection of the relevant in this report.

Convolution 2D is the most interesting layer in a CNN. This layer has a kernel size, often as a square of 3x3 pixels or bigger, with associating weights for each pixel in the filter. These weights are the adjustable parameters that learn the features of the data. The filter will convolve over the data, and the stride decides how many pixels the filter will move horizontal or vertical over the image. This will influence the output from the layer. For instance, if the filter skips every second pixel, the output will be half of the original image. If the filter iterates every pixel and the image has a padding on the edges to including edge pixels, the output will be the same size as the input.

Maxpooling layer has a filter like convolution 2D, but with no weights. This filter extracts the maximum value of in filter area. By doing so, the data gets reduced and hopefully the most important features are extracted.

Concatenate has the ability to merge two tensors, or arrays, in the feature dimension. In an image example, the concatenate layer merges the filters in the third dimension, often the feature color channel.

Add layer is almost like concatenate, but instead of merging it adds the values and keeps its shape throughout the layer.

BatchNormalization normalizes the values so the output mean is close to 0 and standard derivation is close to 1. The layer learns the weights that lead to mean near 0 and standard derivation near 1.

Activation layer is a layer that handles the activation. This filter transforms the data based on the activation selected.

Upsamlingsampling layer upsample the data. It takes the nearest data point and duplicates it. The interpolation of this can vary based on the desired result.

2.5.5 Semantic Segmentation

Semantic segmentation is the process of partitioning a digital image into multiple image segments [70]. This is done by giving each pixel in the image a label. In comparison to object detection, semantic segmentation disregards identifying objects but focuses on classifying every pixel in the image. Instance segmentation identifies the differences between the objects that are not background. Figure 2.17 shows an example where the cat and dog are different from each other and the background.

Figure 2.17: Semantic segmentation of dog, cat and background.

2.5.6 U-net

U-net is a convolution neural network architecture with strong image segmentation results. O. Ronneberger et al. built the network in 2015 and outperformed several state-of-the-art models [54]. The founders of the architecture say "The architecture consists of a contracting path to capture context and a symmetric expanding path that enables precise localization" [54]. This characteristic is suitable for semantic segmentation in image data where the goal is to localize and classify objects.

U-net is an architecture based on CNN layers. As seen in Figure 2.18 the contracting path consists of the typical architecture of a convolutions network. Each step consists of two 3x3 pixels convolutions with nonlinear activation. Followed by a 2x2 pixels maxpooling layer, which takes the max value of a 2x2 pixels filter and compresses the information into a smaller scale [54]. At each downsample the filter channels double. On the expanding path, 2x2 pixels up-convolution layers do the opposite of maxpooling by scaling up the data and entering a two 3x3 pixels convolution layers per step. At the same time, samples from the contraction path get copied and concatenated with corresponding parts after up-convolution.

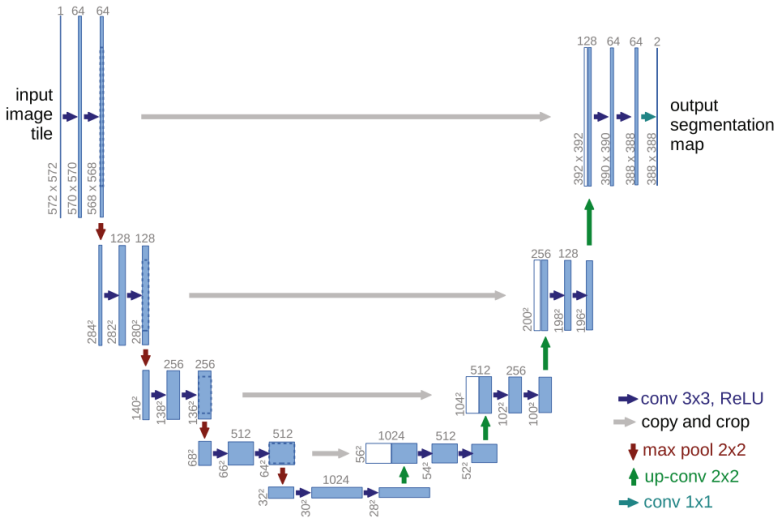


Figure 2.18: U-net as shown in [54]. This U-net is not symmetric due to no padding layers. Figure used under licence by SPRINGER NATURE © [2015].

In Ronnerberg's application, the U-net was primarily used on medical examples with heavy data augmentation. But the idea of semantic segmentation has been applied to other scientific fields. For instance, Z. Pan et al. used U-net in 2020 to classify buildings and objects in urban villages with satellite data [71].

2.5.7 Deep Residual Network - ResNet-34

Deep residual network (ResNet) was first introduced by K. He et al. in 2016 as a response to the challenge of training deeper neural network [72]. From the original paper K. He et al. write: "We explicitly reformulate the layers as learning residual functions with reference to the layer inputs, instead of learning unreferenced functions. We provide comprehensive empirical evidence showing that these residual networks are easier to optimize, and can gain accuracy from considerably increased depth." [72].

The trend of neural networks was that they became deeper and the number of adjustable weights skyrocketed. These deeper networks perform better [73, 74] than previous networks. Driven by this correlation between deeper nets and well-performing nets, a question arises: *Is learning better networks as easy as stacking more layers?* [72]. However, when the nets become too deep, new hitches arise. One obstacle for a very deep network was that the error from the prediction did not reach the start of the network when performing backpropagation. This problem is reduced by normalized initialization [75] and intermediate normalization layers [76]. K. He et al. then address a new challenge when deeper networks start to converge, a degradation problem exposes. Accuracy metrics saturate then degrade rapidly.

This problem is not caused by overfitting, but adding more layers lead to higher training error, K. He et al. reported and experienced in [72, 77].

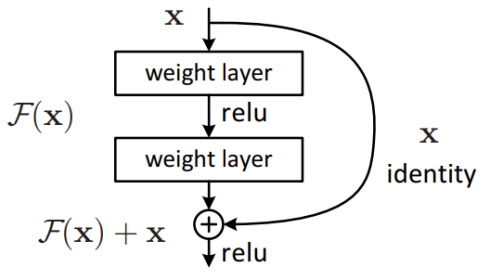


Figure 2.19: Deep residual network building block. In each convolution block a skip connection lets the network learn the residual mapping [72] © [2016] IEEE.

The solution for the degradation of the accuracy problem is deep residual networks. K. He et al. say that preceding neural network with many stacked layers will hopefully learn the desired underlying mapping, but with ResNet, these layers explicitly fit a residual mapping [72]. As seen in Figure 2.19, a ResNet block shows how the weighted layers need to learn a residual mapping to optimize. The hypothesis is that optimizing residual mapping is easier than optimizing the original, unreferenced mapping [72]. Additionally, the skip connections give a shorter way back to the earlier layers when performing backpropagation.

ResNet-34 got its name from the 34 convolution layers the network has. Other ResNet configurations span from 18 to 152 layers. Originally, the ResNet-34 output was a straightforward vector classifier of images, but then F. Milletari presented a U-net-like architecture that incorporates ResNet-like residual blocks [78]. Combining the ResNet blocks and U-net architecture, with internal skip blocks in down- and up-sampling. Alongside the larger skip connections between contracting and expansion paths. The new combined architecture can lead to faster optimization and improvement for semantic segmentation [79].

2.6 Accuracy Assessment

Measuring the model's performance is an important part of evaluation and building good, reliable statistical models. There are many metrics to measure performance, and the main idea is how good a model predicts compared to the true labels. Some metrics emphasize the number of recognized samples, and others punish wrongly classified samples. Many of the metrics depend on a confusion matrix. A confusion matrix tells how many samples that are correctly or falsely classified. In a binary problem these represent as true positive (TP), correctly classified positive class. True negative (TN) is correctly classified negative class. False positive (FP) is

falsely classified as positive. And lastly, false negative (FN) is falsely classified as a negative class. In Figure 2.20, all four categories are in the confusion matrix.

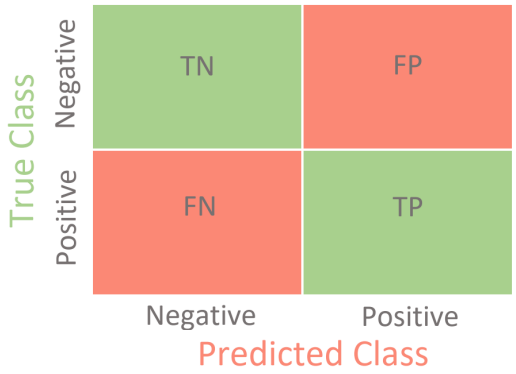


Figure 2.20: Binary confusion matrix.

When a confusion matrix has a multi label problem, the positive and negative classes are substituted with the labels in the data set. Then the confusion matrix C will be $K \times K$, where K is the number of classes. C will show which samples are correctly classified, and if not — which other class it gets classified as.

The most convenient metric is accuracy. Accuracy tells us how many of the samples were classified correctly. Taking a look at Figure 2.20 with the confusion matrix, accuracy uses TP and TN divided over all the samples. Formulated as

$$Accuracy = \frac{TP + TN}{TP + TN + FN + FP}, \tag{11}$$

accuracy gives an easy understanding metrics and performs well on balanced data sets.

2.6.1 F1 Score

F1 score is a well-suited metric for imbalance data sets. C. J. van Rijsbergen is viewed as the founder of F1 score from his book *Information Retrieval* [80]. He explained the metrics with "... measures the effectiveness of retrieval with respect to a user who attaches β times as much importance to recall as precision.". Here he named the score E, but it later got its well-known F1 name. The β is often set to 1, and then the recall and precision are equally weighted. A good F1 score means a low share of false positives and negatives. The formula

$$F1 = \frac{TP}{TP + \frac{1}{2}(FP + FN)} \tag{12}$$

shows the F1 score. It ranges between 0 and 1, where 1 is a perfect score. Some criticize F1 for not being symmetric. F1 ignores the true negatives, and can be

misleading if a set is imbalance in that favor. In multi class problems the weighting of F1 can be done in three different ways: micro, macro and weighted. Micro calculates the metric globally using TP, FN and FP. Macro calculates the metric for each class and finds the unweighted mean. Weighted calculates the metric for each class and then takes the average weighted for each class based on the number of samples in that class compared to the whole set.

2.6.2 Matthews Correlation Coefficient

Matthews Correlation Coefficient (MCC) is originally a metric to measure the quality of binary classifications introduced by B. W. Matthews in 1975 [81]. The coefficient is especially suited for imbalanced data sets [82]. Taking true positives and false negatives into account makes it easy to spot if the minority class predicts wrongly. MCC spans from -1 to +1, where 0 is the same as a random guess. MCC formulates as

$$MCC = \frac{TP \cdot TN - FP \cdot FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}}. \tag{13}$$

There are some cases when the denominator becomes zero or near to zero. This will result in an artificial high score, and needs to be adjusted. Compared to other metrics like accuracy and F1 score, MCC performs well and flags out both false positives and false negatives. Figure 2.21 is a binary problem with slopes of accuracy, F1 score and MCC. On top of the graph are confusion matrices of three episodes. The first is the worst-case prediction, then the last is the best-case prediction. Firstly, all samples are misclassified and MCC is the only metric that detects this. Secondly, all false negatives have become true negatives, and half of the samples are recognized. Accuracy yields 0.5, and F1 score yields over 0.6, but MCC 0.0. This shows the weakness of F1. When all samples are rightly classified all three metrics are in unison.

Figure A.1 in Appendix I shows an imbalanced data set (80/20 distribution). The majority class can be seen as the background and the minority as an object. The graph goes from all wrong answers, and then all the background samples get classified as correct. After that, the foreground samples get classified correctly. MCC can explain the whole situation, from all wrong to all right. This is especially evident when the class distribution is imbalanced. When every sample is predicted as background, accuracy yields 80% correct, but MCC and F1 do not increase before the foreground is predicted right.

For multi class problems, MCC is generalized. J. Gorodkin introduced this generalization in 2004 [83], and called the generalization for R_K statistics. K is for K-different classes defined by a confusion matrix C which is $K \times K$. The formula

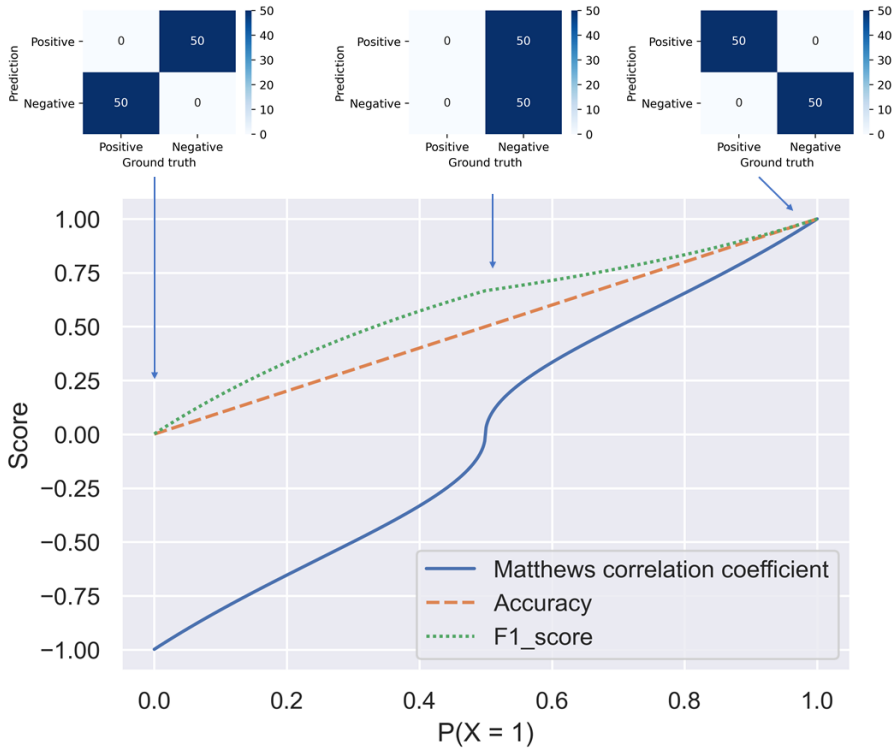


Figure 2.21: Comparison of MCC, F1 score and accuracy. The situation for the score is described as confusion matrices above.

$$MCC = \frac{\Sigma_k \Sigma_l \Sigma_m C_{kk} C_{lm} - C_{kl} C_{mk}}{\sqrt{\Sigma_k (\Sigma_l C_{kl}) \cdot (\Sigma_{k'|k' \neq k} \Sigma_{l'} C_{k'l'})} \sqrt{\Sigma_k (\Sigma_l C_{lk}) \cdot (\Sigma_{k'|k' \neq k} \Sigma_{l'} C l' k')}} \tag{14}$$

explains MCC for multi class. It still ranges in the same area and has the same attributes as the binary version.

2.7 Loss Function and Optimization Strategy

The loss function is a method that determines how well a particular algorithm learns the data. The loss function explains the gap between prediction and actual values. A smaller gap indicates a better imitation of the data. An optimization strategy is an algorithm to optimize the loss function. The goal is to find the minimum of the loss function, e.g. the point at the prediction is most similar to the truth. Choosing the proper loss function decides how a neural network will learn the underlying patterns and structure of the data. Therefore, the loss function characteristics must be well suited for the problem. Loss functions exist in many variants. Some are made for classifications, and others are made for regression. Some are better on imbalanced data sets and others are better on object reconditions.

2.7.1 Focal Loss

Focal loss is a loss design for classification on extremely imbalanced data sets. T. Lin et al. introduced this to address a one-step solution for imbalanced data in image sets [84] where the background had a majority of the samples. This was earlier done in an R-CNN framework and two-stage decoder [85], but T. Lin et al. showed that their Focal loss matched both computation speed and performance by outperforming the earlier state-of-the-art solutions. Focal loss consists of the normal categorical crossentropy (CE) part, and a γ part where high value of a γ gives a more forgiving loss when making an error. Figure 2.22 illustrates Focal loss with different values for γ.

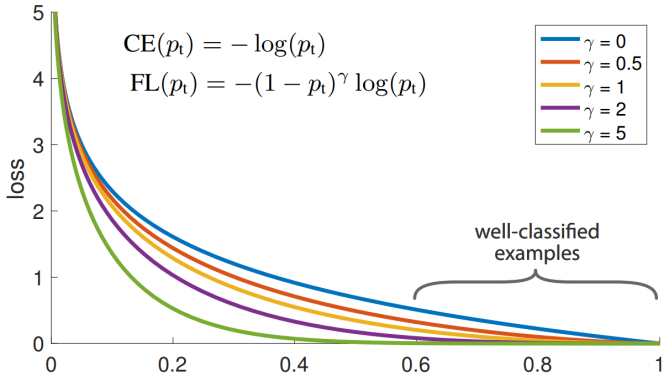


Figure 2.22: Focal loss from the original paper by T. Lin et al. [84]. Focal loss explains the loss function based on the correctness of a classification. γ is editable and adjusts the amount of loss © [2017] IEEE.

2.7.2 Jaccard Loss

Jaccard index is a metric known as the Jaccard similarity coefficient. It is used to detect similarities and dissimilarities of sample sets. Paul Jaccard introduced the Jaccard loss [86] and derived it from the ratio of intersection over union (IoU). Jaccard takes the size interception and is divided by the union of the two samples, shown in formula

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|}. \tag{15}$$

This takes the true positives and divides them by true positive, false negative and false positive. Figure 2.23 shows three examples of the Jaccard index for overlapping shapes.

Jaccard index can be used as a loss function, often called Jaccard Distance. Jaccard index ranges between 0 and 1, so to apply Jaccard as loss use

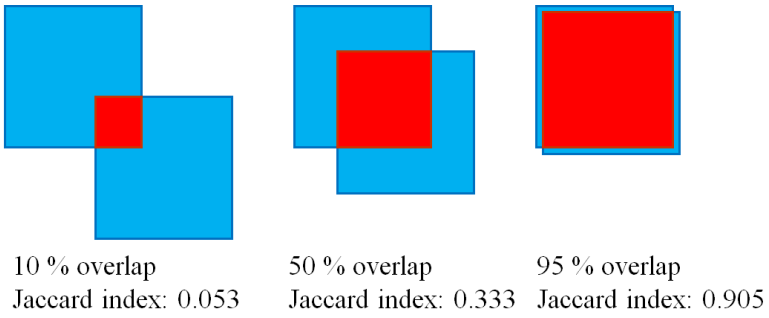


Figure 2.23: Overlapping shapes with their Jaccard index. A higher score is better.

$$J_{loss}(A, B) = 1 - J(A, B). \tag{16}$$

This loss function performs well in object detection and is a good fit when used in computer vision [87].

2.7.3 Adam - Adaptive Moment Estimation

The key to optimizing a neural network is finding the minimum of a loss function. A state-of-the-art method for this is called Adam (adaptive moment estimation). Adam was introduced by D. P. Kingma et al. in 2014. Adam uses adaptive momentum and learning rate for effective optimization. Momentum accelerates the weight updates in promising directions. The authors say, "Adam is an algorithm for first-order gradient-based optimization of stochastic objective functions, based on adaptive estimates of lower-order moments. Adam is computationally efficient, has little memory requirement, invariant to diagonal rescaling of gradients, and is well suited for problems that are large in terms of data and parameters." [88]. Adam has three parameters that need to be declared. The first is the learning rate. This is usually a low default value at 0.001. Then there are two exponential decay rates for the moment estimates; β_1 and β_2. They are set on values between 0 and 1. The most important steps in the algorithm are to find the gradients at a timestep, and then correct the first and second momentum bias, then update the parameters.

2.8 Object Detection

Object detection is a technique in computer vision for finding objects in digital images or videos based on qualitative attributes (e.g., color homogeneity), low-level features (e.g., color model component's distribution), object spatial relations and multimedia processing methods (e.g., color clustering) [89]. This technique has plenty of applications, and in the later years excelled in popularity due to the commercialization of neural networks. Neural networks are rather complex and require a lot of data and computation power. Even though machine learning yields

sturdy results, there are older, simpler, faster and more manual algorithms that give robust products.

2.8.1 Watershed Algorithm

The watershed algorithm is an object detection algorithm used in image processing. The algorithm was first introduced by S. Beucher and C. Lantuejoul in 1979 [90]. They explain it as a non-parametric method for contour extraction on a grayscale image. The method relies on defining the contour as the watershed gradient modulus of the light function.

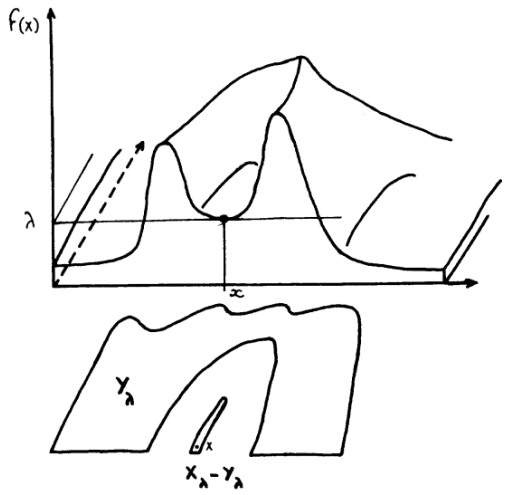


Figure 2.24: Part of the watershed algorithm. The landscape $f(x)$ show the distance from the center of the object to an edge. The original paper finds a local minimum to watershed [90] © [1979] IEEE.

The idea behind the algorithm is to find edges in an image, then use morphology to find the center of objects. Thus measure the distance from the center to the closest edges. Then this distance can be seen as a landscape of $f(x)$ as seen in Figure 2.24. The further away the center of the object is from the edge the greater the value. In the figure, the larger object also gets a high value that can be represented on the third axis. S. Beucher and C. Lantuejoul used a local minimum to set a threshold for the watershed. The next operation in the method is to flood everything beneath this local minimum for $f(x)$. Like water floods a mountain landscape, this method floods and removes objects that are not high enough. The remaining areas peak out of the water, and objects get extracted from an image.

3 Method

3.1 Chapter Description

This chapter presents the method in this thesis. The chapter initially starts with a description of the acquisition, processing and correction of the data. Then it continues explaining what kinds of software and hardware were used. The rest of the chapter is a detailed description of preparing the data for analysis, the workflow, a step-by-step description of the process of generating the ground truth and the progress of model parameter selection. The chapter rounds off by summarizing the final model.

3.2 Data Acquisition

In this study, airborne hyperspectral images and LiDAR scans are analyzed to extract valuable information from an area. The images and scans have been produced and preprocessed by Terratec AS, Norway's largest geodata supplier. Bærum municipality is the project manager and wants to use the data for data analysis and mapping of their areas. The region of interest is Høvik, a district in Bærum municipality in Norway, and is illustrated in Figure 3.1.

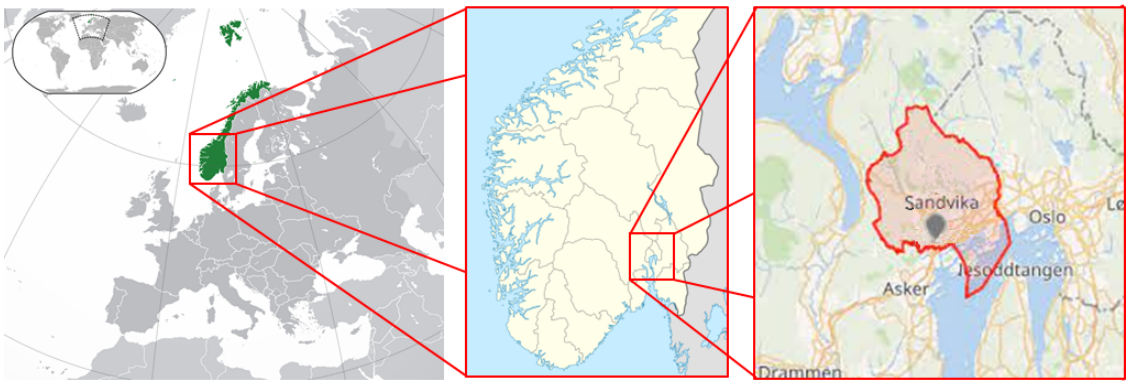


Figure 3.1: Area of interest: Høvik, a district right outside the city Sandvika in Bærum municipality.

