An AI Approach for Detecting Soil Sealing Areas with VHR Image Data

Max Beckmann*

*Department of Geoinformatics, University of Salzburg. Email: max.beckmann@stud.sbg.ac.at

Contents

Abstract

The mapping of impervious surfaces is crucial to monitor the impact they have on environmental variables such as water quality, water run off capability, soil fertility, biodiversity or the urban heat island effect. Advances in remote sensing make very high resolution images available. The Pléiades satellites provide optical imagery with red, green, blue and near infrared bands at a very high resolution of 50 cm^2 per pixel. These channels were combined with a normalized digital surface model. Recent developments of deep learning methods have proven that deep neural networks are well suited for semantic image segmentation. In this work, we use the enhanced Pléiades imagery to train two convolutional neural network models for binary classification. One is trained using automated label data generated from open government and open-source vector data, the other was enhanced by manual digitization. The results show that the later label data helps the model learn differences between impervious and pervious surfaces better. The confidence matrix, on the test dataset, shows an accuracy of *70.2%* for the automated and *91.6%* for the manually enhanced model. A visual inspection of the predicted surfaces confirms the assessment. Keywords: Convolutional Neural Networks, Soil Sealing, Very High Resolution Imagery, UNet, Pléiades

1 Introduction

Sealed soil or impervious surfaces can be defined as areas which have been made impermeable through man made build up structures like roads, parking lots, buildings or walkways [Burghardt, 2006a]. In the field of Hydrogeology, many different delineations of the permeability of soil are made. Some surfaces can be semi-sealed, which means that the infiltration capacity of the soil is better than with fully sealed surfaces [Cheremisinoff, 1997]. In this thesis, sealed or impervious surfaces are seen as fully sealed while unsealed surfaces are permeable. With no available data regarding how large the infiltration capacity of each surface is and the nature of the approach of trying to detect the surfaces from satellite images, no further delineation between fully sealed and open surfaces will be made. While build up structures like roads and houses serve necessary functions in society, the right balance between sealed and unsealed surfaces has to be kept to prevent adverse effects of soil sealing. Impervious surfaces have many negative effects on the health of the soil underneath such as destroying the biodiversity, reduction of water and dust absorption as well as an increased risk of floods [Enzinger, 2021]. The hydrological impacts are not limited to

flooding, but also affect the water quality itself through the decrease of possible run-off and therefore filtering of water [Hurd & Civco, 2004]. Soil sealing is also a major contributor to the urban heat island effect [Weng et al., 2004] [Guo et al., 2015]. Some direct impacts mentioned above can be seen in figure 1.

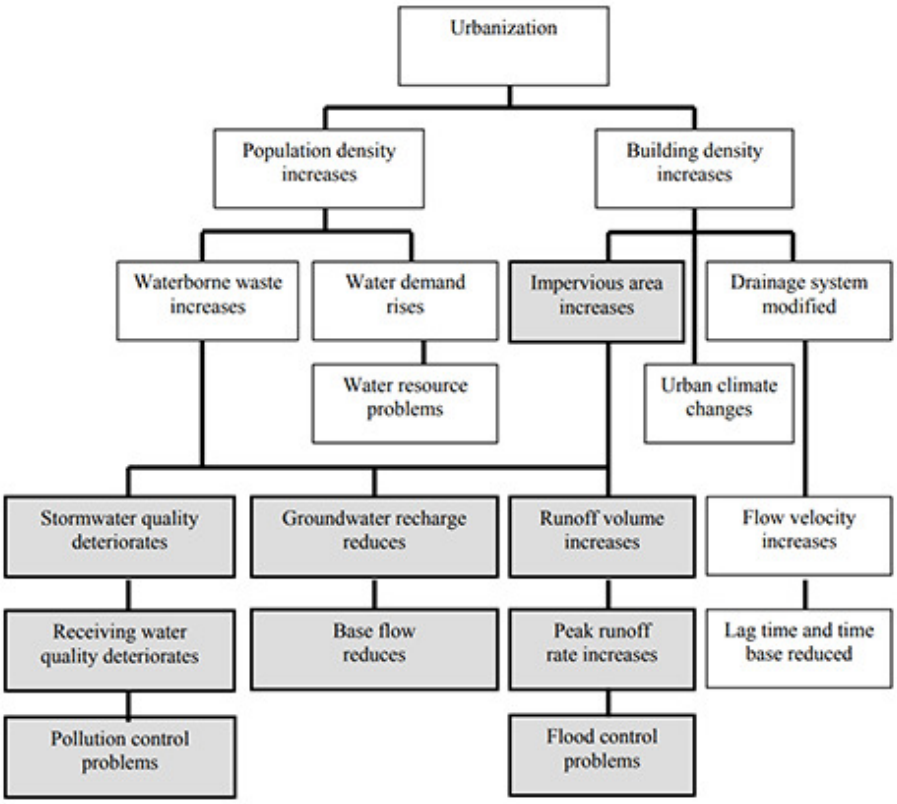

FIGURE 1: Graphic showing the effects of urbanization. In gray, direct effects of growing sealed areas are shown. Graphic from: [Hurd et al., 2004]