The hyperspectral images and LiDAR scans were acquired in August 2019 and June 2021. The original project in 2019 had three different locations including Bærum with a total area of 37.4 km^2. In this study, only scans from Høvik are applied. The flight in 2021 centered around Bærum municipality and covered a

larger area in total 53.0 km^2. Both flights had an altitude of 1100 meters above the terrain and a maximal speed of 130 knots. In the first flight, the direction of the strips was east-west as seen on the left in Figure 3.2. In the second data set from 2021, the flight direction was north-south seen on the right in Figure 3.2. One overlying goal is to see changes over time from the data, and therefore only areas with overlapping flights are used. To narrow the area further, three regions are extracted to examine the hyperspectral and LiDAR data. The flight numbers are 04, 05 and 07 from 2019, and flights 044 to 048 from 2021. Figure 3.3 shows these three areas with a red borderline, and they cover approximately 0.65 km^2. All three regions have information from 2019 and 2021, but 04 was mainly used in the analysis in this study.

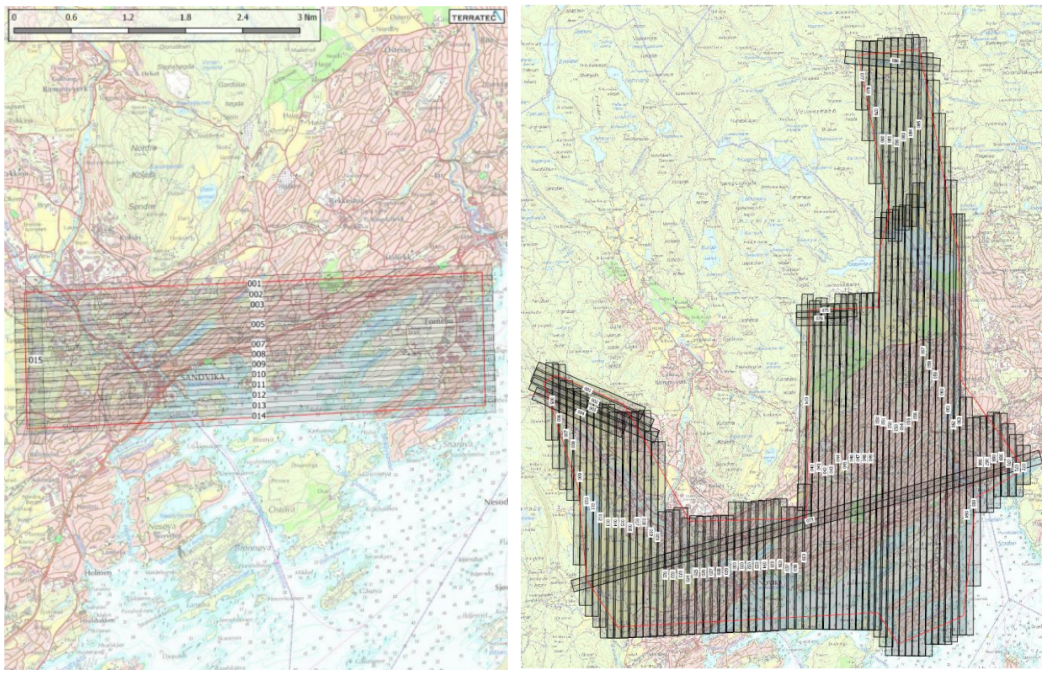


Figure 3.2: Flight plan from 2019 and 2021 over Sandvika. To the left is the plan from 2019, and to the right is 2021. Flight plan by Terratec [91].

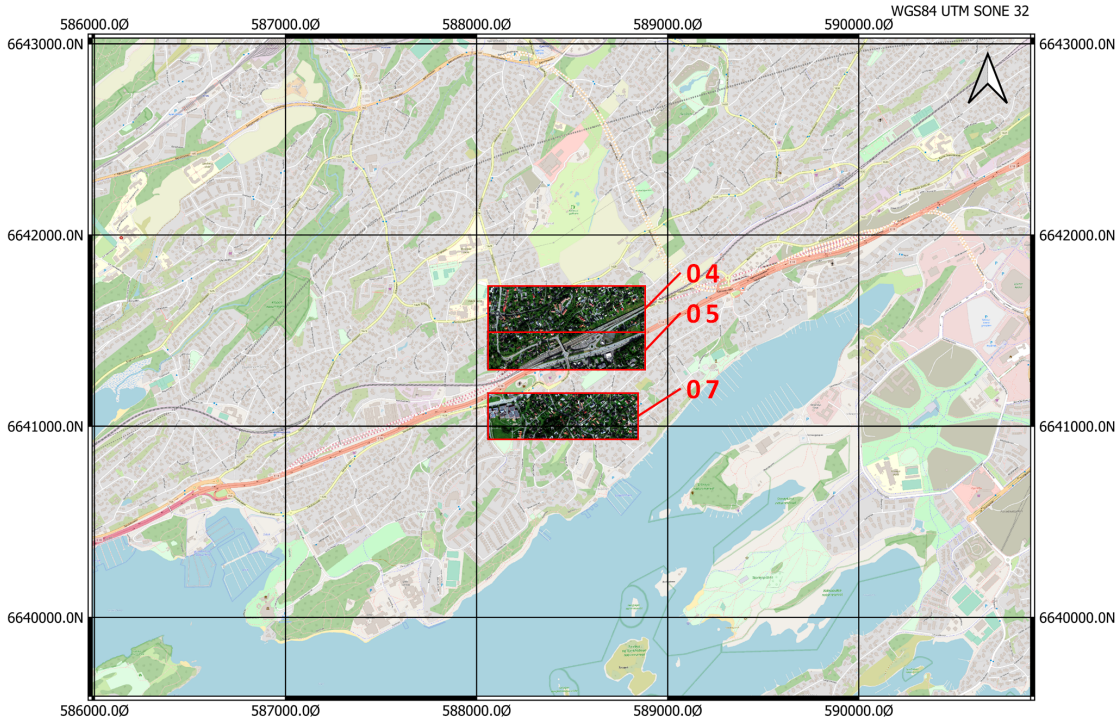


Figure 3.3: A closer look at areas examined in the study. RGB images from three flights.

3.2.1 Airborne System Layout

Both sensors are mounted to a gyro frame and laid for a maximum of 16 degrees opening angle for one of the cameras. The system layout in the aircraft is shown in Figure 3.4, and shows the gyro mount for both LiDAR and hyperspectral sensors. The sensors are mounted next to each other in the airplane.

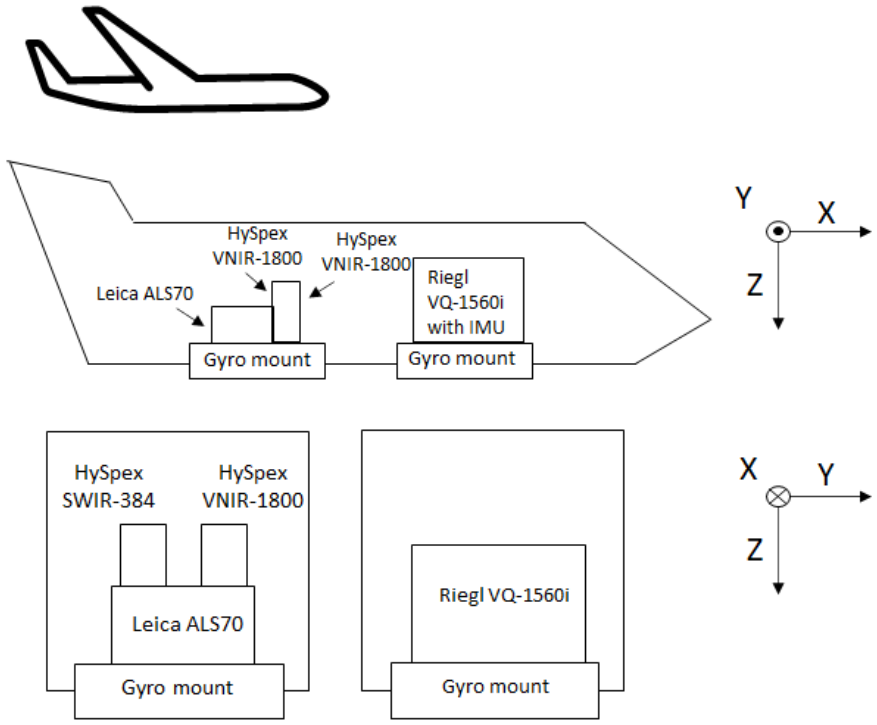


Figure 3.4: System layout from the flights by Terratec.

3.2.2 Sensor Specific Information

The hyperspectral system contains two sensors: HySpex VNIR-1800 and HySpex SWIR-384. Capturing respectively visible near infrared (VNIR) and short wave infrared (SWIR) [91]. These two sensors combined can capture wavelengths ranging from 400nm to 2500nm with a total of 474 spectral channels. The sensors are manufactured by Norsk Elektro Optikk AS (NEO), a Norwegian company specializing in the field of photonics. HySpex is NEO's product of hyperspectral cameras. Both cameras are based on pushbroom scanning principle. The sensor captures a line of spectral data while the airplane moves forward. The LiDAR data was collected simultaneously as the hyperspectral images. A Riegl VQ-1560i laser scanner records it with an ingratiated Inertial Measurment Unit (IMU). The LiDAR sensor uses a laser beam with wavelength of 1064nm [91]. Table 3.1 is the specifications for both the hyperspectral cameras.

3.2.3 Preprocessing Raw Data

All scans are georeferenced and orthorectified. The data was orthorectified with Parge (3.4) software based on a 30 cm digital terrain model obtained using the Leica ALS70 laser next to the hyperspectral sensors. Georeferencening and ortorec-

Table 3.1: Specifications for hyperspectral cameras [91].

Specification	VNIR-1800	SWIR-384
Spectral range	400 - 1000nm	930 - 2500nm
Spatial pixels	1800	384
Spectral channels	186	288
FOV	17 degree	16 degree
Pixel FOV across/along	0.16/0.32 mrad	0.73/0.73 mrad
Spatial resolution	0.3 m pr pixel	0.7 m pr pixel
Bit resolution	16 bit	16 bit
Dynamic range	2000	7500
Noise floor	2.4 e-	150 e-
Max speed	260 fps	400 fps

tification are done using nearest-neighbor interpolation. In the flight over Bærum it was control points on the ground for more exact measurements. Combined with GNSS and control points, the images and scans are georeference to the WGS84 UTM32 coordinate system.

The hyperspectral data obtained by Terratec AS is in radiance. The airborne photos were taken at an altitude where atmospheric noise affected the data. Therefore, the hyperspectral data is atmospheric corrected. Atmospheric correction is executed using ATCOR-4 software for airborne imagery. ATCOR-4 performs atmospheric corrections and estimates the surface reflectance. ATCOR uses AFRL MODTRAN code to determine the atmospheric look-up table database, and the parameters were set up manually.

VNIR data has a resolution on 0.3 meter per pixel, and SWIR has a resolution on 0.7 meter per pixel. Gram Schmidt Spectral Sharpening [92] was performed to increase the resolution of the SWIR images. By doing so, both hyperspectral images cover the same area with the same amount of pixels. The LiDAR data is extracted from the point cloud to a raster file using the program Quick Terrain Modeller (QTM).

Some of the wavelength channels are removed due to interference with water in the atmosphere, these channels are so noisy and inconsistent there is challenging to use in an analysis. Channels removed because of the water interference are wavelengths between 1354nm and 1475nm, and between 1803nm and 2033nm. Some channels have a high amount of outliers and are removed, and all the channels above 2400nm are so noisy there are not in use.

3.3 Software, Hardware and Memory

Python is mainly used in data management and analysis. I use QGIS, an open source GIS platform for visualization and exploration of hyperspectral data. For

the LiDAR point cloud I use Quick Terrain Model (QTM). Table B.1 in Appendix II is an overview of program and all module versions used in Python. Tensorflow is the main framework when working with machine learning, and OpenCV has a built-in method for image processing. Matplotlib is used for plotting and visualization of the images.

I was fortunate to have the opportunity to use effective computational hardware. Minor tasks and exploration can easily be done on a standard laptop. More computational demanding calculations need better hardware and a strong GPU. Therefore, training of the neural networks was done on external computers with high capacity. The experimental phase of machine learning was done in Google's Colaboratory, an IPython Kernel in the cloud service by Google. This virtual environment provides GPU and all the other Python 3 environments needed in the study. The PRO version of Colaboratory was used to expand computation time on GPUs. Another high-capacity computer cluster applied in the project is Orion High-Preforming Computing cluster located at NMBU with even more computational power. It runs on Singularity a free, cross-platform and open-source computer program that performs operating-system-level virtualization, also known as containerization. These containers need to be built inside a system for each project, and demand more technological insight by the user. In Table 3.2 is an overview of the hardware used. In the Orion cluster it is possible to combine more GPU and CPU to get up to 100 GB of GPU and over 1 TB of RAM.

Table 3.2: Overview of Hardware used in the study.

System	**CPU**	**RAM**	**GPU name**	**GPU RAM**
Standard Laptop	2.7 GHz	8.00 GB	-	-
Google Colab Pro	2.2 GHz	25.7 GB	Tesla T4	27.3 GB
Orion cluster NMBU	2.0 GHz	30.8 GB	Quadro RTX 8000	48.0 GB

The data in this study requires a lot of disk space. In particular the hyperspectral data takes a lot of memory. Just for the study area in Figure 3.3 the areas 04, 05 and 07, the hyperspectral and LiDAR data require 32 GB of memory. The data is put on external drivers to ease up the storage.

3.4 Region of Interest

The region of interest in this study is the 04 area from Figure 3.3. The area has overlapping flights from several years and is of high quality. A smaller area is easier to work with on a standard computer with computational limits. Figure 3.5 shows the area in RGB image from the hyperspectral data.

Figure 3.5: Region of interest, flight 04 from June 2021.

3.5 Preparation of Data in the Study Area

Preparations of the study area must be done before the data is ready for analysis. The data from the area comes in different file formats. LiDAR data is in a laser point cloud (.las), and needs to be converted to a 2D raster file that fits the area. Both DSM and DTM raster files are used to create nDSM. The hyperspectral data is stored in High Dynamic Range (.hdr) raster file, one for VNIR and one for SWIR. All three files were converted into raster files fitted for the exact study area and then stacked in the channel dimension. Lastly, the data was saved as a numpy file (.npy) fit for machine learning algorithms. The machine learning algorithms do not use the georeference properties from a raster file, so any file format is viable. Numpy was chosen for it is convenient for handling and manipulating high-dimensional data.

3.5.1 Create nDSM

A nDSM is produced from the DTM and DSM. Metadata and georefrences from DSM were copied directly to nDSM. Figure 3.6 shows all three models. With nDSM it is possible to examine if an object is elevated from the ground. Hyperspectral images show the surface reflectance and combined with height data it can be possible to tell the exact height of the object independent of terrain surface and shape.

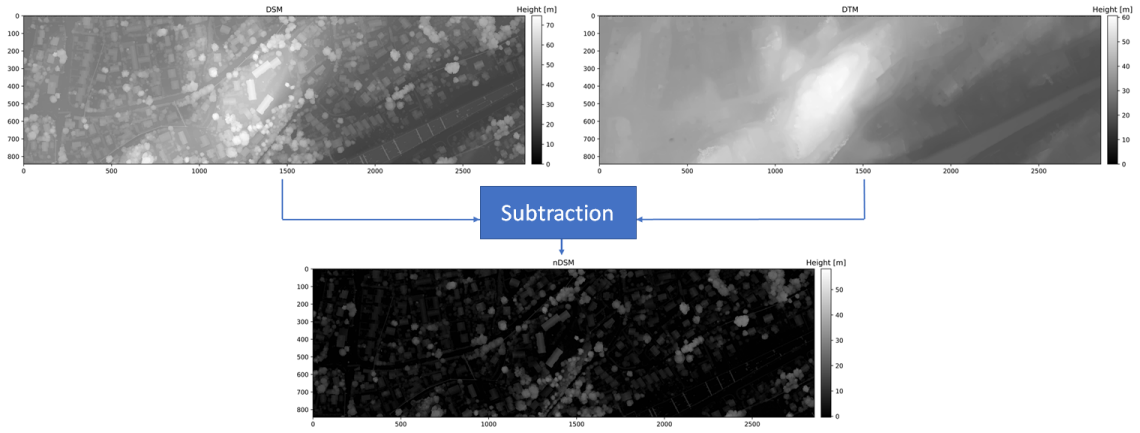


Figure 3.6: DTM is subtracted from DSM to make a nDSM model. Heights are relative to the lowest point in the area.

3.5.2 Stacking SWIR, VNIR and nDSM

SWIR and VNIR photos are taken from two different sensors, but it is desirable to combine the data into a nearly continuous spectrum from 400nm to 2400nm. VNIR contains 176 bands from 400nm to 1000nm, and SWIR contains 222 bands from 930nm to 2500nm. Some of the bands are overlapping, but in that from SWIR is kept in that case. The stacking of the data results in a total of 398 spectral bands with almost continuous spectral information. In addition to spectral information, it is beneficial to use available data about height. Therefore, nDSM data is stacked in the same format as hyperspectral data. The data now contains 399 features in the third dimension, where the last in the relative height of objects. Figure 3.7 shows an illustration of the three data sets stacked. Different examples from the data are extracted to illustrate all the information available.

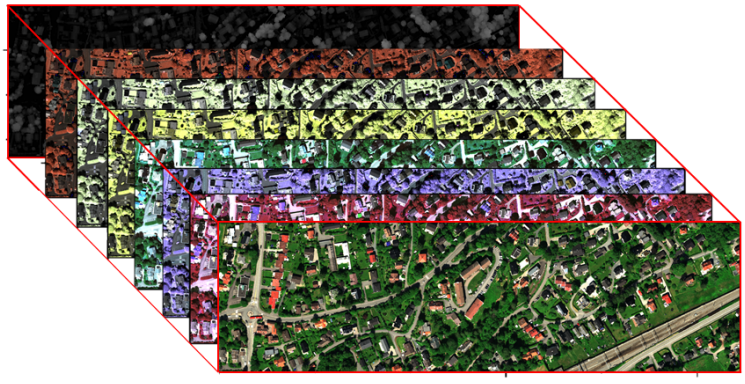


Figure 3.7: VNIR, SWIR and nDSM stacked into one data set. The rear image is nDSM, the rest are some of the wavelengths in the set.

3.6 Workflow

This study aims to use machine learning directly on hyperspectral and LiDAR data to localize and classify roof materials. It is possible to extract the interesting pixel and run a shallow machine learning algorithm on the data, but I want to retain spatial structure and interference with surrounding pixels for objects in the image. The workflow in this study is split up into two paths. The first path generates the ground truth, and the second uses machine learning to mimic the ground truth. Figure 3.8 shows the workflow where the lower path is where the generating of the ground truth happens. This path has many steps, and my goal is to bypass these steps by using semantic segmentation.

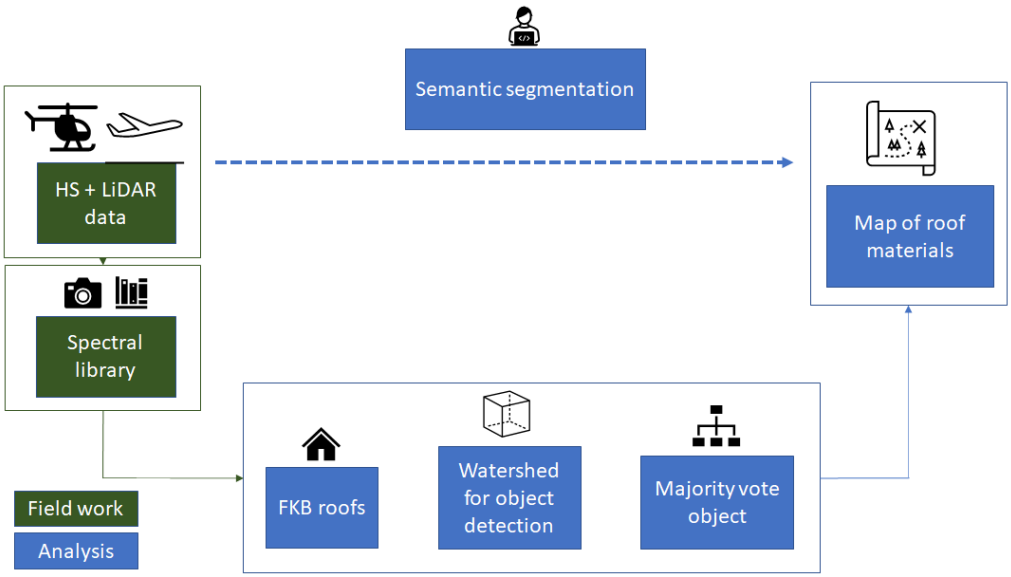


Figure 3.8: Workflow of the project for generating the ground truth and semantic segmentation.

3.7 Generate the Ground Truth

Supervised learning needs a ground truth to train and evaluate a model. There is no premade ground truth in this project, so it needs to be made from aerial photos and field work. It is often necessary with domain knowledge to make reliable ground truths.

This study uses field work and algorithms to create a ground truth. First, a field trip to the region of interest made the basis for the labels. Secondly, data from the field work and FKB buildings are combined to extract the buildings. Then using the watershed algorithm and "fill in"-technique a labeled ground truth was generated. The ground truth sets the foundation for how well the machine learning model can learn the data.

3.7.1 Field Work - Spectral Library

Mapping the materials are one of the most essential steps of the process. This step can require a lot of human capacity and is often the most expensive and tedious part of machine learning projects. In this project, field work from the area has documented all visible roofs and recognizable materials from the ground. There are no data for roofs that are not visible from the ground. All this information created a spectral library of the roof materials in the area. The spectral library is georeferenced, so every documented roof is easily implemented in GIS. Figure 3.9 shows the field work and documented materials.

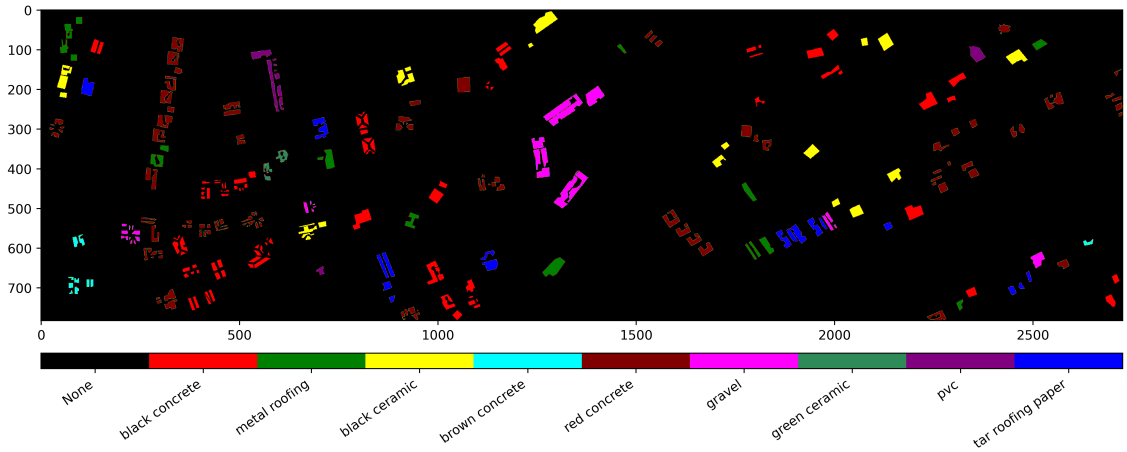


Figure 3.9: Field work of roof materials.

3.7.2 Extracting Rooftops with FKB

The spectral library gives a foundation for some roofs and their materials. Still, there are many roofs not documented in the library. FKB is then used to extract all the buildings in the area. This database has polygons of the buildings, and masking the polygons on the area rooftops are extracted. Every building not containing information from the field work will be marked as unknown. Figure 3.10 shows how FKB is used to extract every building including those with labels. The unknown roof materials are marked as gray.

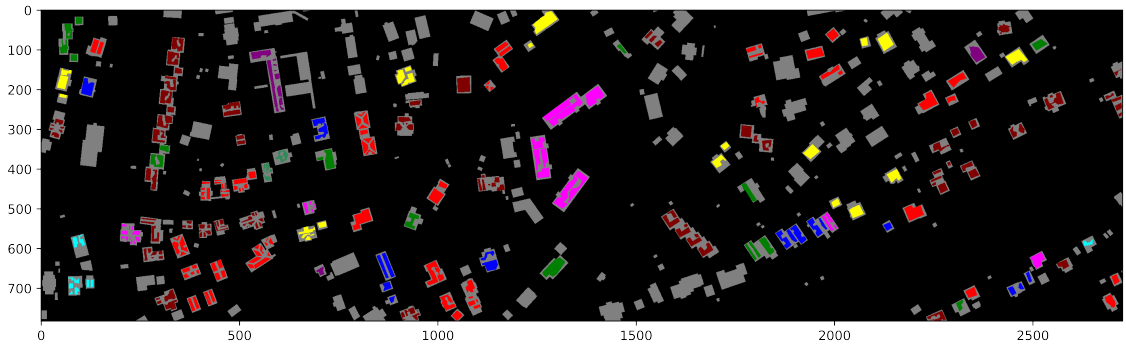


Figure 3.10: The area with data from FKB and spectral library. Gray roofs are marked as the Unknown class.

3.7.3 Object Detection - Watershed and Majority Vote

The watershed algorithm is used on the image to extract the rooftops as objects. When the objects get extracted they will be extracted as cohesive objects and can be manipulated as units. When applying the watershed algorithm to the image, the image needs to be converted to a grayscale image. This is easily done because the image has values from 0 to 11. The local minimum before watershedding the landscape $f(x)$ is set manually in these processes. This is to control how big an object can be. Many buildings are built as townhouses with roofs next to each other in the same construction style, but the roof materials can still be different. Therefore, the algorithm runs three times with three different thresholds for $f(x)$ to extract smaller and smaller buildings. For each time the algorithm runs, the found objects are removed from the original image and stored as temporary images. Then the three images merge as one. The threshold was set in descending order of 0.6, 0.5 and 0.3, of the maximum distance from a center to an edge.

Now that every rooftop is extracted from the image, I assume that a roof only contains one material, and the whole roof consists of that material. Every rooftop can be handled as an individual object after the watershed. Then by iterating all the objects, the material with the majority of the values in each object gets filled in. If there is a mix of labels in the object, and the unknown label is the majority of the object, the roof will be filled with the second dominant label. In this way, all available information from the field trip is getting used. Figure 3.11 shows how the watershed algorithm extracts objects and then merges the images.

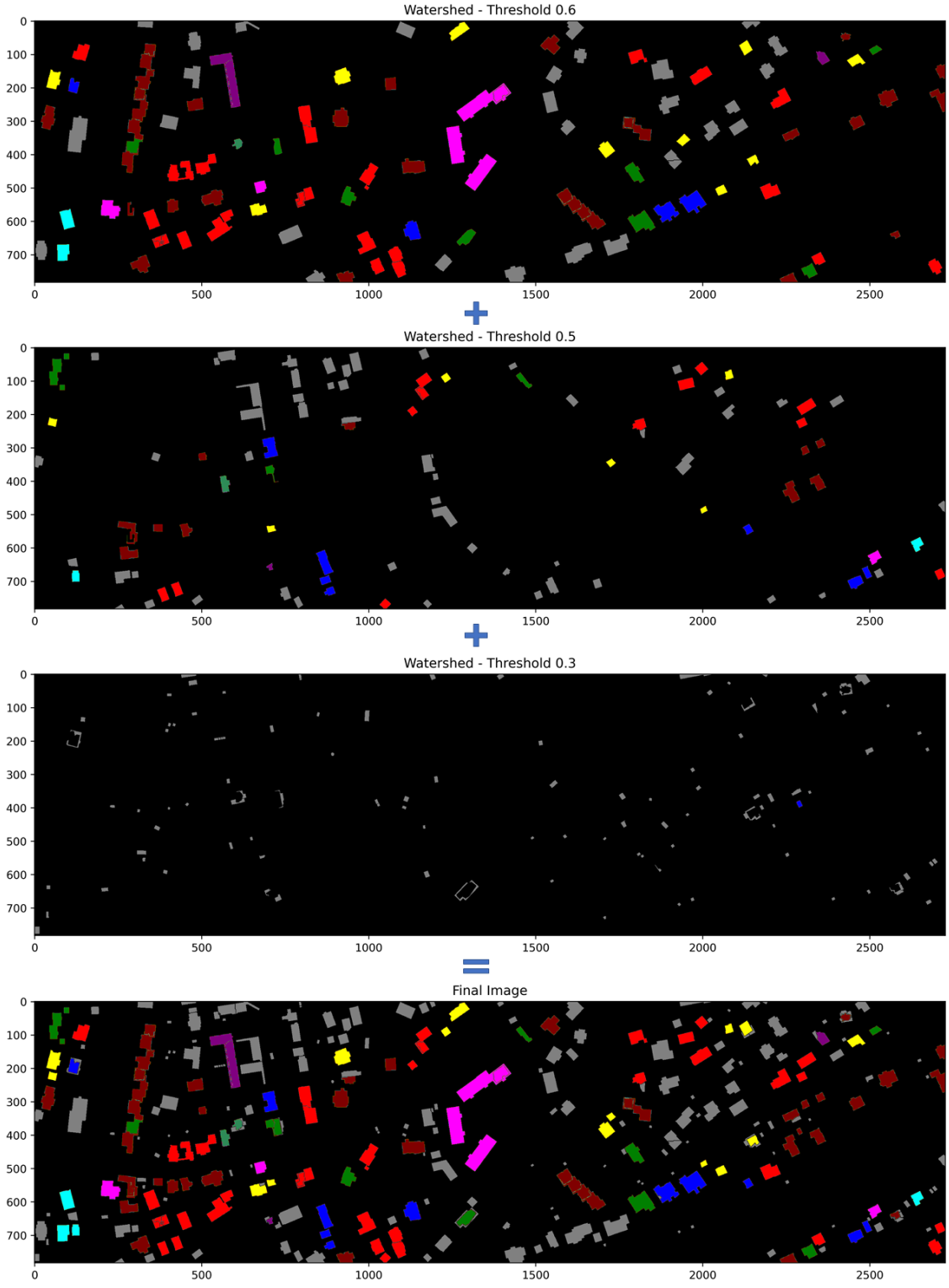


Figure 3.11: Watershed of the area. The watershed algorithm runs three times with descending thresholds to extract most of the roofs.

3.7.4 Change of Class - Gravel

When examining the images, one class was questionable. The class gravel did not concur with the RGB images and seems to be something else. On closer examination with GIS, satellite photo and Google Maps Street View to verify the gravel, there was doubt that it was gravel on the roofs. The roof looks more like red concrete and based on the similar surrounding buildings and their roof material, the gravel class got discarded and replaced with red concrete.

3.7.5 Finish Labeled Map

The result for generating the ground is shown in Figure 3.12. This map will be the fundament for further machine learning and semantic segmentation. The map consists of various roof materials and has almost all the roofs from FKB. Every roof from the field work is in use.

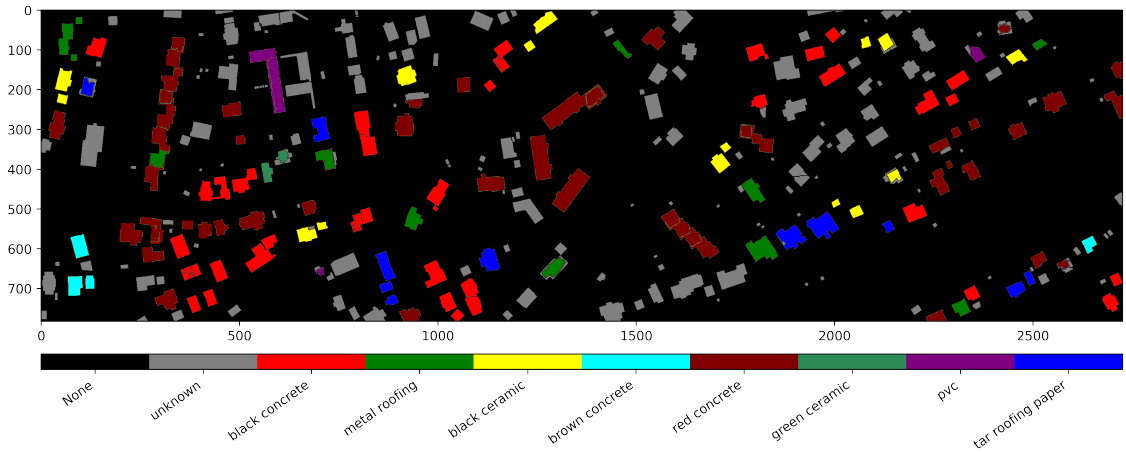


Figure 3.12: The ground truth for the study.

3.8 Data Distribution

The data comes in two sets: the fused hyperspectral and LiDAR data, and the ground truth. The data indexes are set so the ground truth corresponds to the same area in the hyperspectral and LiDAR data. The hyperspectral and LiDAR data will be referred to as the X, explanatory variable. The ground truth will be referred to as y, the response variable.

The distribution of the data sets can give an indicator of how machine learning models perform. The goal is to make a general model that can handle all events in the data sets. The X data's distribution needs to have so much variance that the model can distinguish between the different labels.

3.8.1 Separate the Data

When applying machine learning, the model must be generalized to solve the problem. This can be controlled by separating the data into train and test section and then evaluating the model on unseen test data. It is possible to see if the model can solve the underlying problem or if it has just learned the data by heart. The data is separated with 80% of the data in train, and 20% of the data for test. Figure 3.13 shows how the data is split up into four areas, two training and two test sets. The reason for this split is to have a variety of objects in each set. For instance, the railway is only on the eastern side of the map.

Figure 3.13: Separation of train and test set. This separation is in four section to vary the materials and objects in test set.

The images and labels need to be in a distinct format to use the data in a semantic segmentation model. In this study, the images are split up into images of 128x128 pixels. The images are separated with no overlapping areas. A fragment of the train set can be seen in Figure 3.14. The white lines indicate the borders in the images. In this case, 18 smaller images are put together as one image. The test data set is also sliced into smaller images.

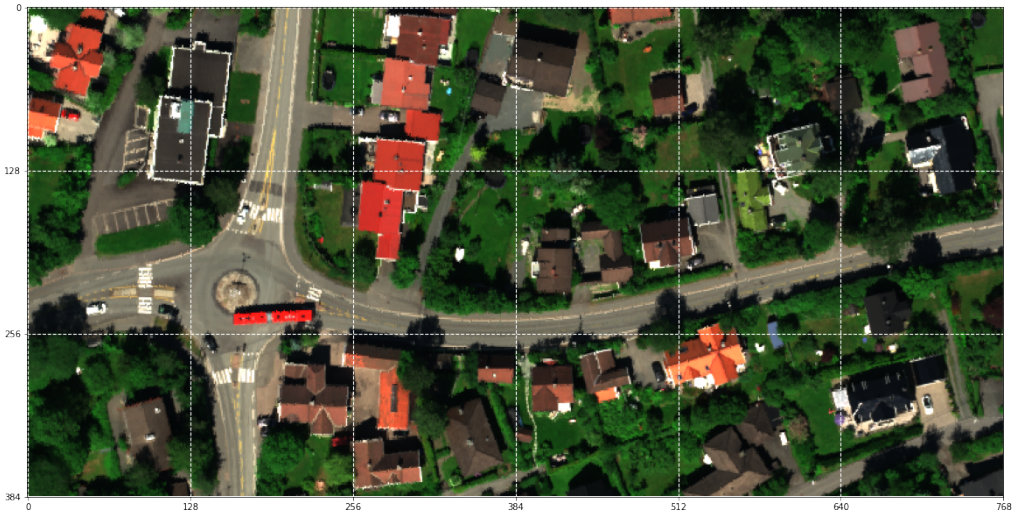


Figure 3.14: A fragment from the train set in RGB of 18 smaller images.

3.8.2 Class Distribution

This labeled map consists of 10 different classes. The majority class is the least interesting class in this study. It is interesting to see if the algorithm can detect interesting classes out of the not-so-interesting surroundings. Figure 3.15 shows the distribution of the classes. Here the imbalanced of class distribution apparent. On the right side is the same distribution in a logarithmic scale. That makes the minor classes more visible in the distribution.

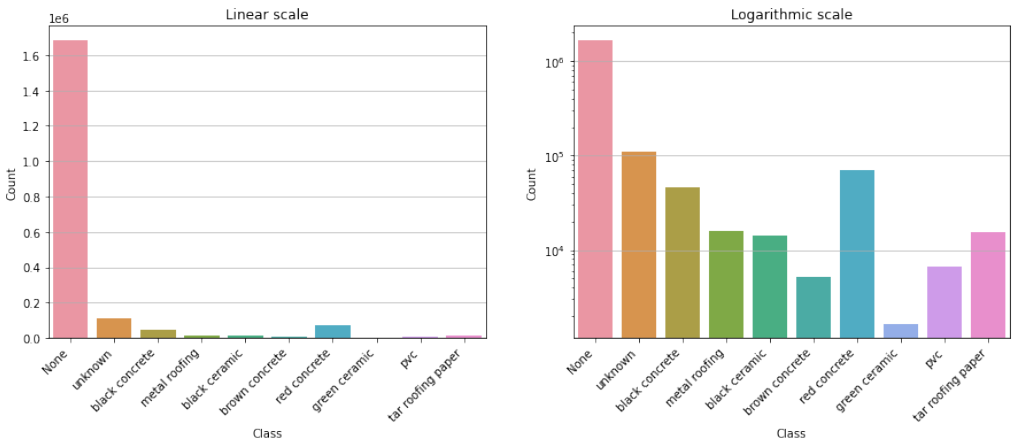


Figure 3.15: Distribution of the ground truth. Linear to the left and logarithmic to right.

3.8.3 Input Data Distribution

The distribution of some explanatory input variables is shown in Figure 3.16. The distribution is from the top left image in Figure 3.14. Figure 3.16 shows the probability distribution for the red, blue, green, nDSM and wavelength at 1500nm. The median of the values is the white dot in the rectangle, and the interquartile range is the rectangle.

X data is scaled from 0 to 1. This was not necessary to make the model optimize faster, but for simpler data handling in plots and manual checks. The data was originally a 16-bit data type ranging from -32767 to +32767. The height data is also scaled from 0 to 1, where 1 is 30 meters. The highest point found in the nDSM was 26.5 meters.

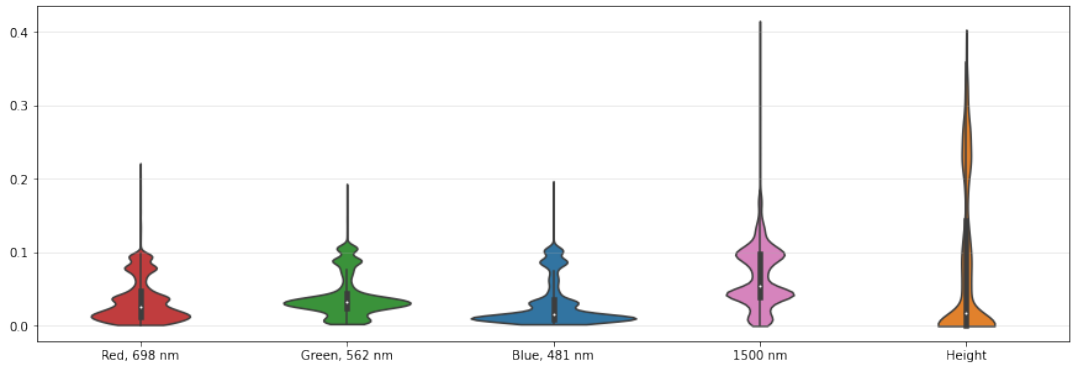


Figure 3.16: Violin plot for some of the X data. This plot shows the distribution of red, blue and green color channels. To the right is the height from the nDSM and distribution from the channel with 1500nm.

3.8.4 Train, Validation, Test Split

To evaluate the model, it is essential to have a training set and a validation set. Then a test set is used as a visual evaluation of the model. First, the data set is split up into train and test parts. The test parts are taken from two areas and are shown in Figure 3.13. The rest is used as training and validation.

For both areas, smaller non-overlapping images of 128x128 pixels are created. These images are will be inputs and outputs for the model. The area gets split up into 96 smaller images in the training set, and the test area has 24 images.

The validation set is then made by splitting up the train set into training and validation. Since the images are not overlapping, the model has not "seen" by the validation data. The split between training and validation is done so all classes are represented as equally as possible. This results in a distribution of train, validation and test split shown in Table 3.3. In each images there are over 16 thousand data points with 399 features. Resulting in more than 1.23 million data points in the training set and almost 1.90 million data points total. It is ideal to have the same

distribution in all sets, but that is a challenge when there are few images and it is random what class the images contain. Figure 3.17 shows the distribution of the classes in the train, validation and test sets. One disadvantage of the test set is that that are no samples with classes of *green ceramic* and *pvc*.

Table 3.3: Shape of train, validation and test set.

Data set name	**Shape of X data**	**Shape of y data**
Train set	(77, 128, 128, 399)	(77, 128, 128, 10)
Validation set	(19, 128, 128, 399)	(19, 128, 128, 10)
Test set	(22, 128, 128, 399)	(22, 128, 128, 10)

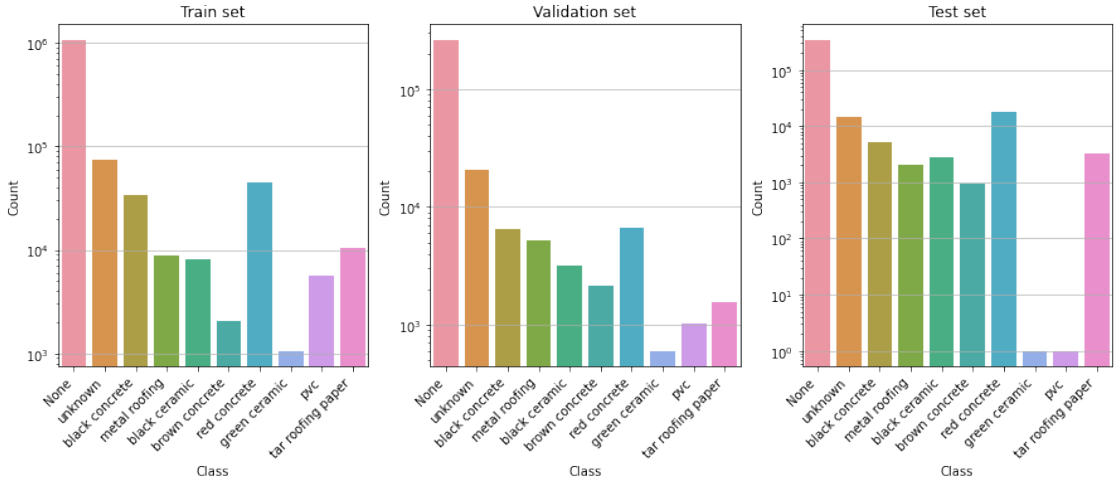


Figure 3.17: Distribution of y data when split in train, validation and test set.

The labeled data is redesigned to fit a onehot-style. This means that the values goes from scalar to a vector, where the index of the vector represents the class. This style ensures that the model does not believe a higher number is more important. At the same time, there are many machine learning frameworks made to work with onehot encoding. Table 3.3 shows this dimension of the y data set. Equation 17 shows a simple example of what the redesign to a onehot-style looks like. The dimensions will change slightly.

$$[0, 1, 2] \implies [[1, 0, 0], [0, 1, 0], [0, 0, 1]] \tag{17}$$

3.9 Semantic Segmentation on Ground Truth

The goal for semantic segmentation is to recognize the ground truth based on the input hyperspectral and LiDAR data. The ResNet was set as a backbone to a

U-net, which means that the U-net's encoder (downward path) is a ResNet. This combination will be called Res-U-Net to make it clear that it is a combination of the two. First, several configurations of U-net and Res-U-Net were tested. They got measured on how they performed with different losses, optimization strategies and weighting of the classes. These tests result in a model configuration that performs overall best and has the suiting characteristics.

3.9.1 Parameter Optimization

To find the best configuration for a model, several tests were preformed. The U-net architecture is kept as a fundament because of good performance in semantic segmentation. Beyond the architecture, the configurations tested were different loss functions, combinations of loss functions, a weighting of these combinations and weighting of classes. Additionally, model structures were tested with the different available ResNet configurations of backbone in the Semantic models module [93]. A vanilla U-net is also included in the test as a benchmark.

The tests were done with 10 models with randomly generated initialized weights. They train for 200 epochs. The measurement for all tests were Matthews Correlation Coefficient. The tests take around 3 to 6 hours on the Orion High Performing Cluster.

Firstly, the model structures are compared. The structures examined are a vanilla U-net and Res-U-Net with these bacbones: ResNet18, ResNet34, ResNet50, ResNet101 and ResNet152. Focal loss is used in all models. The models' name and amount of trainable parameters are shown in Table 3.4. The result from the test in Figure 3.18, and the lines are the mean of all tests and the colored area is the 95% confidence interval (CI). There is a clear trend that the Res-U-Nets perform better than vanilla U-net. Therefore, with a compromise between best performing and lowest complexity, the ResNet34 backbone is selected to be the model of choice in the later tests.

Table 3.4: Trainable parameters for the different models that were compared.

Model name	**Amount of trainable parameters**
U-net	2,218,843
ResNet18	15,574,346
ResNet34	25,682,506
ResNet50	33,757,258
ResNet101	52,749,386
ResNet152	68,393,034

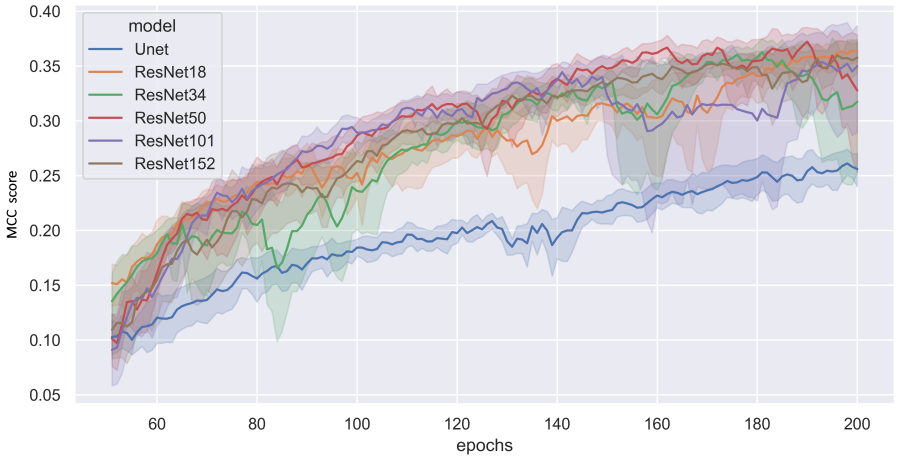


Figure 3.18: Comparison of different models and backbones. The tests span from 0 to 200 epochs and score in MCC.

Secondly, the different loss functions are examined on Res-U-net with ResNet34 as the backbone. These functions are tested in the same manner as the model test. Each loss function, and combination of loss functions, is built and trained 10 times on randomly generated initialized weights. The different loss functions are Categorical Crossentropy, one of the most used losses in multi class problem. Dice loss, Jaccard loss and Focal loss. Additionally, a combination of Focal loss, Dice and Jaccard was tried. Figure 3.19 shows the results from this test. The line is the mean and the colored areas is the 95% CI. A surprise here is that Focal loss did not perform well compared to the other losses. In combination with Jaccard or Dice, Focal loss yields a good score. The differences between these top scores are minor and can be random, so the loss chosen is Focal + Jaccard loss. I want to keep the characteristics of both of the losses. Focal loss is chosen to handle the extreme imbalance, and Jaccard loss is chosen to recognize objects.

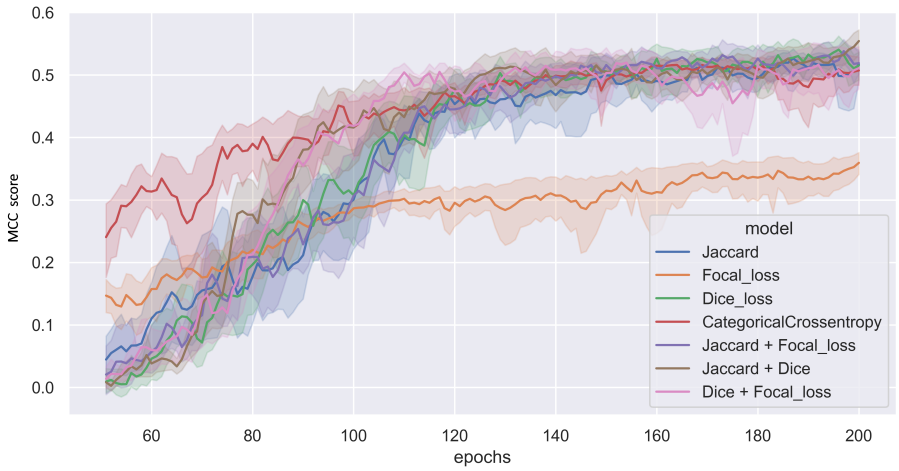


Figure 3.19: Comparison of different loss functions and combinations of these. The tests span from 0 to 200 epochs and score in MCC.