Between 2018 and 2021, an average of 15 to 20 km^2 of soil were sealed each year in Austria. The main driver of this development is the building and construction industry [Enzinger, 2021].

To better communicate the importance of natural unsealed soil and simulate the effects sealed areas have on heavy rain or heat periods, it is of high interest to be able to assess the percentage of sealed versus unsealed surfaces, and where those surfaces are located. Being well-informed about the soil status will help to mitigate possible natural disasters.

The aim of this work is to answer the following research questions:

Is the data availability of open source data regarding sealed surfaces good enough to train a convolutional neural network to detect sealed surfaces? Or is it necessary to manually digitize sealed surfaces in order to achieve a sufficient classification accuracy.

The case study area for this project is the city of Salzburg and the city of Hallein. Salzburg is densely populated with many build up areas. A river and few green spaces cross the living spaces. Hallein features large forests and agricultural areas, as well as a less densely populated city. The reason to choose this area was due to its representativeness of Austrian landscapes, as well as the availability of high resolution imagery data.

In order to answer the research questions, the following tasks will be fulfilled.

First, a suitable model architecture has to be found. Different model sizes will be tested to see which is the best fitting for the following experiments. This model will then be the baseline for all following training runs.

For the label data, two approaches will be compared. First, an automatic label generation approach is developed to generate labels from open source data. Secondly, the automatic labels will be manually enhanced to fit more accurately and compared with the first approach. This should determine whether the less resource intensive automatic approach is sufficient to detect impervious surfaces.

The convolutional neural networks will be using very high resolution imagery (50 cm^2 pixel size), provided by the University of Salzburg, to detect sealed areas on previously unseen image data in Salzburg state. The goal is to create a robust prediction model which is able to detect sealed areas on images from different timeframes and diverse landscapes.

The resulting convolutional neural network models will be tested on 500 random stratified sample points. The 500 points are divided in 10 subgroups with 50 points each. Each group represents distinct areas in the real world and should have homogeneous predictions. The first 6 groups contain points on sealed areas like streets, buildings, railways, sports facilities, driveways and mineral/dumping sites. The last four groups contain points on

unsealed areas like forests, pastures, arable land and water.

Comparing the hitrate of the models on the generated sample points should show which model can classify impervious surfaces more accurately. Furthermore, a look at the different classes should show where strength and deficiencies of each model lie.

In order to analyze the model further, a confindence map of the predicitons on the test set will be generated and discussed. This should produce insights on how the model could be improved, through a visualization of its struggles.

Finally, a larger image, roughly 1 km^2 or 2000 px^2, in the south of Hallein is used to test the model's practical viability. The predicted layer will be vectorized in ArcGis Pro and visually interpreted.

This work is structured as follows. In Section 2 related research is briefly mentioned, and the work is set into context. Section 3 encompasses the main work of this project and is split into three subchapters. 3.1 discusses the different data sources. How the data is preprocessed to be read by the CNN is the content of chapter 3.2. 3.3 describes the iterative process of training the network. How the testing of the model works is mentioned in 3.4. The merging of predicted tiles and georeferencing is explained in 3.5. The results are shown in 4, before the paper concludes in a discussion part with chapter 5.

2 Related Work

The importance of unsealed soil and the effects that soil sealing has on the environment have been widely discussed in literature [O'Riordan et al., 2021] [Weng et al., 2004] [Hurd & Civco, 2004] [Jacobson, 2011] [Burghardt, 2006b]. The detection of impervious surfaces from remote sensing images is crucial in order to compute maps of soil sealing at large scale and relative low resources.

In the following subchapter 2.1, different non-deep learning methodologies for impervious surfaces detection are highlighted. The subchapter 2.2 briefly explains how deep

learning with convolutional neural networks works and what components a convolutional neural network typically consists of. In 2.3 an introduction into current research endeavours using deep learning in the field of remote sensing is given. Finally, 2.4 concludes the chapter with deep learning approaches in impervious surface detection, and the differences to the approach in this paper are highlighted.