Thirdly, when two losses are combined, it is possible to weigh each loss. Focal loss will converge to a lower loss faster because it tolerates more misclassficiation, as seen in Figure 2.22. Jaccard on the other hand takes a longer time and needs to be closer to the truth before the loss converges. Optimistically, these two losses will fill each other out, and take advantage of both functions: handling extremely imbalanced data and spotting objects.

To check the best balance between these two losses, a test needs to preformed. The test is run with weight for Jaccard loss as the variable. The weight spans from 0 to 2. The test is done as the other tests. Random initialized weights 10 times for each model. The model that is used is the Res-U-Net with ResNet34 as the backbone. The mean and 95% CI is shown in Figure 3.20 in the same way as earlier. The results do not show any of the configurations to be better than the others, and losses are weighted equally to keep it simple.

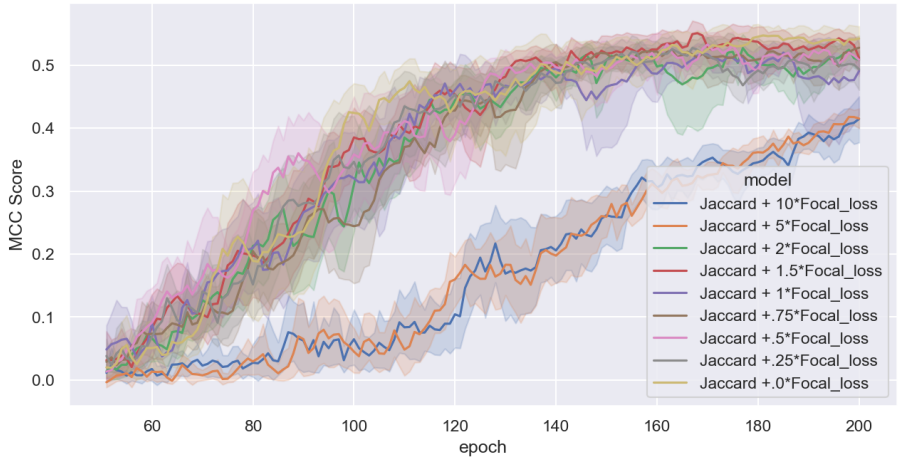


Figure 3.20: Comparison of different weighing of Jaccard and Focal loss. The tests span from 0 to 200 epochs and score in MCC.

The fourth optimization that can affect the performance of the model is the optimization strategy. Four kinds of optimization strategies are examined. These are: Adam, SGD, stochastic gradient descent, a classic gradient descent. RMSprop, that uses the root mean square propagation for a decaying average of partial gradients. Lastly, AdaMax, a variant of adam based on the infinity norm. All the algorithms optimize the loss function, and the goal is to find the global minimum as fast as possible. The results from this test are shown in Figure 3.21. All losses except SGD did almost similar, and I keep Adam as the optimization strategy.

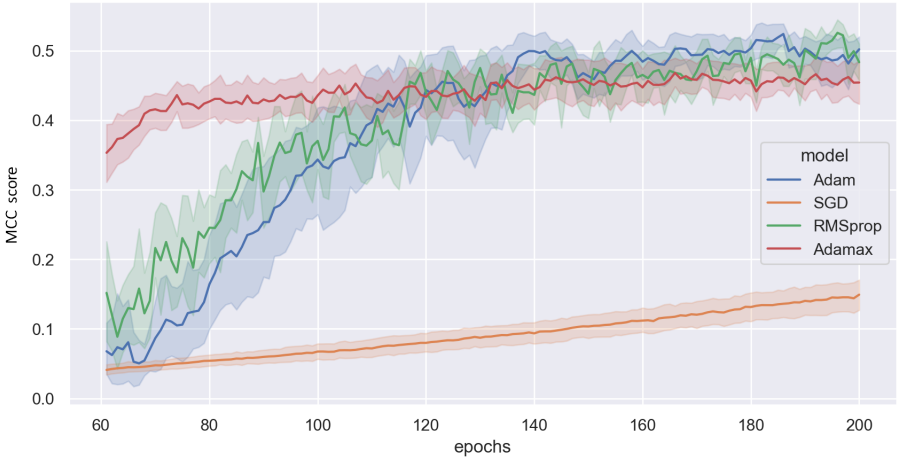


Figure 3.21: Comparison of different optimizers. The tests span from 0 to 200 epochs and score in MCC.

The last configuration that is easy to optimize is the class weighting in the Jaccard loss. In other words, the magnitude of each class in Jaccard loss. If a class has a high weighting, this will result in a greater loss if that class is not classified correctly. This can be done opposite, that a class weights lower than the others and cost less to classify wrongly. This is what I want to see if it affect the model. The hypothesis is weighting the None class lower will give the model more motivation to learn the differences between materials. The weighting for None is 0.1 and 0.01, the rest are 1. Then all the other classes have 10 and 100 times more influence on the loss function. Focal loss also has the option of weight classes, but the intention of using Focal loss is to regulate the extreme imbalance classes by itself, and weighting will not be done with this loss function. Results from this test are seen in Figure 3.22. There is no significant difference between weighting or not weighing. Therefore I try to keep it simple and weigh the classes equally.

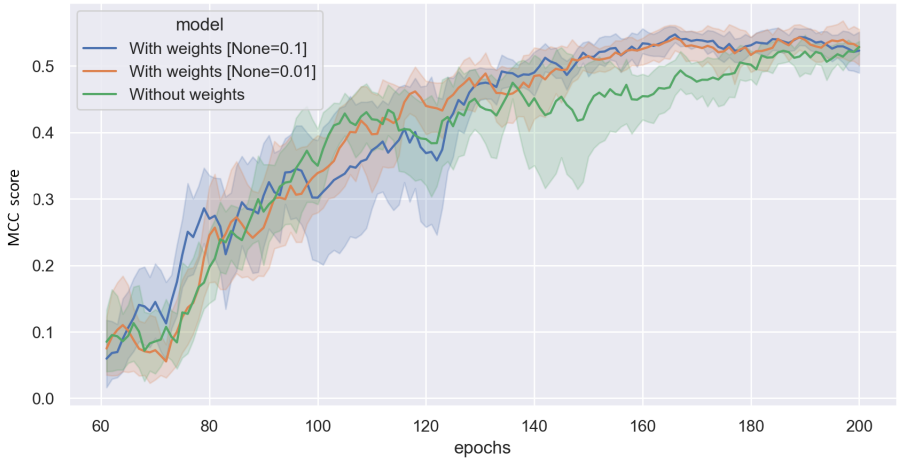


Figure 3.22: Comparison of the difference of weighting the None class. The tests span from 0 to 200 epochs and score in MCC.

Based on the optimization test, the final model used in this project is a fully connected convolution neural network with a combination of ResNet-34 and U-net architecture in a Res-U-Net. The loss function for the models is a combination of Jaccard Loss and Focal loss equally weighted. In Jaccard loss it is possible to weigh the classes, but all the classes are weighted the same based on results from Figure 3.22. In Focal loss, γ is set to the default value of 2. The optimization strategy for the model is Adam. The values for Adam are learning rate of 0.001, β_1 at 0.99 and β_2 at 0.999. The metric for the model is Matthews Correlation Coefficient and is used for validation of the model.

It is worth mentioning that these tests were done in series after each other, and there might be a combination that is not tested that could perform better. Testing all combinations is a demanding job, and there is some risk of overfitting this small

data set. Appendix VIII shows an approach to testing different combinations in figures H.1, H.2, H.3. There all model configurations are trained independent. This was done at a late stage in the study and did not affect the report.

3.9.2 Model Specifications

The model is then created based on the optimisation as a Res-U-Net with ResNet34 as the backbone (encoder). The input for the model is a set of 128x128 pixels with 399 dimensions. The data looks like this: $(Batch\_size, 128, 128, 399)$. That means that each image that goes through the model must have a height and width of 128 pixels, and 399 features. Data that goes in the model in a later state needs this shape. The output is $(Batch\_size, 128, 128, 10)$. This is the same width and height as the input, but now a vector with a length of 10. This vector represents each class that the model can predict. They return the prediction in a one-hot-encoder style. Each pixel has a vector with a length of 10, and the highest value indicates the answer. The output activation function for the model is softmax.

Figure 3.23 shows the final model. This is a U-net architecture with a ResNet 34 as backbone. The InputLayer is the initial input of the image. This is the input described above. Before each Activation is a BatchNormalization layer that normalizes the values. Then the data goes to the Activation layer. ReLU is used for every activation except the last one. This gives the model ability to learn nonlinear context. At the start, there is a MaxPooling2D layer that reduces the data. This is only done once. For the rest, the stride in the convolution is set so the data gets half the size. ZeroPadding sets a padding of zeros around the data to create a border and extracts features from the edges of the data in the same manner as the middle parts. The residual block can be seen as a combination of the Conv2D (convolution layer), BatchNormaliztion, Activation and ZeroPadding. There is a small skip connection to an Add block for each block that connects the residual. In Appendix VII, Figure G.1 shows this small skip connection of a residual block. The thickness of the building blocks in Figure 3.23 indicates the number of filters. The deeper the network goes, the thicker they become. In the deepest section, there are 512 filters per convolution layer.

The decoder path has UpSampling2D layers that will upsample the data. Then sent through a residual block. After each upsampling, the Concatenate layers receive information from the encoder path like the original U-net. Then in the end, a convolution layer gives the data the right output dimension, with 10 classes, and a Softmax Activation layer to normalize the last output to a probability distribution.

The model trains for 300 epochs, where the last epoch is used to predict and evaluate. The batch size is 32. 32 images will pass through before it changes the weights. This is a good number because it is important there is some variation of the materials before the weights change.

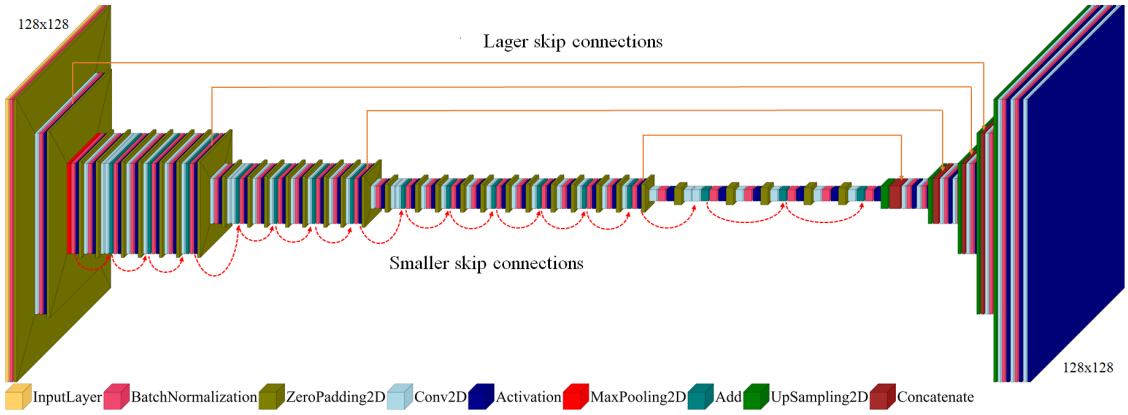


Figure 3.23: Final Res-U-Net model with ResNet34 as backbone. Generated with visualkeras package in Python.

3.9.3 Ensemble Models

An ensemble of 10 models is trained to give a more robust prediction. The model can struggle to find the right material of a rooftop, so they might collectively find a better result by putting more models together. 10 models are trained with random initial weights. When making a prediction the results are summed, and a majority vote selects the class the models agree upon. This is done by *arguments of the maxima* that takes out the index of maximum value of a vector.

3.9.4 Dimension Reduction

There is a belief that the model has trouble dealing with the high dimensional data and therefore be beneficial to perform a dimension reduction on the X data. With the model described above as a tester, three different dimension reductions were done on the data to see if some of them yielded better results. The dimension reductions performed were principal component analysis (PCA) with 10 components, extracting every 3rd wavelength (reducing the dimensions to $\frac{1}{3}$ of the size) and extracting groups of 10 wavelengths and reducing each group to the mean of the 10. None of these dimension reductions gave any better results seen in Appendix III in Figure C.1. Therefore, all the bands were kept in further analysis.

4 Results

4.1 Chapter Description

In this chapter, results from the image semantic segmentation are presented. The chapter initially starts with results from the training of 10 models. Their training results and computation time are documented and graphed here. Then a table presents the model performance on three different metrics: accuracy, F1 score weighted and Matthew Correlation Coefficient. The rest of this chapter is separated into two similar parts, where the first part shows results from one of the best models, and the second part shows results from the ensemblement of models. The results in the chapter are these three: a visual representation of the prediction of the test area, a confusion matrix of the different materials and a binary confusion matrix consisting of the labels roof and not roof. All score values and confusion matrices are based on a merge between validation and test set. The validation and test set are fused because all classes are not in the test set. The validation set is used to validate the model and it is not ideal to use it once again, but it is better to have all classes represented than missing two. The validation set contains 19 images and the test set has 24 images. When these two merge into 43 images, the model gets exposed to a vast amount of objects and diversity in the data, and the metric performance is hopefully more robust.

4.2 Training Results

The training is done for 300 epochs. After around 250 epochs the metric saturates and does not seem to get any better. 10 models are trained to obtain representative results due to randomly initially weights for each model. The model predicts results from the last epoch. On Google Colab's servers, computation takes around 1 to 3 seconds per epoch, so training 300 epochs take up to 10 minutes. The training time is halved to 1 second per epoch on the Orion cluster. The training was monitored in Matthew Correlation Coefficient in both train and validation sets. The validation score evaluates the performance of the model. A higher validation score for MCC indicates that the model performs better on the entire data set.

Figure 4.1 shows the training and validation MCC score for 10 models. The line is the mean of all models, and the colored area above and below is the 95% CI.

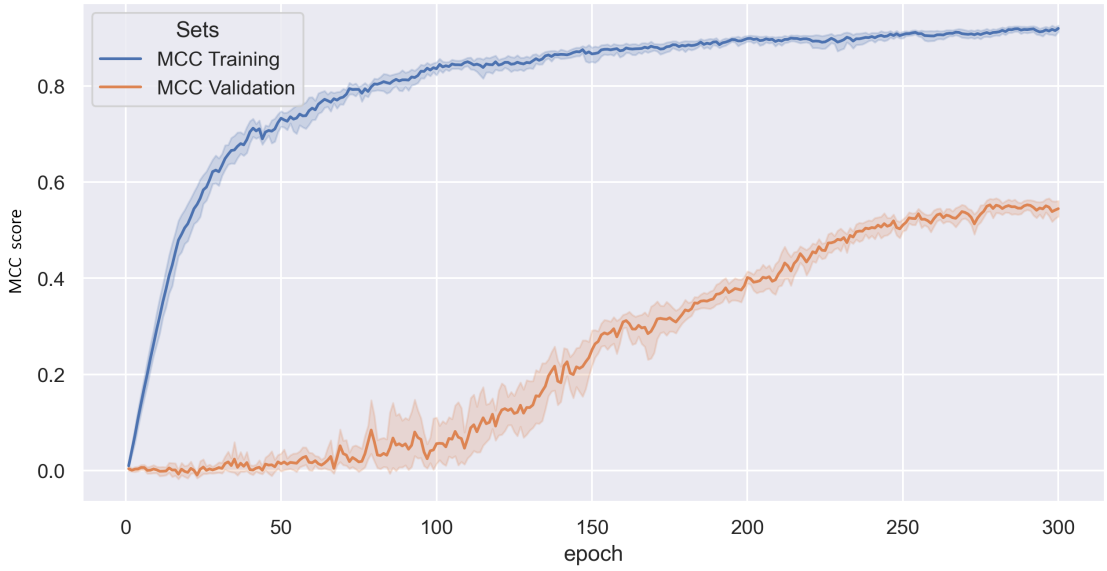


Figure 4.1: Training of the models. Model performance is measured in MCC for training and validation set. The line is the mean of the 10 models with associate standard deviation.

Figure 4.2 shows the training time per epoch on the Orion cluster. The line is the mean of 10 models' training times, and the colored area is the 95% CI. This shows that the training takes 0.7 to 0.8 seconds per epoch.

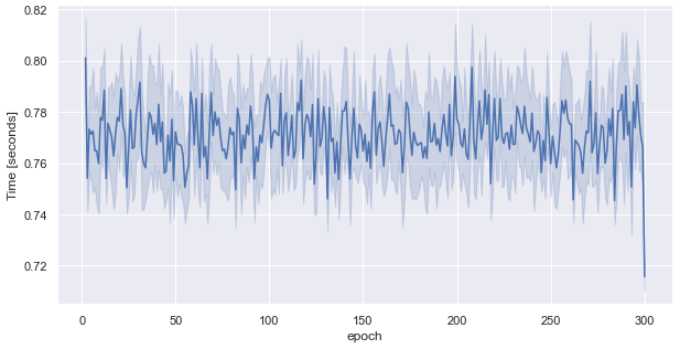


Figure 4.2: Training time per epoch over all the epochs. The blue line is the mean, and the colored area is the 95% CI.

Figure 4.3 shows the training time accumulated. The figure shows the training as almost linear with a small amount of variance. The line in the plot is the mean of the 10 training, and the variation is the blue color part around. This color area is the 95% CI of the training time data. From this chart, the training for the ten models took an average of 240 seconds, 4 minutes, to finish 300 epochs.

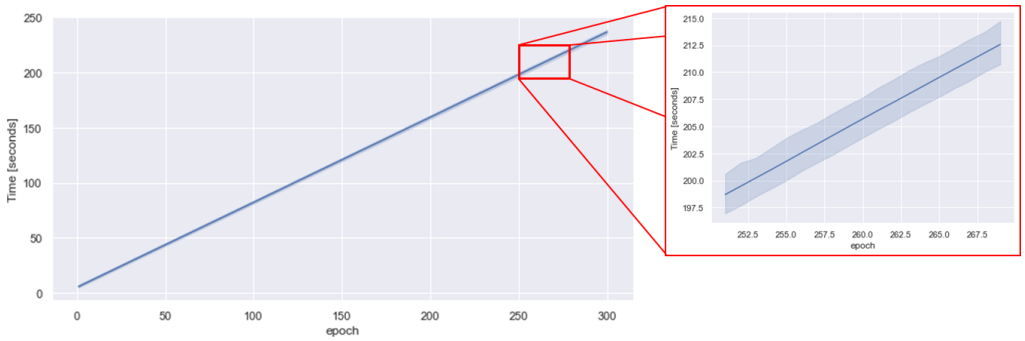


Figure 4.3: Accumulated training time for the models.

4.3 Loss Results

The loss results show the results from the two loss function. This is the optimization slope of the model, and the goal is to get as low loss value as possible. Figure 4.4 shows the different slopes of the losses. The blue line, MCC Validation is the metric evaluation of the model's performance. This is ideal to maximize.

The orange line is the total loss of Focal and Jaccard loss. The two losses are summed up in one, and that is what the network tries to optimize. In the start the variance of focal loss impacts the total loss, and when that settles the total loss follows the trend of Jaccard.

Figure 4.4: Slopes of the different loss functions on the validation data. The blue line is MCC which is used as a metric in the evaluation of the model performance.

4.4 Metric Results

The following results are based on data from the 10 models and their training. As the models start with random initiated weights, they all have different starting points. This can affect their converging pace and a model might end with a local minimum where it recognizes different characteristics and materials than the other models. Results in Table 4.1 show model performance with three different metrics. The score is the mean of ten predictions with associated standard derivation. F1 Score is weighted by the number of samples in the respectively classes. Table 4.2 shows the score for the binary problem of recognizing roofs with the same metrics as above.

Table 4.1: Metric score for models on the multiclass problem.

Metric name	**Score ± standard deviation**
Accuracy	0.903 ± 0.006
F1 Score weighted	0.896 ± 0.008
Matthews Correlation coefficient	0.579 ± 0.034

Table 4.2: Metric score for models on the binary problem.

Metric name	**Score ± standard deviation**
Accuracy	0.948 ± 0.005
F1 Score weighted	0.946 ± 0.007
Matthews Correlation coefficient	0.789 ± 0.031

4.5 Best Model Results

This chapter shows results from the model that performs the best of the ten trained models. This best model has an MCC score of 0.599 on the validation set for the multiclass problem. For the binary problem the best model got an MCC score of 0.794 on the same set.

4.5.1 Visual Results

Figure 4.5 shows the visual results of the test area. This area is split up into two different areas to get various materials in the visualization. The model predicts the smaller 128x128 pixels images, and then the predictions are set together as one larger image to show the whole area. The white lines sever the images. The colors represent the different materials, and black means there is not a roof. To the left in Figure 4.5 is an RGB image of the area, in the middle is the ground truth the model should mimic and to the right is the model prediction. Only the test set is used for this visualization.

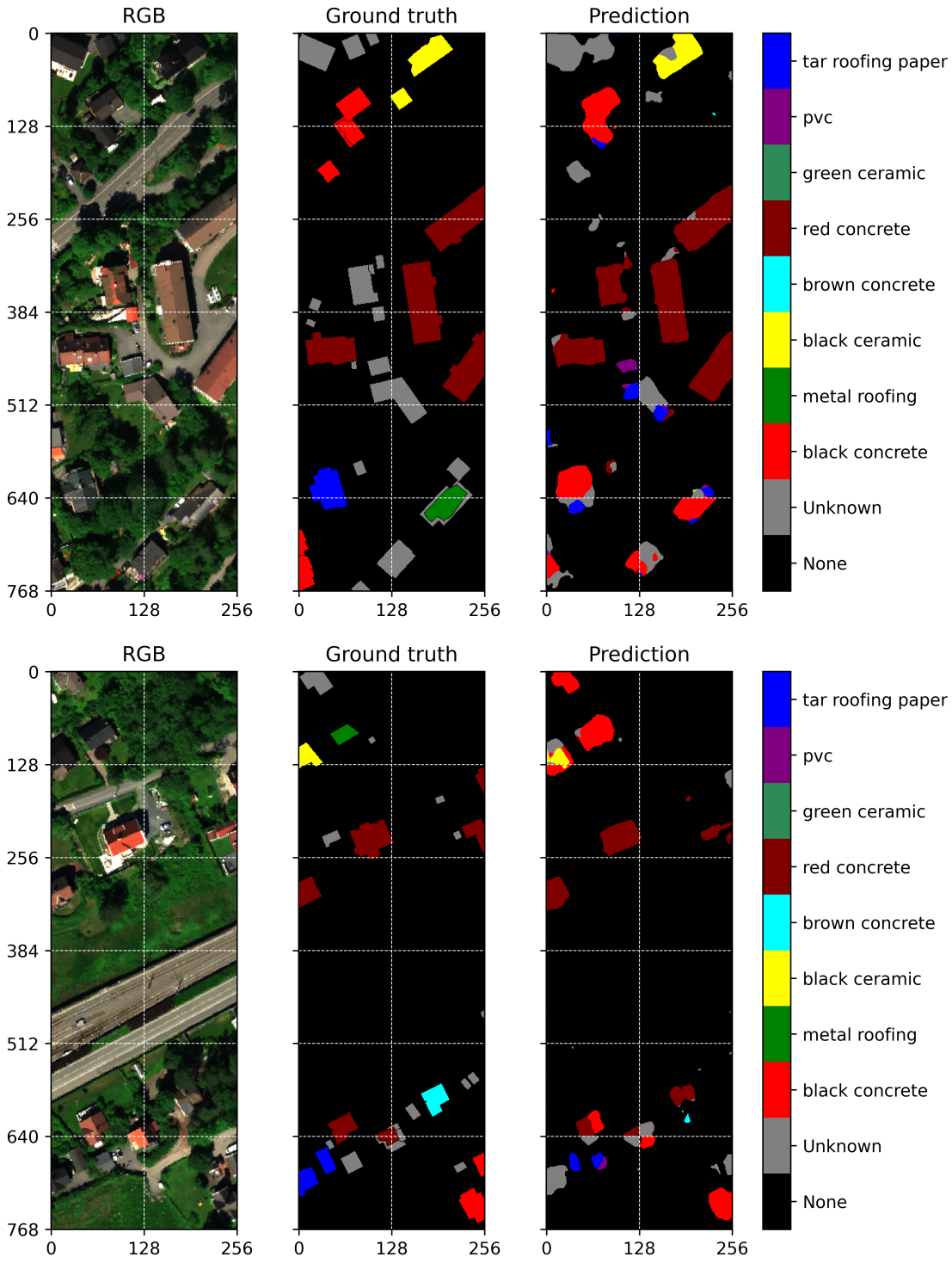


Figure 4.5: RGB, ground truth and prediction of the test area made by the best performing model.

4.5.2 Confusion Matrix

The confusion matrix shows true and predicted labels of the samples. The predicted labels are on the horizontal axis, and the true labels are on the vertical axis. The values are normalized on the true axis, meaning that the values sum up to 1 in the horizontal direction. Then it is possible to see the share of samples classified correctly or as another class. The colors correspond with the value of this share and get stronger with higher values. The results are seen in Figure 4.6.

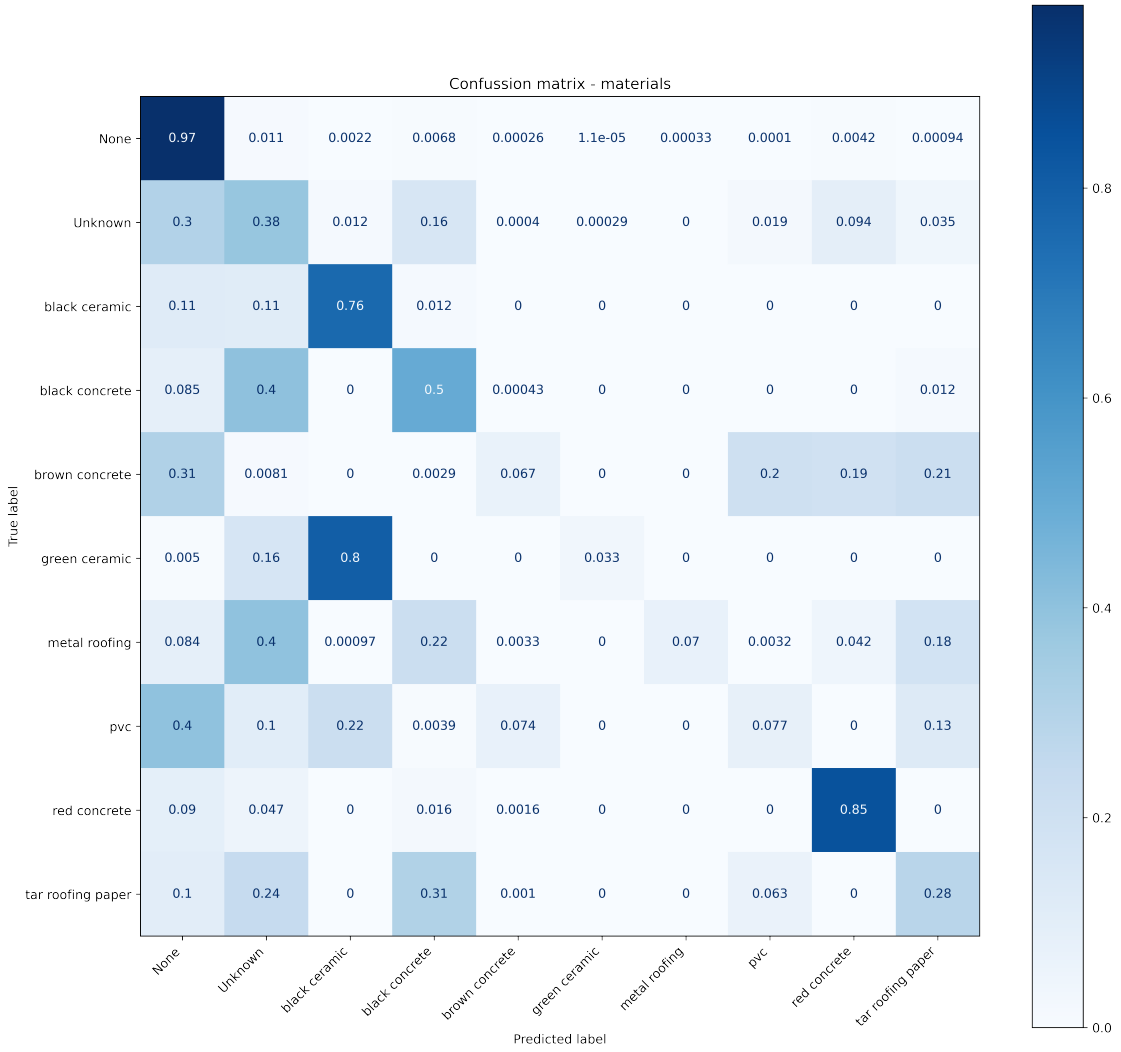


Figure 4.6: Confusion matrix for prediction of the best model. The values are normalized on the true labels, so they sum up to 1 on the horizontal axis.

4.5.3 Binary Results

Figure 4.7 shows the best model's binary classification results. The model predicts whether the samples are roof or not. The confusion matrix is also normalized on the true label. The number beneath in parenthesis is the amount of samples. In Appendix V Figure E.1 visualizes the test area results in binary format by the best model.

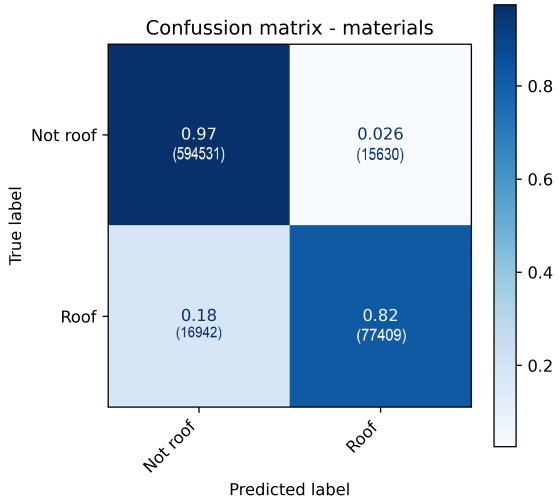


Figure 4.7: Confusion matrix for binary classification. The best model predicts if samples are roof or not roof.

4.6 Ensemble Model Results

Ensemble models can give robust results. This is done by taking all the 10 trained models and using the predictions in a majority vote. The most frequent answers is selected in the final prediction. The score is the same as above. The visual results and confusion matrix is a collaboration between the models.

4.6.1 Visual Results

Figure 4.8 shows a visualization of the results from the majority voting of the prediction from the 10 models. Every sample has 10 suggestions from the 10 models, and the suggestion with the most votes is kept.

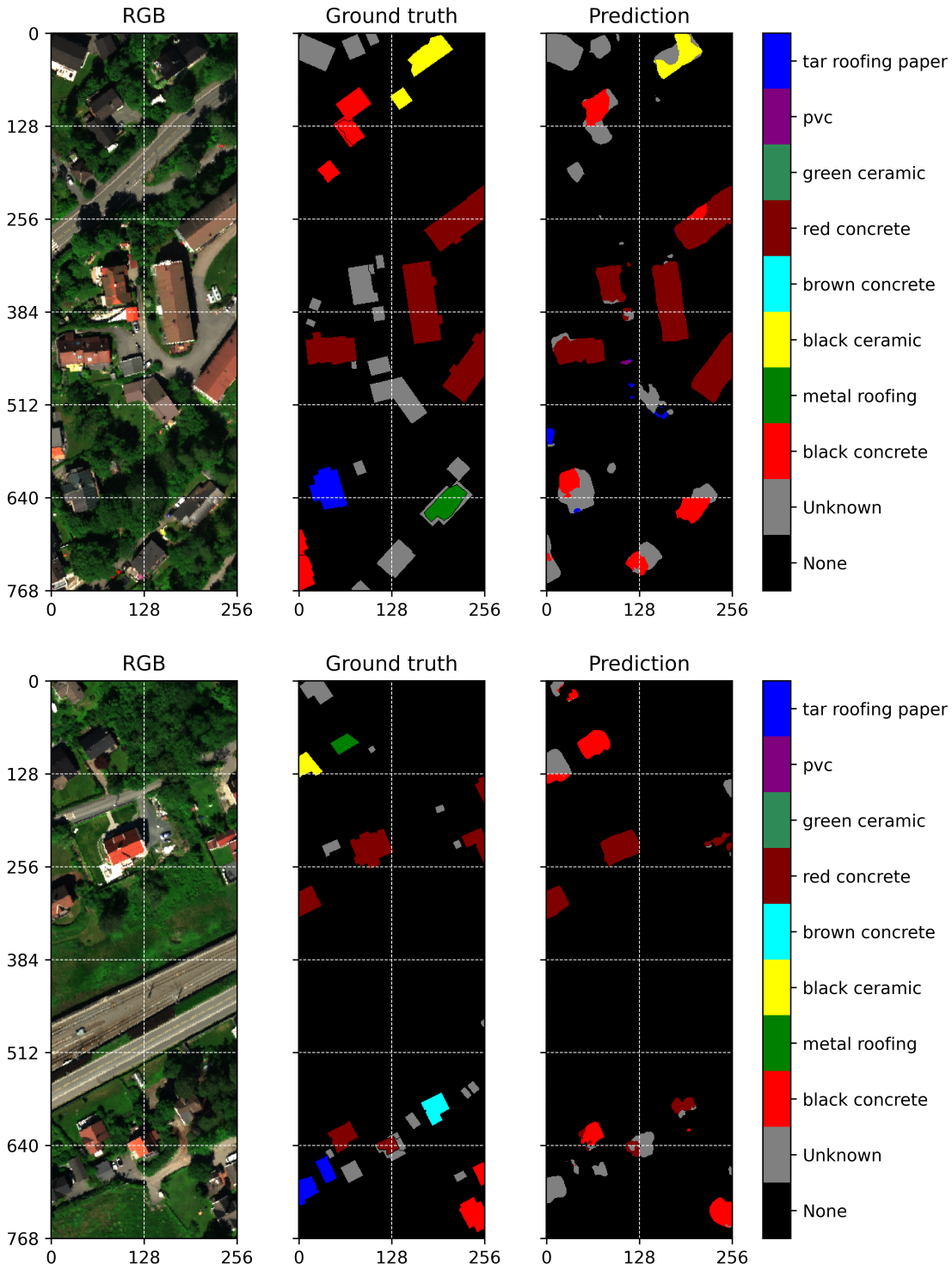


Figure 4.8: Majority vote on predictions from 10 models.

4.6.2 Confusion Matrix

Figure 4.9 shows the confusion matrix for the majority voting from the 10 models. This matrix is also normalized on true labels.

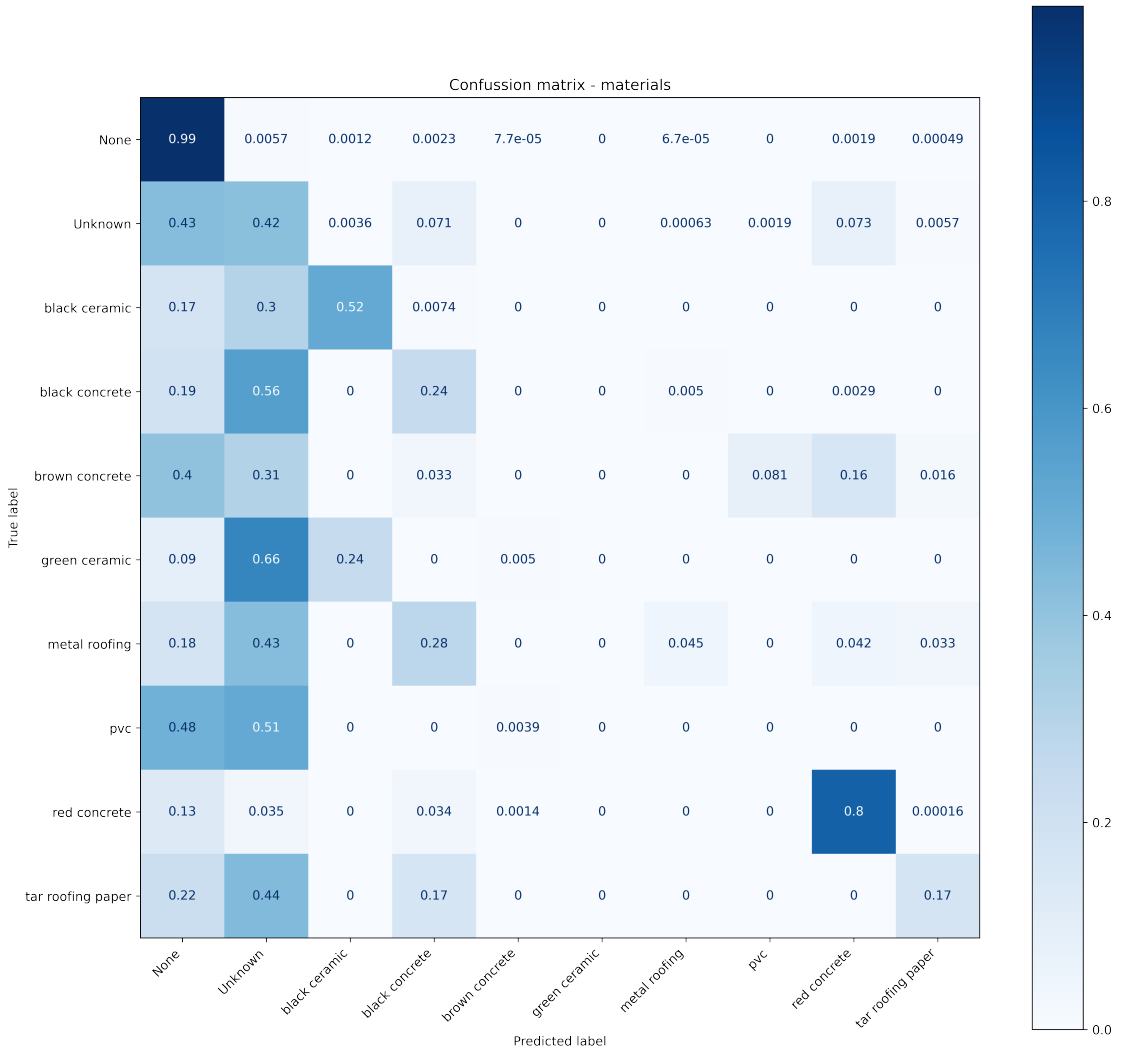


Figure 4.9: Confusion matrix from 10 models and normalized on the true labels.

4.6.3 Binary Results

Binary confusion matrix for the ensemblement of the models. Figure 4.10 shows the binary confusion matrix in the same fashion as the others, and the number beneath in parenthesis is the number of samples.

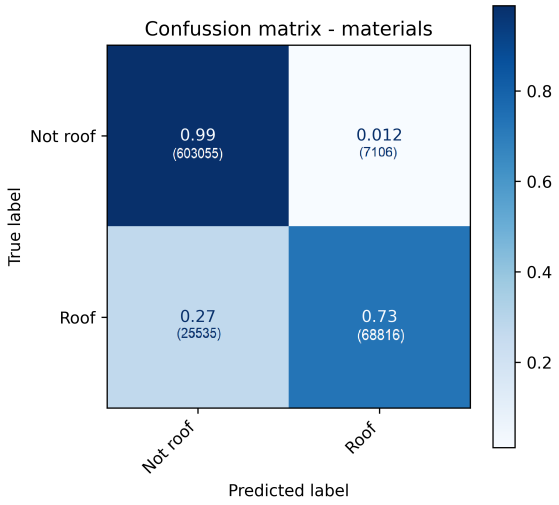


Figure 4.10: Binary confusion matrix for roof or not roof with the majority vote of all models.

4.7 Dimension Reduction Results

Results for dimension reduction can be seen in Appendix IV figures D.1, D.2 and D.3. These figures show results from PCA with 10 components, results from extraction of every third wavelength, and results from a batch-wise extraction of the mean of ten wavelengths. All three variants have nDSM added as the last dimension.

5 Discussion

5.1 Chapter Description

This chapter discusses central aspects of data quality, model performance, comparison of related work and future studies. In the data quality subchapter, the addressed topics are quality of the raw data before the analysis, challenges with the aspect that the hyperspectral sensors only capture surface materials, quality of the generated ground truth, correctness of FKB and the ground truth, and the advantages of this data set. The model performance subchapter disputes the model's performance, results and if the model obtains desirable characteristics. Then some related work is compared against the results and the chapter ends with some recommendations and suggestions for future work.

5.2 Data Quality

The data from the hyperspectral and LiDAR has generally high quality. The data has a wide range of usable bands, and the overlap from VNIR to SWIR is smooth. The data presents few unwanted effects since it is already converted to reflectance with an atmospheric correction before the analysis. Even though the data is of high quality, it is not perfect. Some bands have noise that needs to be removed. For instance, some bands have negative values that should not be possible. This can be sensor calibration or other factors that the producers of the scans are responsible for. Anyhow, the easiest way to deal with these small issues is to eradicate these data points. By doing so, there is some information lost in the process that can affect the analysis, but hopefully the data is good enough for this type of analysis.

There are some aspects of the data set that cause difficulty. First of all, the hyperspectral sensors only capture the surface materials directly exposed to the sensor on the airplane. This can cause some unwanted incidences, for instance where trees are growing and hanging over buildings. The vegetation covers the roof material making it impossible to observe it. When it is impossible to observe the roof, it is challenging to evaluate if values from this area are purely from the tree or if some roof material shins through the vegetation. This leads to another problem, in the ground truth these covering obstacles are marked as roofs. These misinterpretations can cause unwanted effects in the learning process. If a tree is marked as a roof in the ground truth, the machine learning algorithm learns that this tree might be a roof, and future similar trees can be marked as roofs. Such trees can be seen in Figure 5.2 where the roofs are underneath trees. Another issue directly caused by the data is the slicing of the images. Since the algorithm learns

from smaller images, there is not much room for many objects in one image. In some lucky cases a roof is in the middle of the image, but in most cases the roof is cut off at one or the other end. This counts for all the other elements in the image as well. For instance, the road is marked as None and is not interesting in the seek for roof materials, but for the algorithm is it equally important to mark this road as None as marking a roof as red concrete. Cutting out parts of almost any types of objects by slicing the area into smaller images, the model needs to learn objects from a fragment of the original shape of the objects. The last unused potential of the LiDAR data is the utilization of the point cloud. I mentioned that the passive sensors from hyperspectral cameras only capture the surface. The active LiDAR sensor makes it possible to see several returns from penetrable objects like trees. Only the first returns from the points cloud are used in the nDSM. This is some lost potential and information, and the analysis can benefit by exploiting such potential. These challenges are not impossible to deal with, but can reduce the results and outcome of the analysis.

Despite the challenges in the data set, it still has many good qualities. There is a wide variety of roof materials in a relatively small area. The area contains characteristics of a suburb area, like smaller and larger roads, houses, railways and vegetation. Most of the buildings stand alone and are easy to isolate. They are also built in all directions, giving the machine learning model the variance in spatial localization it might need to generalize a typical building shape. The model is also exposed to different light conditions on the roofs. Since the sun shines from the south, it creates shadows on the northern side of the roofs. This causes the material to reflect somewhat lower values and allows the model to learn a material in two different light conditions.

The label data contains some challenges too. The classes are extreme imbalances, especially the None and unknown labels take the majority of the samples. This imbalance is not great for a general analysis, but in this case it is the genuine problem the machine learning algorithm will solve. The data is high-dimensional and contains a lot of information. There might that there is not enough labels to cover the variance good enough. The split between training, validation and test sets is not equally balanced, but is a result of real world imbalanced areas.

The training and validation set have all classes represented, but the test set misses two classes. This is a weakness of the whole data set and can give misleading model performance results. It is therefore impossible to visual examine the model recognizes these missing classes. This problem appears because the area is limited and the sets are split up in distinct places. The validation set is carefully extracted from the train set to contain all class samples. But, it is impossible to put the validation set together to visualize a greater area as it is done with the test set. When the areas are split into distinct parts and the content of the images is random, it requires more knowledge of the area before splitting to get the wanted equally distribution in all sets. When the model gets evaluated, validation and test set

are merged to get a representation of all classes. This is a weakness because the validation set is already used on the model to select the best performing, but still better than missing out on some classes by only using the test set.

Two other complications in the labeled map are FKB and the watershed algorithm. FKB is a database manually updated by the responsible authority. These updates might not always be correct at the given time and can in some cases be wrong. Figure 5.1 shows an area where the ground truth shows there should be a building, but the building is not there when looking at the RGB image. In this data set FBK from 2019 is used on photos from 2021. When the FBK is two years from the actual data it is inevitable that there will be some errors.

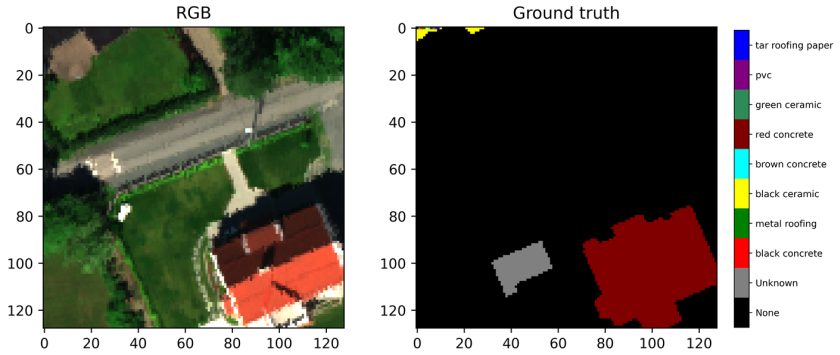


Figure 5.1: Mismatch between ground truth and the airborne photos. Extracted from the middle images of Figure 4.5.

The watershed algorithm extracts data based on its length from the center to the edge and can be misleading in some cases. For instance, if an object is so small the $f(x)$ gets so low that the threshold for water flooding is above the highest point of $f(x)$. Then the object disappears from the data and I lose information. Another example is the watershed algorithm struggles when two roofs are adjoining with different materials. Then the border between these two roofs is hard to identify by the algorithm and the exact differentiation is blurry. This border is distinct when looking at the orthophoto, resulting in the model learning something that is not true.

The unknown class can be challenging. This class includes every roof that is not documented which can be materials belonging to another class. Having the same types of values resulting in two different classes is not optimal. Anyhow, the labeled ground truth has some flaws, but is of high enough quality for semantic segmentation.

5.3 Model Performance

The training procedure for models as seen in Figure 4.1 shows that the models quickly comprehend the training set, but take some time to generalize to the validation set. After around 250 epochs, metrics for validation data saturate and the models learn much slower. This might indicates that the models have reached their potential on the data set. A test was done for training the models for over 500 epochs the metric did not get any better results, but a closer visual examination must be done to verify this.

The training time of the models is stable. For each epoch a network uses around 0.78 seconds, where an epoch consists of feeding the network 77 shuffled images in 3 batches. The first two batches have 32 images, and the last has the remaining. After each batch the model changes the weights based on the optimization strategy and loss functions. At the end of the epoch the model does a prediction on the validation set, and returns the evaluation for the metrics. There are over 25 million adjustable weights in one model, and the training set contains around 1.5 million data points. All of this is done in under a second, and it confirms that the high capacity hardware plays an important role in making the learning process effective.

Figure 3.15 shows the distribution of the data is 85.6% None and 14.4% roofs. If the models constantly guess None for all samples, the accuracy will yield 0.856, but F1 and MCC will yield 0. The scores from Table 4.1 clearly show that the models guess more than random or other classes than None. The MCC score of 0.579 cannot confirm if it is a great score when there is nothing to compare against, but the model clearly learns some the data structure. The low standard derivation indicates that the models learn generally stable. All of this indicates that this model architecture is well suited and robust for this type of data.

Figure 4.4 shows the slopes of the different loss functions. In addition to the losses in the plot, MCC is included to evaluate the model performance. MCC gives an indication of when the model starts to learn the data structure. I interpret from the figure that the model first seems to learn the Focal loss problem. Thus, when the Focal loss decreases slowly, the Jaccard loss keeps decreasing while the MCC rises and the models learn. It seems like the model manages to utilize the best of the two losses. Firstly, the extremely imbalanced characteristic of the Focal loss is solved, then the IoU object detection for Jaccard makes the model learn the correct structures of the roofs. Keep in mind that this is just a hypothesis and an approach to explaining a complex model. The complexity of the model can be challenging to explain if it does tasks that require justification.

The visual results in Figure 4.5 show that the best model gives a good approximation to the ground truth. The buildings are distinct and clearly recognized as objects with their spatial form. The results show that most of the roofs are marked as just one material for the whole roof. Especially the class *red concrete* has solid forms on the entire buildings. Some roofs have a mixed set of materials, and where

the images are sliced on the white lines many of the roofs change material. These results indicate that the models are good at finding the spatial structure of a roof, but not so good at finding the material. A conceivable explanation can be that there is an inadequate amount of data of the minor classes. The models perform well in finding red concrete, and that can be because red concrete is one of the major classes in the data set. When there are many data points in the class, there is a much higher chance the samples are correct and of high quality. Obstacles between the roof and sensors are reviling in the visual results. Vegetation is above the roofs in several of the images, and by examine the results the sensors probably only see the vegetation. Figure 5.2 shows the effect of dense overhanging vegetation. According to the ground truth there is a building beneath, but a challenge to detect.

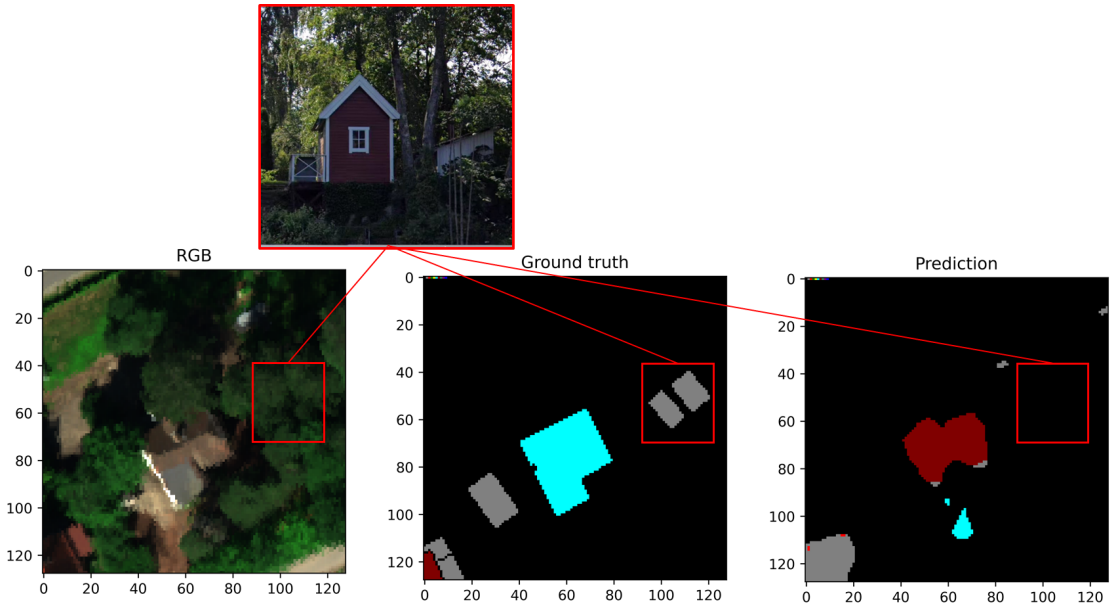


Figure 5.2: Roof under vegetation that is challenging to spot by the model and the sensors. Extracted from the lower images of Figure 4.5.

The confusion matrix in Figure 4.6 shows the relationship between true and predicted classes. It is normalized on the true axis, meaning that it is possible to see the percent share of what a class gets classified as. For instance, red concrete with the values seen in Table 5.3 of share predicted samples.

Table 5.1: Share of predicting for red concrete samples from confusion matrix in Figure 4.6.

Red concrete	Brown concrete	Black concrete	Unknown	None
0.85	0.0016	0.016	0.047	0.09

This table shows that 85% of the red concrete samples are classified as red concrete. The second frequent guess is that there is not a roof where it is red concrete, and 4.7% is classified as unknown. This indicates that the model has learned the spectral signature of brown concrete and is able to recognize it on unseen data. For the minor classes the model performs poorly. Brown concrete, green ceramic, metal roofing and pvc are rarely classified and are mostly categorized as other materials. This can be because the model does not have the capacity and characteristics learn this complex material. A more convincing reason can be that the data quality of these classes is poor and the model struggles to learn them. However, based on the confusion matrix the model indicates to learning some materials, but others are not recognizable.

The binary confusion matrix substantiates the indication that the model performing well in recognizing roofs. The model might struggle to distinguish between the material of the roof, but it successfully localizes many of roofs. By studying the binary confusion matrix I learned that the model finds 82% of the roof samples. The share that is marked as false negative, not roof but actually is, is much higher than the true positive. This demonstrates that the model will rarely predict non-existing roofs. For a model like this, the goal is to localize and classify roofs. For the use case of this model where it operates alone. I think it is better to have a low rate of type I, false positive, because the model is reasonably sure it is a building with a roof. It is better to miss one building instead of new buildings suddenly spawning. This is dependent on the use case for the data. If the model is used as an initial search for roofs and materials and is regulated afterwords. It might be better to have a higher type II, so it finds more roofs. One important notice of the binary confusion matrix is that the matrix is normalized on the true axis. By looking at the same figure the samples beneath tell that there is almost the same amount of misclassification in both classes, but the share is much lower in the non roof class.

The goal of using all the models in an ensemblement is to get more robust results. The majority voting is meant to remove abnormal behavior in the belief that the majority knows the best. This seems not to be the case for this model and data set. The ensemble models performed worse than the best model. Looking at the confusion matrix the models do not recognize the minor classes. Every class has a lower score except the None class. This outcome is reflected in the visual representation of the area. On the other hand, there are fewer false positives of roofs. As seen in the confusion matrix the share rightly classified None class is higher, and the models classify less of the other classes when it is None. I conclude that an ensemblement of models is more conservative and cautious in finding roofs, but performs lesser.

Saying that the ensemblement is lesser than the best model indicates that the metric is well suited for the problem. As the models are measured in Matthews Correlation Coefficient, the best performing model's metric looks better in the confusion matrix and visual examination is a great indication that a higher score equals

a better model. Therefore, the best model will be the one with the highest MCC validation score.

Dimension reduced data did not give the model better performance. This confirms that when reducing the dimension some information is lost in the process. One thing to notice is that the models did not perform significantly less with the dimension reduced. The computation time is considerably lower when there is less dimension. So it is possible to get almost the same results using less time and resources. Using a less complex model with reduced data it can be possible to get the same results, but that needs to be tested.

5.4 Compared to Similar Works

There are few similar studies that use hyperspectral cameras combined with LiDAR to classify roof material. Expect for the shallower models used by S. van der Linden et al. [6], and F. Trevisiol et al. [8] with their support vector machine and preprocesseing to extract the roof before classification. Many have tried to recognize and extract just the building structure from aerial photos. For instance, M. Bassier et al. managed to classify buildings with an accuracy around 87% based on random forest exposed to a point cloud [94], and M. Guo et al. [95] used a multiloss U-net architecture to extract buildings and showed good performance on the Aerial Imagery data set for roof segmentation. This is probably one of the most similar study to this thesis. It is impossible to directly compare metric results with other studies if they are not on the same data set. But, it gives a supportive indication that this type of model shows good performance on the same problems this study examines. The losses M. Guo et al. tried on their model were categorical cross entropy in combination with dice loss, and the probabilistic Rand index (PRI) combined with Jaccard loss. That indicates that the losses I have selected have been chosen by other scientists and are suitable for the task. In A. Kuras review on classification algorithms [28], there are many CNN classifiers that use fused hyperspectral and LiDAR. These models find the differences between larger objects like roads, vegetation and buildings. Many of these classifiers are exposed to the same data set: *2013 IEEE GRSS DF Contest data set* [96, 97, 98, 99]. This data consists of background, different kinds of vegetation's health conditions and types, different kinds of roads, parking lots, tennis courts and even running tracks. This requires the models to learn a vast variance of elements. I believe the model might do at least as well with a lower spectral resolution, like an RGB image, because of the great differences in the objectives. I believe the interesting aspect of hyperspectral images is to tell the difference between materials that look similar and are hard to distinguish. Such materials can be roofs, they are hard to classify because many look almost identical. As far as I can see, few, or maybe none papers that classify roof materials with semantic segmentation.

The most comparable works to consider are earlier master's thesis using similar

data from the area. One of the challenges R. Senchuri found during his study was an inconsistent pixel-wise prediction over a larger area. A prediction of an area can contain all the different classes without directly relation with the neighboring pixels. He recommended implementing advanced classification and segmentation algorithms, which has been done in my thesis. The results from my study indicate that these segmentation algorithms reduce this inconsistent salt-pepper effect from shallower machine learning algorithms. This exact problem I experienced myself on this data set. Appendix VI Figure F.1 shows the difference between a random forest classifier and the segmentation model. Obviously, the last model gives desirable characteristics compared to the random forest. The metrics for the random forest classifier are equally good, and for MCC even better compared to the segmentation model. The visual result tells another story where the segmentation model performs much better. In another master's thesis by A. Primstad and Å. Stemme [100], they write about pixel-based area classification of urban environment with deep learning and hyperspectral images. They also recommend applying segmentation models as future work for finding objects in an image.

This study has explored new methods of analyzing fused hyperspectral images and LiDAR data. Previous experiences from other master's thesis and papers have been used in the choice of algorithm structures. Using these experiences on a specific problem, finding roof materials, the study shows the practicality and versatility of detail mapping using semantic segmentation on hyperspectral and LiDAR data.

5.5 Future Work

The results from this study show that semantic segmentation of roof materials is promising. The results can be used to demonstrate the usage of the algorithm on the data, but are far from constant enough to industrialize and automatically map areas without being authorized. However, there are several opportunities for the data foundation and model architecture for future research that can yield better results.

The immediate improvement is data collection. The more data the models are exposed to, the better and more generalized they typically get. This project has an enormous amount of airborne data, so the limitation lies in the labeled data. I recommend documenting more roof materials for a larger ground truth for future research. It might be a good idea to engage experts in the field of roof materials to increase the data quality and ensure the correctness of the ground truth. Based on the applications of such information collection and the value it gives, it might be one of the most impactful improvements.

If it is not possible to collect more data, a maximal usage of the available data might increase the performance of the models. By augmenting the data it is possible to generate a larger data set based on the accessible images. The images can be augmented in many ways, and the most useful modifications are brightness, rotation,

distortion and blurring. If the brightness is manipulated in the image, the image will appear either darker or lighter than the original. This can be beneficial for recognizing the spectral signature in different light settings. The reflection from the material heavily depends on an external light source, and throughout the day the light from the sun changes. The area changes brightness, shadows and sun angle depending on time of the day. If an image brightness is augmented, it is possible to expose the model to artificial data that can mimic different light conditions, and the model might generalize independent of the light conditions. Another advantageous augmentation of the data is rotation. By rotating the image and setting the roofs in different angles, the model will experience varying localizations of houses and probably find all kinds of houses independent of their position in the terrain. Both distortion and blurring the image can be useful to get more variation of the elements in the image. If the image is distorted there is the same data just in another spatial form, and blurring is an unclear image. These are not desirable characteristics the model can learn but are probably harmless modifications to generate artificial data.

Some data augmentations can be especially good at extracting features for classifying the roof materials. Therefore, a feature engineering of the data to isolate the model's essential features can be a valuable step toward better results.

Semantic segmentation is a growing scientific field with many innovations. The performance of the U-net probably made it one of the most known and popular, but in the later years newer and better architectures have come along. Most of the architectures have a backbone like the ResNet, and beyond that there are many configurations. Some popular architectures that have a lot of open source solutions are: LinkNet, PSPNet and FPN. The website `www.paperswithcode.com/sota` tracks ranking over best performing architecture on known data sets. One popular data set *SkyScapes*, aerial photos of urban environments, has a net called *DeepLabV3+* and one called *SkyScapesNet-Dense* that scores the overall best [may 2022]. The DeepLabV3+ and SkyScrapeNet scores respectively 38.2 and 40.1 in the IoU, compared to a plain U-net only has a score of 14.2.

In the jungle of architecture, it might be a jungle of backbones. There can be beneficial to search for backbones that are trained on hyperspectral images, and ideally airborne hyperspectral images.

The last recommendation for future work to improve the models is to change the number of filters in the Res-U-Net. The Res-U-Net compacts the data from 399 dimensions to 64 filters in the first layers. This might lose some spectral information, and it can be beneficial to have a starting filter size that can handle and be larger than the amount of dimension.

To sum up the recommendations for further work: First, generate more quality data, augment available data, then experiment with other architectures or backbones. A good place to start is tensorflow's model garden: `https://github.com/tensorflow/models/tree/master/official`.

6 Conclusion

In this study, hyperspectral and LiDAR data are used in analysis to localize and classify roof materials. Hyperspectral and LiDAR data are obtained simultaneously over Bærum municipality right outside of Oslo, Norway. A ground truth was generated based on local field work, information from FKB and the watershed algorithm. A DSM and DTM are extracted from the LiDAR point cloud to make a nDSM that explains the heights of objects in the area. The hyperpectral data that contains VNIR and SWIR are fused and then merged with the nDSM. The data is then split into three sets: a training set, a validation set and a test set. Validation set is used to validate the training performance and the test set is used to examine the results visually. Both of these sets are used to evaluate the metric score of the model due to lack of classes in the test set.

The model in this thesis is a supervised fully connected convolutional neural network. This model has a U-net architecture with ResNet34 as the backbone. It learns to localize and classify roof materials. The metric results from Table 4.1 and 4.2 for accuracy, F1 Score weighted and Matthews Correlation Coefficient are respectively for the multiclass and binary problem. The metric scores for the multiclass problem are 0.903, 0.896 and 0.579. The scores for the binary problems are 0.948, 0.946 and 0.789.

Based on the score from the results and the visual examination of the train set I conclude that semantic segmentation is a viable technique in the answer of the two research questions. The technique can detect roof in urban environments with high precision, and it is viable on fused hyperspectral and LiDAR data to localize and classify roof materials. Compared to shallower machine learning algorithms, semantic segmentation significantly reduces the salt-pepper effect in the visual results. With quality data of all classes, this method yields desirable objectifying characteristics that consider both an object's spectral and geometrical information and its surroundings spatial and spectral information.

References

[1] C. Chisense, M. Hahn, and J. Engels, "Classification of roof materials using hyperspectral data," *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. 39, no. B7, p. 103, 2012.

[2] B. Forster, "Coefficient of variation as a measure of urban spatial attributes, using spot hrv and landsat tm data," *International Journal of Remote Sensing*, vol. 14, no. 12, pp. 2403–2409, 1993.

[3] J. C. Coiner and A. L. Levine, "Applications of remote sensing to urban problems," *Urban Systems*, vol. 4, no. 3-4, pp. 205–219, 1979.

[4] D. Shen, G. Wu, and H.-I. Suk, "Deep learning in medical image analysis," *Annual review of biomedical engineering*, vol. 19, pp. 221–248, 2017.

[5] D. Lemp and U. Weidner, "Segment-based characterization of roof surfaces using hyperspectral and laser scanning data," in *Proceedings. 2005 IEEE International Geoscience and Remote Sensing Symposium, 2005. IGARSS'05.*, vol. 7, pp. 4942–4945, IEEE, 2005.

[6] S. Van der Linden, A. Janz, B. Waske, M. Eiden, and P. Hostert, "Classifying segmented hyperspectral data from a heterogeneous urban environment using support vector machines," *Journal of Applied Remote Sensing*, vol. 1, no. 1, p. 013543, 2007.

[7] A. Rangnekar, N. Mokashi, E. J. Ientilucci, C. Kanan, and M. J. Hoffman, "Aerorit: A new scene for hyperspectral image analysis," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 58, no. 11, pp. 8116–8124, 2020.

[8] F. Trevisiol, A. Lambertini, F. Franci, and E. Mandanici, "An object-oriented approach to the classification of roofing materials using very high-resolution satellite stereo-pairs," *Remote Sensing*, vol. 14, no. 4, 2022.

[9] R. Senchuri, "Road edge detection using hyperspectral and lidar data based on machine and deep learning," Master's thesis, Norwegian University of Life Sciences, Ås, 2020.

[10] R. A. Schowengerdt, *Remote sensing: models and methods for image processing*. Elsevier, 2006.

[11] R. J. Lee and S. L. Steele, "Military use of satellite communications, remote sensing, and global positioning systems in the war on terror," *J. Air L. & Com.*, vol. 79, p. 69, 2014.

[12] R. J. Birk, T. Stanley, G. I. Snyder, T. A. Hennig, M. M. Fladeland, and F. Policelli, "Government programs for research and operational uses of commercial remote sensing data," *Remote Sensing of Environment*, vol. 88, no. 1-2, pp. 3–16, 2003.

[13] N. Kussul, M. Lavreniuk, S. Skakun, and A. Shelestov, "Deep learning classification of land cover and crop types using remote sensing data," *IEEE Geoscience and Remote Sensing Letters*, vol. 14, no. 5, pp. 778–782, 2017.

[14] J. Li, Y. Pei, S. Zhao, R. Xiao, X. Sang, and C. Zhang, "A review of remote sensing for environmental monitoring in china," *Remote Sensing*, vol. 12, no. 7, p. 1130, 2020.

[15] J. M. Amigo, *Hyperspectral imaging*. Elsevier, 2019.

[16] C. Kyrkou, S. Timotheou, P. Kolios, T. Theocharides, and C. Panayiotou, "Drones: Augmenting our quality of life," *IEEE Potentials*, vol. 38, no. 1, pp. 30–36, 2019.

[17] W. K. Panofsky and M. Phillips, *Classical electricity and magnetism*. Courier Corporation, 2005.

[18] J. Byrnes, *Unexploded ordnance detection and mitigation*. Springer Science & Business Media, 2008.

[19] R. B. Smith, "Introduction to hyperspectral imagining," *MircoImages Inc.*, pp. 2–10, 1 2012.

[20] D. G. Manolakis, "Taxonomy of detection algorithms for hyperspectral imaging applications," *Optical engineering*, vol. 44, no. 6, p. 066403, 2005.

[21] C. J. Robinove, "Computation with physical values from landsat digital data," *Photogrammetric Engineering and Remote Sensing*, vol. 48, no. 5, pp. 781–784, 1982.

[22] E. Vermote and A. Vermeulen, "Atmospheric correction algorithm: spectral reflectances (mod09)," *ATBD version*, vol. 4, pp. 1–107, 1999.

[23] J. Barrette, P. August, F. Golet, *et al.*, "Accuracy assessment of wetland boundary delineation using aerial photography and digital orthophotography," *Photogrammetric Engineering and Remote Sensing*, vol. 66, no. 4, pp. 409–416, 2000.

[24] X. A. Yao, "Georeferencing and geocoding," in *International Encyclopedia of Human Geography (Second Edition)* (A. Kobayashi, ed.), pp. 111–117, Oxford: Elsevier, second edition ed., 2020.

[25] J. M. Jurado, L. Pádua, J. Hruška, F. R. Feito, and J. J. Sousa, "An efficient method for generating uav-based hyperspectral mosaics using push-broom sensors," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 14, pp. 6515–6531, 2021.

[26] J. P. Kerekes and J. R. Schott, "Hyperspectral imaging systems," *Hyperspectral data exploitation: Theory and applications*, pp. 19–45, 2007.

[27] J. Schiewe, "Grundlagen der fernerkundung - lecture notes," *HafenCity University Hamburg*, 2006.

[28] A. Kuras, M. Brell, J. Rizzi, and I. Burud, "Hyperspectral and lidar data applied to the urban land cover machine learning and neural-network-based classification: A review," *Remote Sensing*, vol. 13, no. 17, p. 3393, 2021.

[29] W. Zhou, G. Huang, A. Troy, and M. Cadenasso, "Object-based land cover classification of shaded areas in high spatial resolution imagery of urban areas: A comparison study," *Remote Sensing of Environment*, vol. 113, no. 8, pp. 1769–1777, 2009.

[30] Kartverket, "Fkb generell del 5.0," *Kartverket*, vol. 5.0, pp. 1–10, 2022.

[31] T. Vo-Dinh, "A hyperspectral imaging system for in vivo optical diagnostics," *IEEE Engineering in Medicine and Biology Magazine*, vol. 23, no. 5, pp. 40–49, 2004.

[32] M. Govender, K. Chetty, and H. Bulcock, "A review of hyperspectral remote sensing and its application in vegetation and water resource studies," *Water Sa*, vol. 33, no. 2, pp. 145–151, 2007.

[33] E. Adam, O. Mutanga, and D. Rugege, "Multispectral and hyperspectral remote sensing for identification and mapping of wetland vegetation: a review," *Wetlands Ecology and Management*, vol. 18, no. 3, pp. 281–296, 2010.

[34] Y.-Z. Feng and D.-W. Sun, "Application of hyperspectral imaging in food safety inspection and control: a review," *Critical reviews in food science and nutrition*, vol. 52, no. 11, pp. 1039–1058, 2012.

[35] M. A. Afromowitz, J. B. Callis, D. M. Heimbach, L. A. DeSoto, and M. K. Norton, "Multispectral imaging of burn wounds: a new clinical instrument for evaluating burn depth," *IEEE Transactions on Biomedical Engineering*, vol. 35, no. 10, pp. 842–850, 1988.

[36] A. F. Goetz, "Three decades of hyperspectral remote sensing of the earth: A personal view," *Remote Sensing of Environment*, vol. 113, pp. S5–S16, 2009.

[37] F. Vasefi, N. MacKinnon, and D. Farkas, "Chapter 16 - hyperspectral and multispectral imaging in dermatology," in *Imaging in Dermatology*, pp. 187–201, Elsevier, 2016.

[38] J. M. Amigo, H. Babamoradi, and S. Elcoroaristizabal, "Hyperspectral image analysis. a tutorial," *Analytica chimica acta*, vol. 896, pp. 34–51, 2015.

[39] G. Polder and A. Gowen, "The hype in spectral imaging," *Journal of Spectral Imaging*, vol. 9, no. 1, p. a4, 2020.

[40] A. Laborde, F. Puig-Castellví, D. Jouan-Rimbaud Bouveresse, L. Eveleigh, C. Cordella, and B. Jaillais, "Detection of chocolate powder adulteration with peanut using near-infrared hyperspectral imaging and multivariate curve resolution," *Food Control*, vol. 119, pp. 1074–54, 2021.

[41] Z. Fu, R. T. Tan, and T. Caelli, "Specular free spectral imaging using orthogonal subspace projection," in *18th International Conference on Pattern Recognition (ICPR'06)*, vol. 1, pp. 812–815, IEEE, 2006.

[42] G. M. Siegmund A, "Satelliten- und luftbildeinsatz zur analyse von umweltveränderungen im geographieunterricht," *Geographie und Schule*, p. 154, 2005.

[43] R. Collis, "Lidar," *Applied optics*, vol. 9, no. 8, pp. 1782–1788, 1970.

[44] R. Harrap and M. Lato, "An overview of lidar: collection to application," *NGI publication*, vol. 2, pp. 1–9, 2010.

[45] S. Crutchley and P. Crow, *The Light Fantastic: Using airborne lidar in archaeological survey*. English Heritage Swindon, 2010.

[46] Z. Li, C. Zhu, and C. Gold, *Digital terrain modeling: principles and methodology*. CRC press, 2004.

[47] A. Khosravipour, A. K. Skidmore, and M. Isenburg, "Generating spike-free digital surface models using lidar raw point clouds: A new approach for forestry applications," *International journal of applied earth observation and geoinformation*, vol. 52, pp. 104–114, 2016.

[48] Q. Chen, D. Baldocchi, P. Gong, and M. Kelly, "Isolating individual trees in a savanna woodland using small footprint lidar data," *Photogrammetric Engineering & Remote Sensing*, vol. 72, no. 8, pp. 923–932, 2006.

[49] F. Morsdorf, E. Meier, B. Kötz, K. I. Itten, M. Dobbertin, and B. Allgöwer, "Lidar-based geometric reconstruction of boreal type forest stands at single tree level for forest and wildland fire management," *Remote sensing of environment*, vol. 92, no. 3, pp. 353–362, 2004.

[50] G. Priestnall, J. Jaafar, and A. Duncan, "Extracting urban features from lidar digital surface models," *Computers, Environment and Urban Systems*, vol. 24, no. 2, pp. 65–78, 2000.

[51] S. Raschka and V. Mirjalili, *Python machine learning: Machine learning and deep learning with Python, scikit-learn, and TensorFlow 2*. Packt Publishing Ltd, 2019.

[52] I. Androutsopoulos, G. Paliouras, V. Karkaletsis, G. Sakkis, C. D. Spyropoulos, and P. Stamatopoulos, "Learning to filter spam e-mail: A comparison of a naive bayesian and a memory-based approach," *arXiv preprint cs/0009009*, 2000.

[53] L. Deng and X. Li, "Machine learning paradigms for speech recognition: An overview," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 21, no. 5, pp. 1060–1089, 2013.

[54] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *International Conference on Medical image computing and computer-assisted intervention*, pp. 234–241, Springer, 2015.

[55] J. Janai, F. Güney, A. Behl, A. Geiger, *et al.*, "Computer vision for autonomous vehicles: Problems, datasets and state of the art," *Foundations and Trends® in Computer Graphics and Vision*, vol. 12, no. 1–3, pp. 1–308, 2020.

[56] M. Garcia, "Racist in the Machine: The Disturbing Implications of Algorithmic Bias," *World Policy Journal*, vol. 33, pp. 111–117, 12 2016.

[57] M. T. Ribeiro, S. Singh, and C. Guestrin, "" why should i trust you?" explaining the predictions of any classifier," in *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pp. 1135–1144, 2016.

[58] M. Mangel and F. J. Samaniego, "Abraham wald's work on aircraft survivability," *Journal of the American Statistical Association*, vol. 79, no. 386, pp. 259–267, 1984.

[59] T. Dietterich, "Overfitting and undercomputing in machine learning," *ACM computing surveys (CSUR)*, vol. 27, no. 3, pp. 326–327, 1995.

[60] H. L. Dreyfus, *What computers still can't do: A critique of artificial reason*. MIT press, 1992.

[61] W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *The bulletin of mathematical biophysics*, vol. 5, no. 4, pp. 115–133, 1943.

[62] B. Widrow and M. E. Hoff, "Adaptive switching circuits," tech. rep., Stanford Univ Ca Stanford Electronics Labs, 1960.

[63] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in neural information processing systems*, vol. 25, 2012.

[64] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural networks*, vol. 2, no. 5, pp. 359–366, 1989.

[65] J. Han and C. Moraga, "The influence of the sigmoid function parameters on the speed of backpropagation learning," in *International workshop on artificial neural networks*, pp. 195–201, Springer, 1995.

[66] K. Fukushima, "Visual feature extraction by a multilayered network of analog threshold elements," *IEEE Transactions on Systems Science and Cybernetics*, vol. 5, no. 4, pp. 322–333, 1969.

[67] S. Hochreiter, "The vanishing gradient problem during learning recurrent neural nets and problem solutions," *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, vol. 6, no. 02, pp. 107–116, 1998.

[68] C. M. Bishop and N. M. Nasrabadi, *Pattern recognition and machine learning*, vol. 4. Springer, 2006.

[69] Y. LeCun, B. Boser, J. Denker, D. Henderson, R. Howard, W. Hubbard, and L. Jackel, "Handwritten digit recognition with a back-propagation network," *Advances in neural information processing systems*, vol. 2, 1989.

[70] L. G. Shapiro, G. C. Stockman, *et al.*, *Computer vision*, vol. 3. Prentice hall New Jersey, 2001.

[71] Z. Pan, J. Xu, Y. Guo, Y. Hu, and G. Wang, "Deep learning segmentation and classification for urban village using a worldview satellite image based on u-net," *Remote Sensing*, vol. 12, no. 10, p. 1574, 2020.

[72] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

[73] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[74] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1–9, 2015.

[75] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pp. 249–256, JMLR Workshop and Conference Proceedings, 2010.

[76] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *International conference on machine learning*, pp. 448–456, PMLR, 2015.

[77] K. He and J. Sun, "Convolutional neural networks at constrained time cost," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 5353–5360, 2015.

[78] F. Milletari, N. Navab, and S.-A. Ahmadi, "V-net: Fully convolutional neural networks for volumetric medical image segmentation," in *2016 fourth international conference on 3D vision (3DV)*, pp. 565–571, IEEE, 2016.

[79] L. Yu, X. Yang, H. Chen, J. Qin, and P. A. Heng, "Volumetric convnets with mixed residual connections for automated prostate segmentation from 3d mr images," in *Thirty-first AAAI conference on artificial intelligence*, 2017.

[80] R. C. J. Van, *Information Retrieval*. Butterworths, 1979.

[81] B. W. Matthews, "Comparison of the predicted and observed secondary structure of t4 phage lysozyme," *Biochimica et Biophysica Acta (BBA)-Protein Structure*, vol. 405, no. 2, pp. 442–451, 1975.

[82] S. Boughorbel, F. Jarray, and M. El-Anbari, "Optimal classifier for imbalanced data using matthews correlation coefficient metric," *PloS one*, vol. 12, no. 6, p. e0177678, 2017.

[83] J. Gorodkin, "Comparing two k-category assignments by a k-category correlation coefficient," *Computational Biology and Chemistry*, vol. 28, no. 5, pp. 367–374, 2004.

[84] L. T.-Y., G. P., G. R., H. K., and D. P., "Focal loss for dense object detection," vol. 2017-October, p. 2999 – 3007, 2017. Cited by: 5859; All Open Access, Green Open Access.

[85] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 580–587, 2014.

[86] P. Jaccard, "The distribution of the flora in the alpine zone. 1," *New phytologist*, vol. 11, no. 2, pp. 37–50, 1912.

[87] T. Eelbode, J. Bertels, M. Berman, D. Vandermeulen, F. Maes, R. Bisschops, and M. B. Blaschko, "Optimization for medical image segmentation: Theory and practice when evaluating with dice score or jaccard index," *IEEE Transactions on Medical Imaging*, vol. 39, no. 11, pp. 3679–3690, 2020.

[88] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[89] S. Dasiopoulou, V. Mezaris, I. Kompatsiaris, V.-K. Papastathis, and M. G. Strintzis, "Knowledge-assisted semantic video object detection," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 15, no. 10, pp. 1210–1224, 2005.

[90] S. Beucher, "Use of watersheds in contour detection," in *Proceedings of the International Workshop on Image Processing*, CCETT, 1979.

[91] V. J. w/ Terratec AS, "Rapport for datainnsamling og prosessering av hyperspektrale data, rff fou maskinlæring for automatisk kartlegging fra laser og hyperspektrale data," *10990*, 2019.

[92] C. A. Laben and B. V. Brower, "Process for enhancing the spatial resolution of multispectral imagery using pan-sharpening," Jan. 4 2000. US Patent 6,011,875.

[93] P. Yakubovskiy, "Segmentation models." `https://github.com/qubvel/segmentation_models`, 2019.

[94] M. Bassier, B. Van Genechten, and M. Vergauwen, "Classification of sensor independent point cloud data of building objects using random forests," *Journal of Building Engineering*, vol. 21, pp. 468–477, 2019.

[95] M. Guo, H. Liu, Y. Xu, and Y. Huang, "Building extraction based on u-net with an attention block and multiple losses," *Remote Sensing*, vol. 12, no. 9, p. 1400, 2020.

[96] P. Ghamisi, B. Höfle, and X. X. Zhu, "Hyperspectral and lidar data fusion using extinction profiles and deep convolutional neural network," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 10, no. 6, pp. 3011–3024, 2016.

[97] R. Hang, Z. Li, P. Ghamisi, D. Hong, G. Xia, and Q. Liu, "Classification of hyperspectral and lidar data using coupled cnns," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 58, no. 7, pp. 4939–4950, 2020.

[98] Q. Feng, D. Zhu, J. Yang, and B. Li, "Multisource hyperspectral and lidar data fusion for urban land-use mapping based on a modified two-branch convolutional neural network," *ISPRS International Journal of Geo-Information*, vol. 8, no. 1, p. 28, 2019.

[99] S. Morchhale, V. P. Pauca, R. J. Plemmons, and T. C. Torgersen, "Classification of pixel-level fused hyperspectral and lidar data using deep convolutional neural networks," in *2016 8th workshop on hyperspectral image and signal processing: evolution in remote sensing (WHISPERS)*, pp. 1–5, IEEE, 2016.

[100] A. Primstad and Å. Stemme, "Pikselbasert arealklassifisering av urbane omrader med hyperspektrale flybilder og maskinlæring," Master's thesis, Norwegian University of Life Sciences, Ås, 2019.

A Appendix I

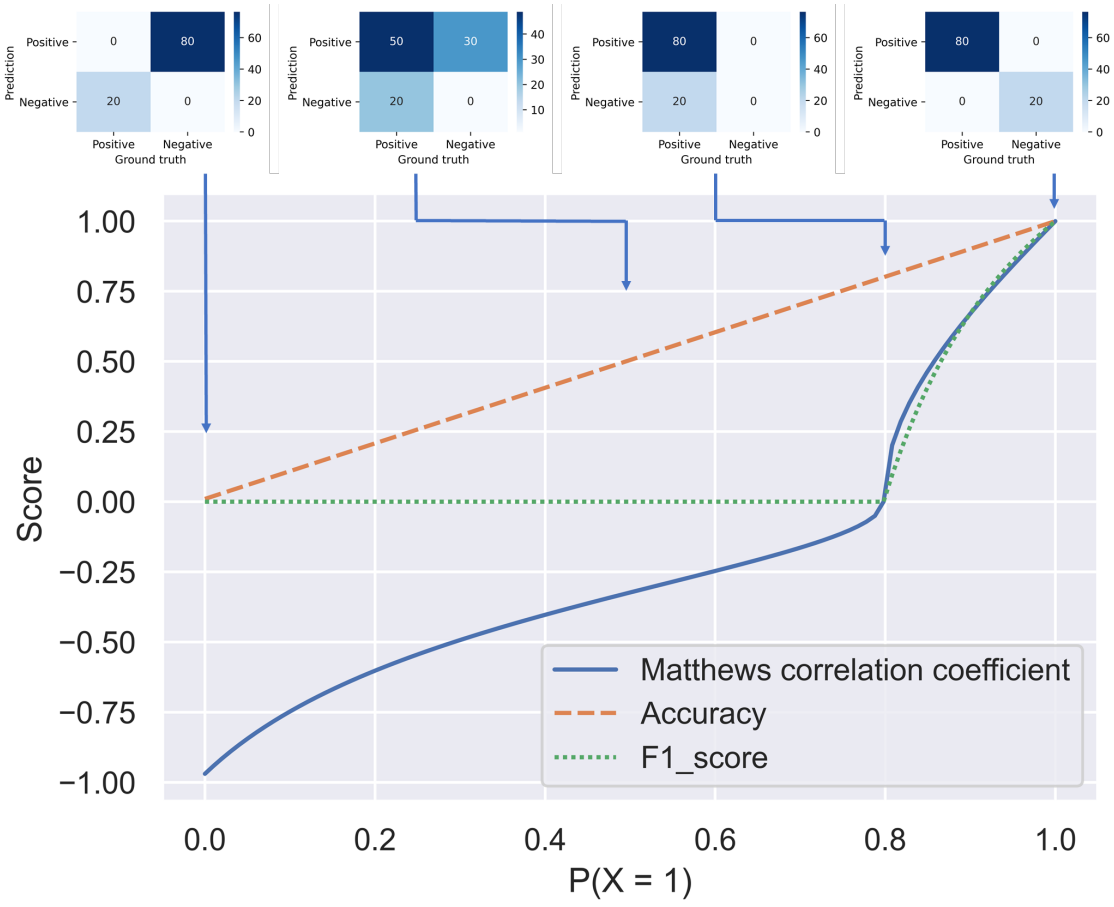


Figure A.1: Metric scores compared on an imbalanced data set. Confusion matrix explains the distribution between true or false positives and negatives

B Appendix II

Table B.1: Program and Python modules used in this study.

Module/Program name	**Appliance**	**Version**
Python	High level programming language	3.7.7
QGIS	GIS program	3.20.3
QTM	Point cloud tool	830_IX
Tensorflow	Deep learning framework	2.8.0 & 2.1.0
Keras	Deep learning API	2.8.0
Segmentation models	Image segmentation API	1.0.1
Spectral	Hyperspectral image	0.22.2
GDAL	Georeference images	3.0.2
PIL/Pillow	Georeference metadata	8.2.0
Matplotlib	Plotting graphic	3.5.1
Seaborn	Plotting graphic	0.11.2
Numpy	Mathematical functions	1.21.5
Pandas	Data analysis	1.3.5
OpenCV	Image processing	4.4.0
visualkeras	Visualise NN models	0.0.2

C Appendix III

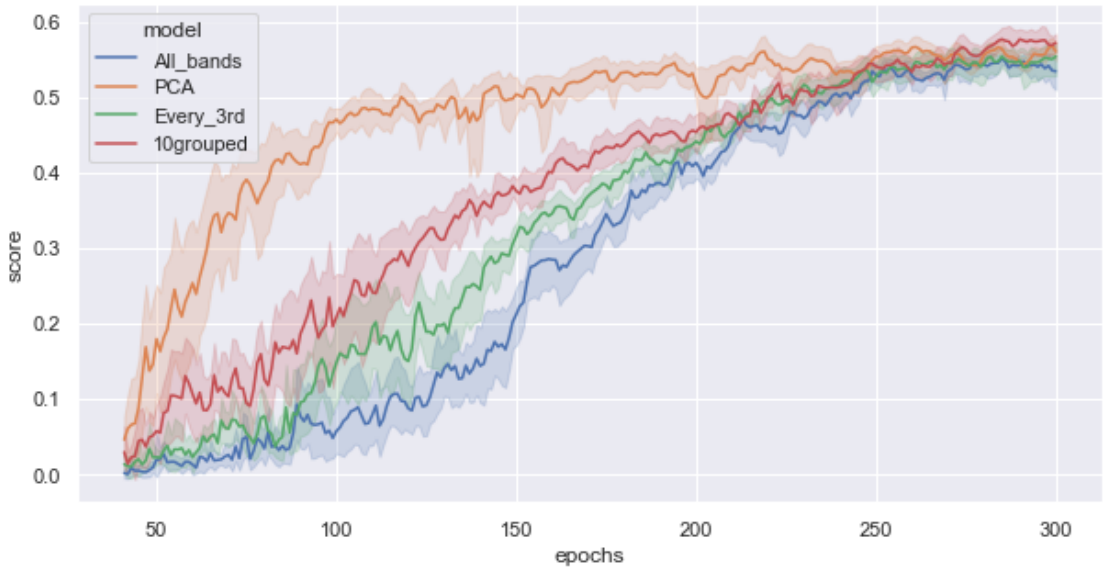


Figure C.1: Model optimizing on dimension reduction for input data. PCA is with 10 components, every 3rd is that every third wavelength is in use, and 10 grouped is batches of 10 wavelengths are extracted and the mean of these are the input data.

D Appendix IV

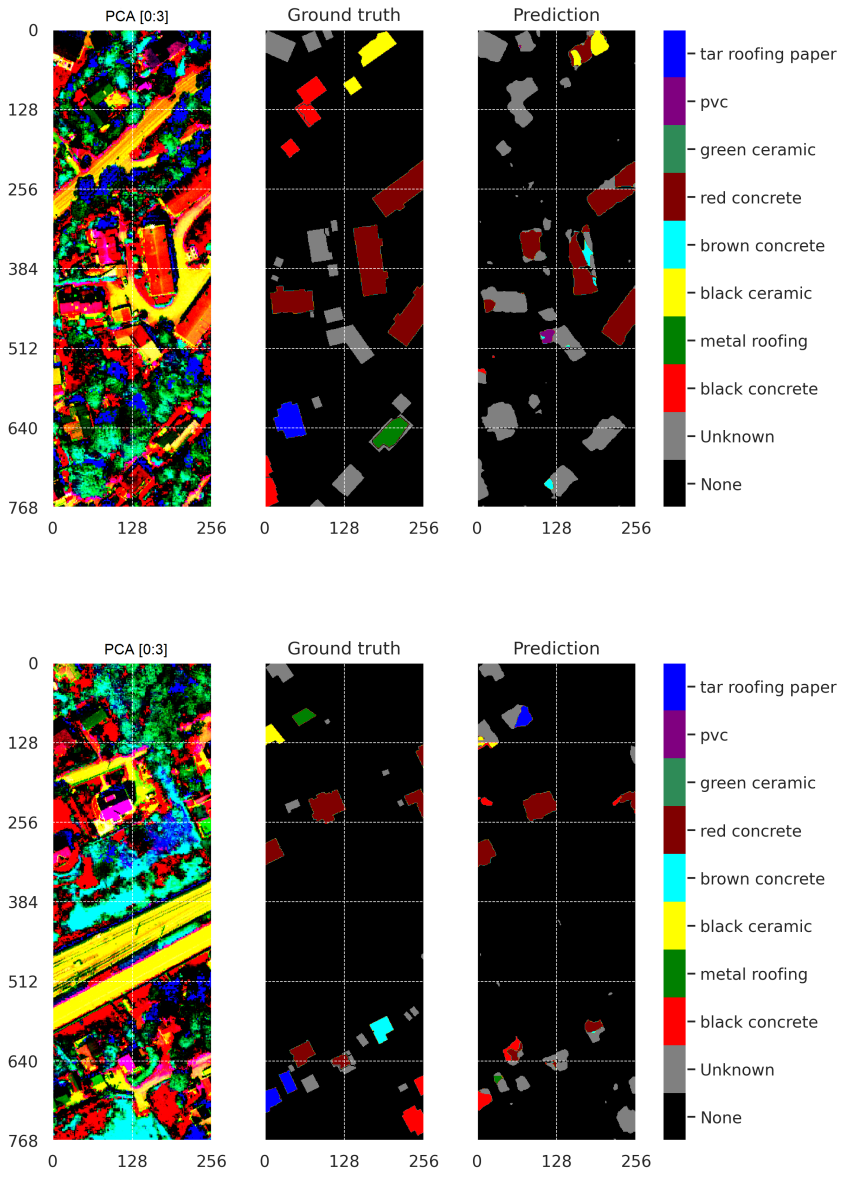


Figure D.1: Visual results for model with dimension reduction PCA with 10 components.

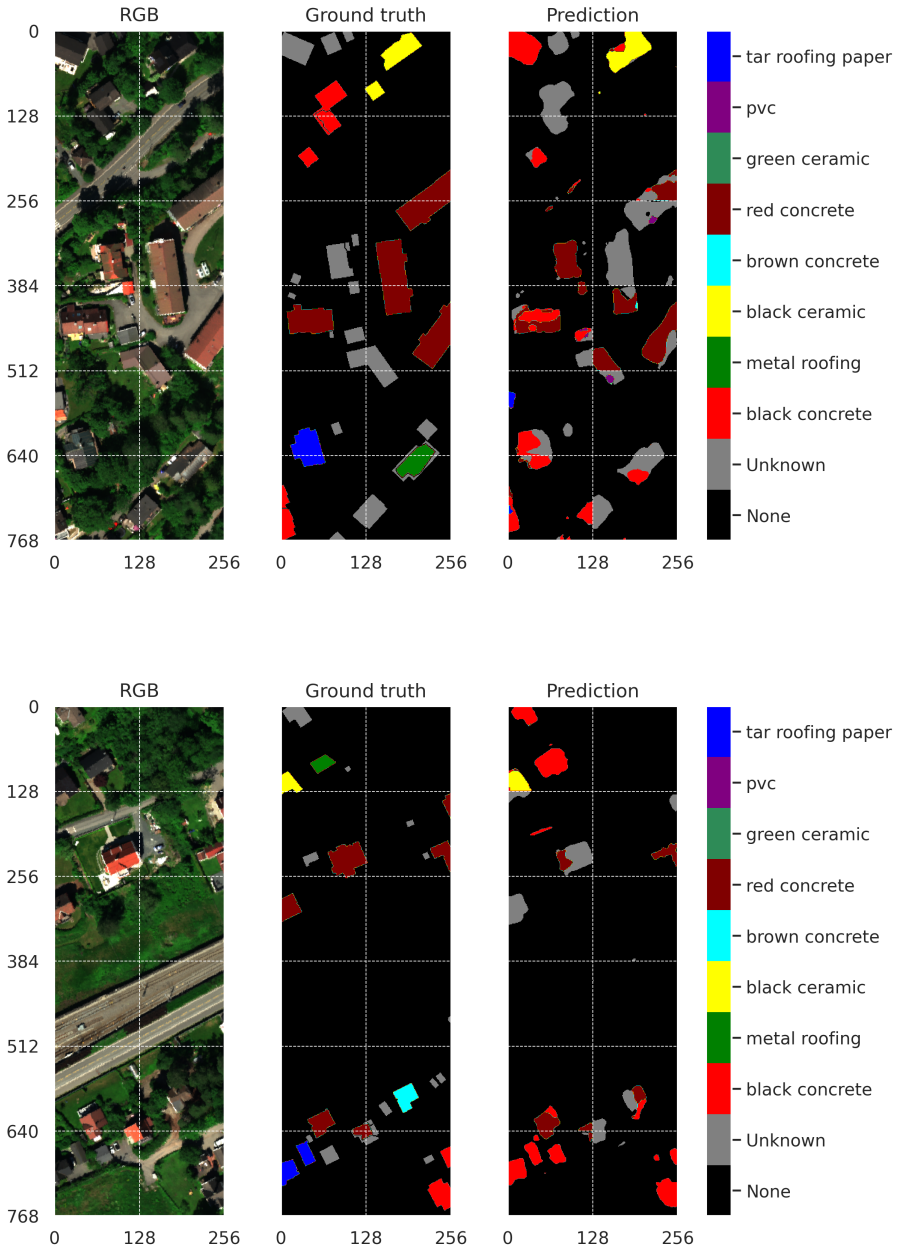


Figure D.2: Visual results for model with dimension reduction where every 3rd wavelength is extracted.

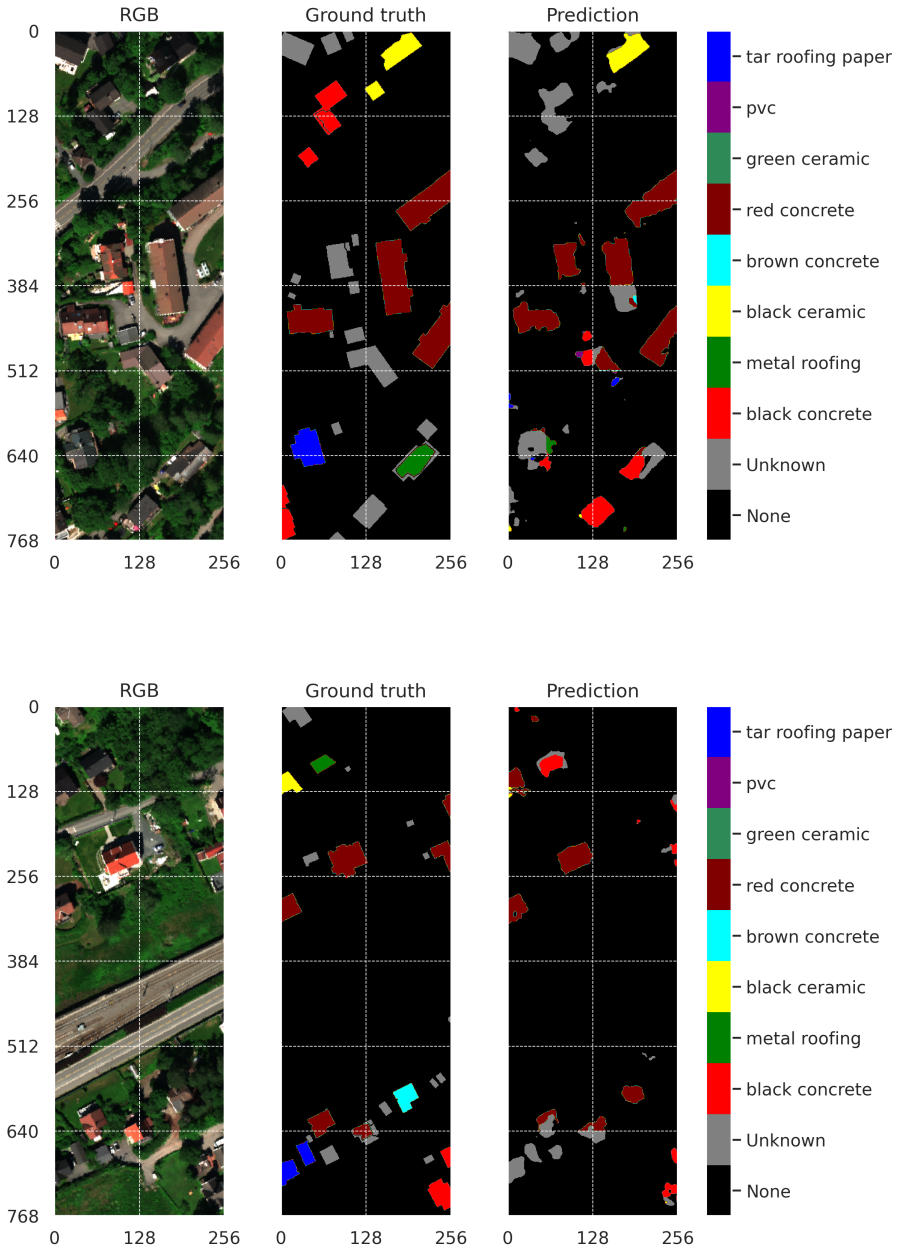


Figure D.3: Visual results for model with dimension reduction of groups of 10 and the mean of them.

E Appendix V

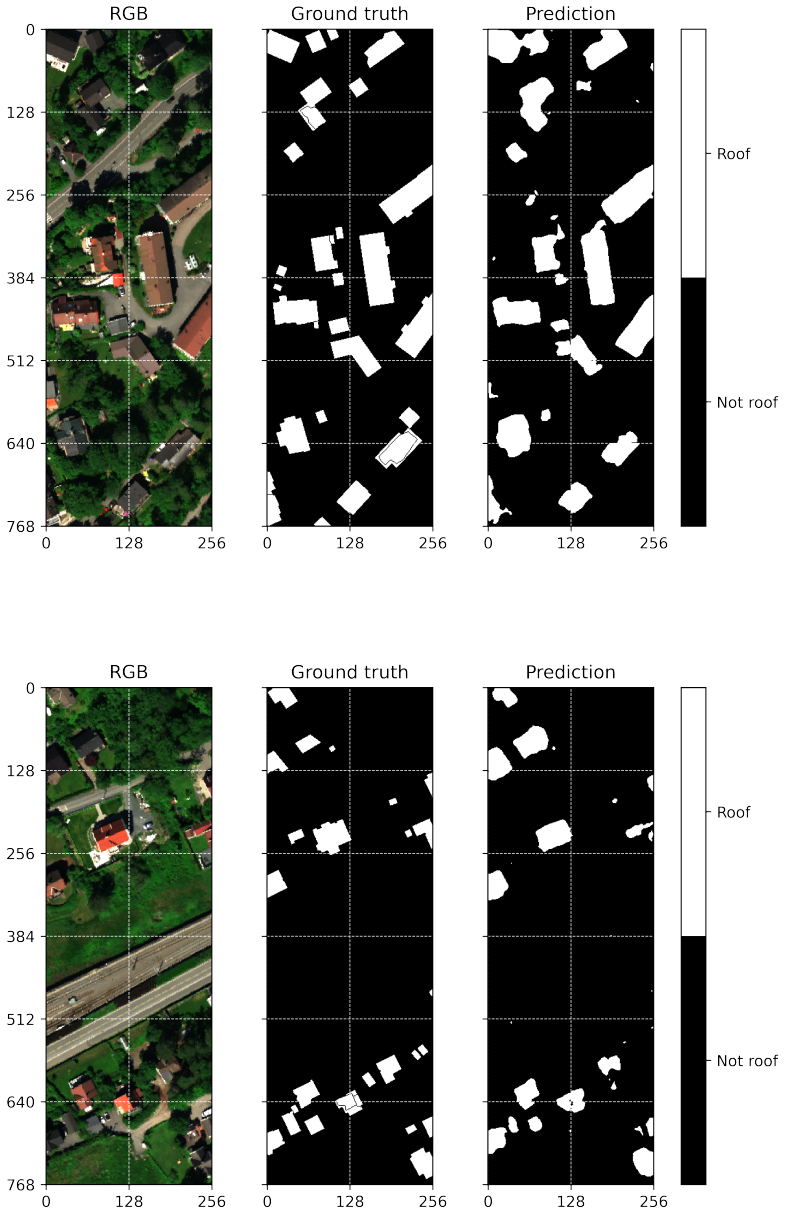


Figure E.1: Binary prediction for the best model.

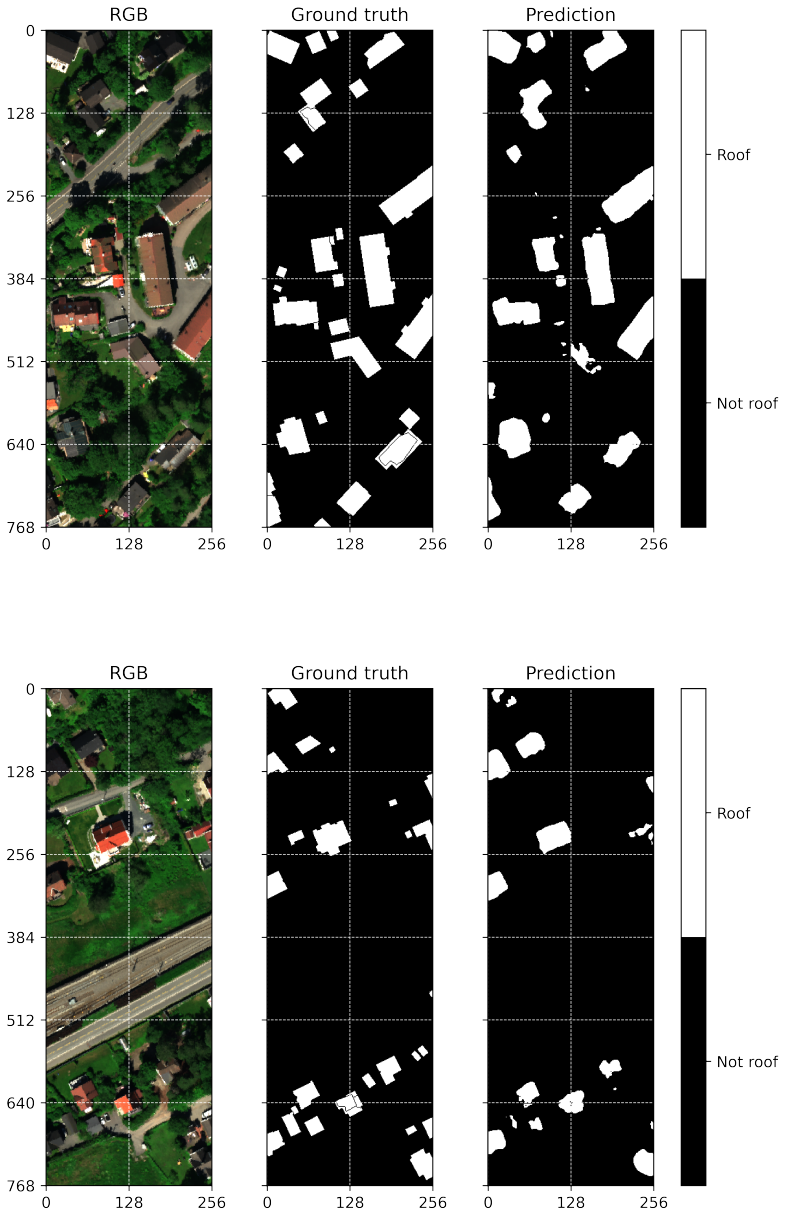


Figure E.2: Binary prediction for all models.

F Appendix VI

Comparison between Res-U-Net with ResNet34 as backbone and Random Forest. Random forest is often the best performing shallow machine learning. This algorithm requires minor computation capability compared to the Res-U-Net, and in some cases with small and medium data set Random forest performs better than a more advanced CNN. Therefore, it is curious to see if a random forest achieves just as good score and if it is worth using a complex and harder to explain algorithm. Table F.1 shows the metrics that looks very good for Random forest, but by looking at the visual results in FigureF.1 Random forest has a lot of unwanted characteristic.

Table F.1: ResNet vs Random forest metrics.

Metric name	**Random Forest**	**Res-U-Net**
Accuracy	0.898	0.903 ± 0.006
F1 Score weighted	0.874	0.896 ± 0.008
Matthews Correlation coefficient	0.618	0.579 ± 0.034

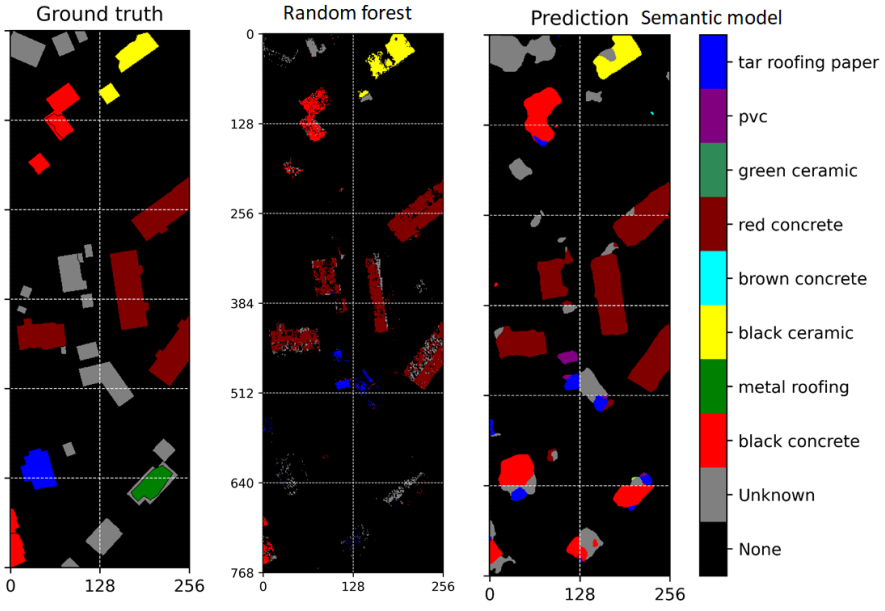


Figure F.1: Visual results from Random Forest and Res-U-Net.

G Appendix VII

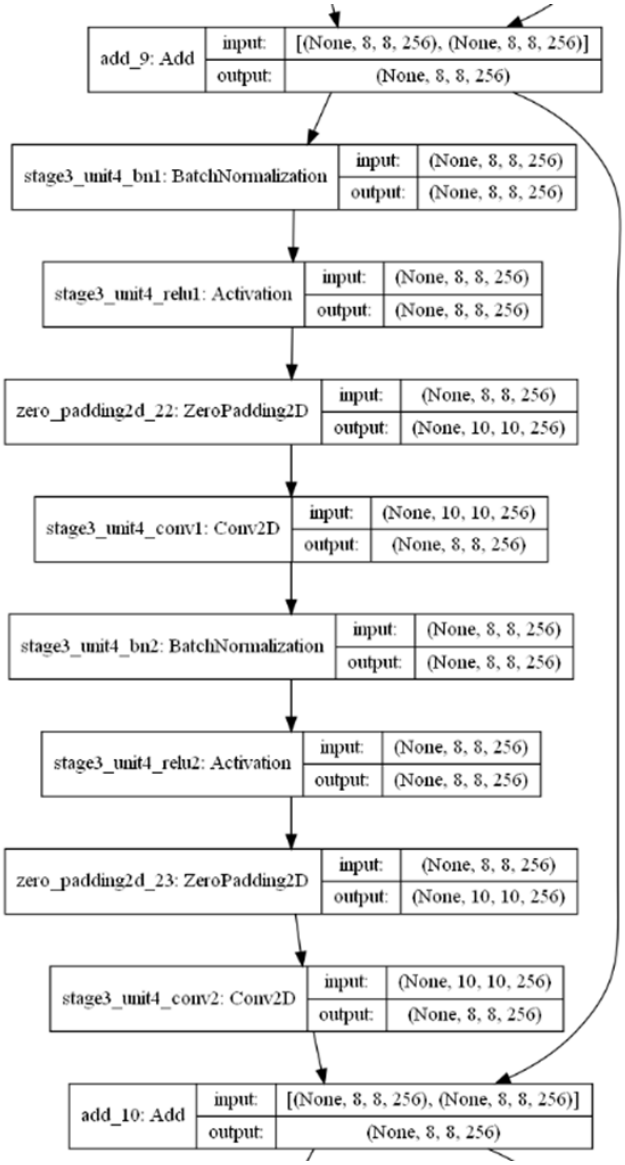


Figure G.1: One residual block for the model. Taken from the second lowest depth for the model.

H Appendix VIII

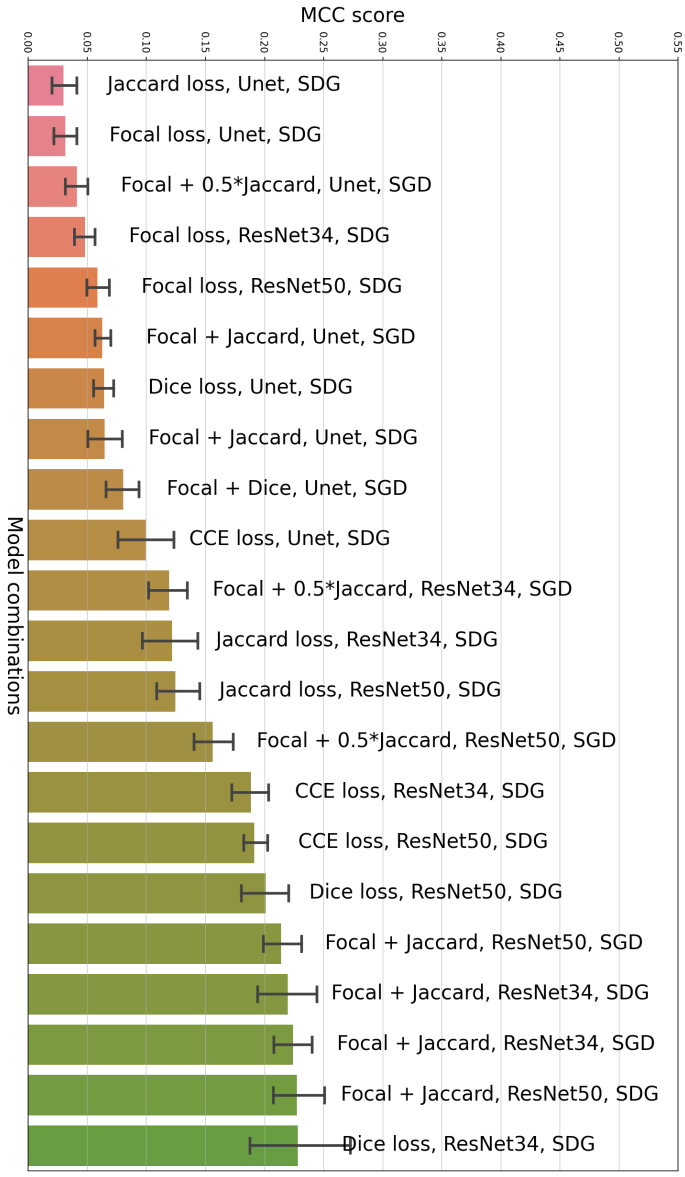


Figure H.1: Independent test for many configurations 1/3.

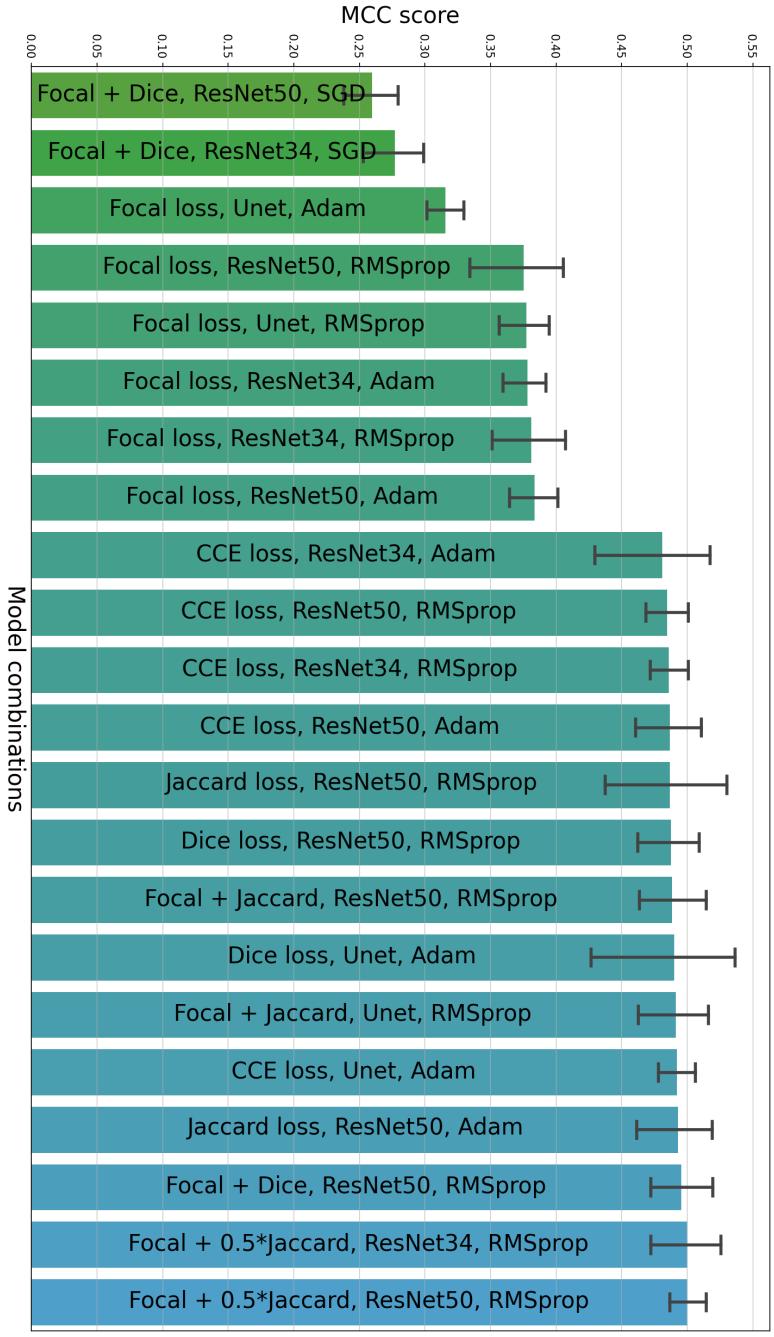


Figure H.2: Independent test for many configurations 2/3.

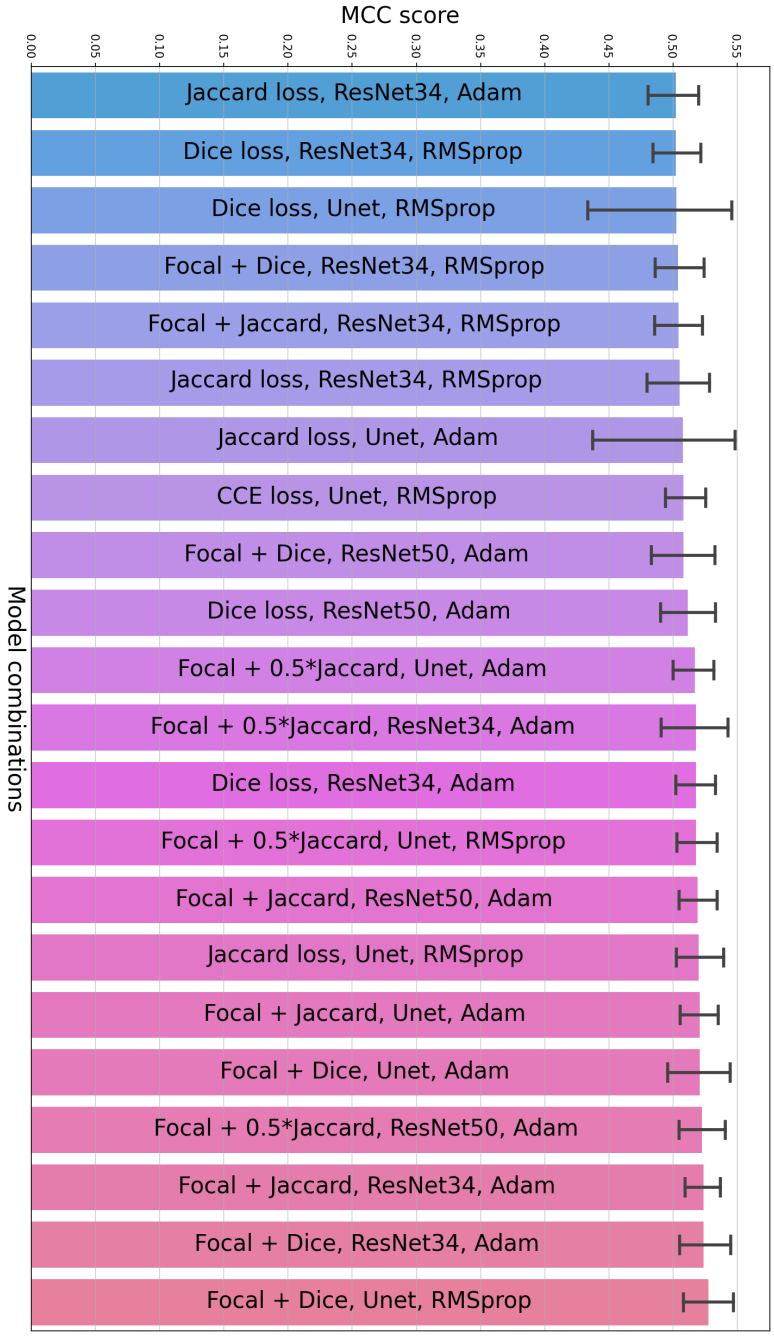


Figure H.3: Independent test for many configurations 3/3.

I Appendix IX

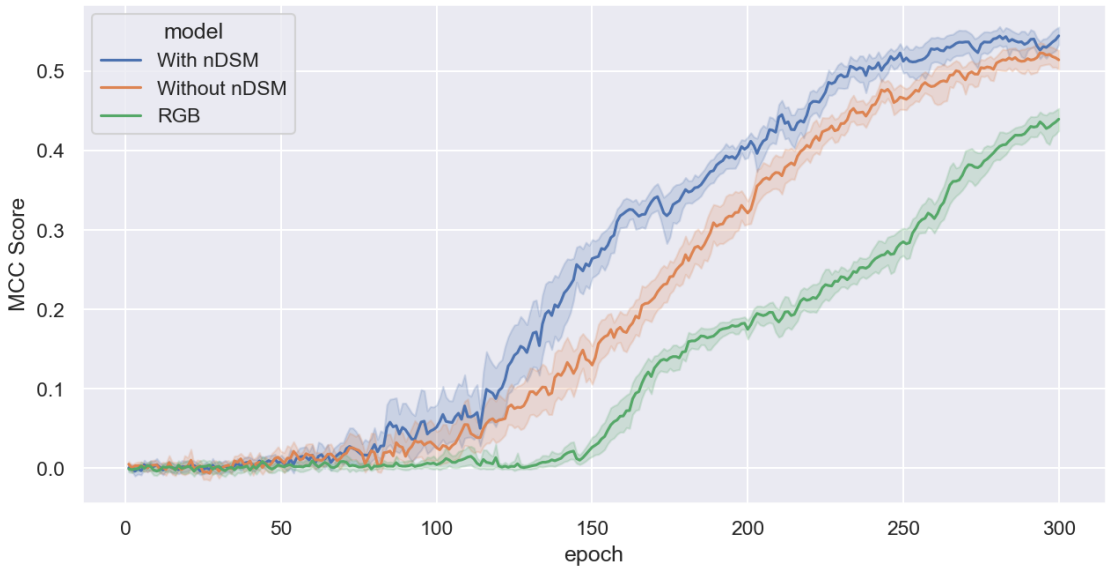
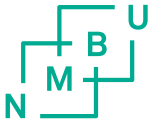


Figure I.1: Comparison on full data set, data set without nDSM and RGB channels.



Norges miljø- og biovitenskapelige universitet
Noregs miljø- og biovitskapelege universitet
Norwegian University of Life Sciences

Postboks 5003
NO-1432 Ås
Norway