2.1 Non-Deep Learning Methods of Impervious Surface Detection

There are many different methodological approaches to detect impervious surfaces from image data. Historically, statistical indices have been used to classify sealed surfaces. Indices such as the normalized difference vegetation index [Pettorelli, 2013], bio physical composition index [Deng & Wu, 2012], normalized difference build-up index [Zha et al., 2003], or perpendicular impervious surface index [Tian et al., 2018]. In order to compute impervious surfaces through these indices, expert understanding and vast samples are needed to calculate correct thresholds.

Another approach would be to use regression and classification algorithms such as the random forest or support vector machine. A multi feature classification, using the random forest classifier, was chosen by [Zhang & Huang, 2018] to include not only spectral features but also textural, shape and class-related features. The authors classified on high resolution images (up to 1.2 m^2 per pixel) using time series data, to later show the change in sealed surfaces.

[Shrestha et al., 2021] calculated the impervious surfaces of pakistani cities utilizing the random forrest alortithm. Their image data was a fusion of Sentinal 1 and 2 images sampled to a 10m resolution, with a combination of optical and radar channels.

[Guo et al., 2020] mapped impervious surfaces, combining optical and synthetic aperture radar images with the help of the random forest classifier. The support vector machine was used to calculate an uncertainty classification of impervious surfaces using Landsat 8 data by [Shi et al., 2017].

[Shao & Liu, 2014] combined MODIS data with night time light satellite data to map impervious surfaces at a large scale.

2.2 Deep Learning with Convolutional Neural Networks

Convolutional neural networks are the most used form of Artificial Neural networks for spatial pattern analysis [Kattenborn et al., 2021]. While there are many Artificial Neural Networks including but not limited to Recurrent Neural Networks, Autoencoders, Deep Boltzmann Machines, Long Short Term Memory Networks or Modular Neural Networks, Convolutional Neural Networks are the most prevalent in remote sensing [Kattenborn et al., 2021].

The distinction between a neural network or shallow neural network and a deep neural network is the number of connected layers. A network is usually considered deep when there are more than two connected layers.

Like most neural network models, Convolutional Neural Networks are composed of neurons arranged in layers. Each neuron in a neural network calculates a weighted sum of data inputs from neurons in the previous layer, which is subsequently passed through a nonlinear activation function. This process enables the network to learn complex patterns and relationships within the data [LeCun et al., 2015].

A distinctive component of the convolutional neural network is the convolutional layer. A convolution kernel, whose size is determined by the architecture of the model, passes over each input value and aggregates all values within its kernel size. This process takes into account the spatial relationship of image pixels, allowing the convolutional neural network to learn their spatial properties. Earlier layers learn simpler features such as edges, while later layers look at more complex relationships [Brodrick et al., 2019].

The pooling layer is another important part of a convolutional neural network.Through pooling, the output of the previous convolution layers is aggregated based on the pooling kernel's size. While there are many pooling operations like, mixed pooling, stochastic pooling, spatial pyramid pooling or spectral pooling, the max pooling operation remains

the most common [Gu et al., 2018].

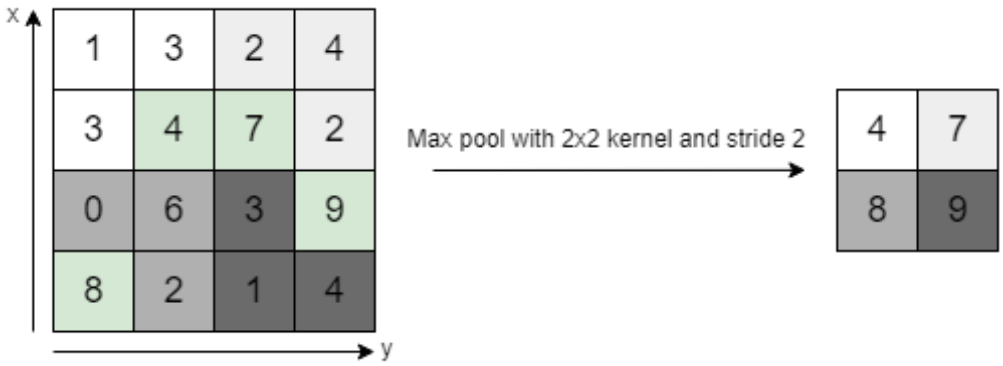

FIGURE 2: A visualization of a max pooling operation on a 4x4 matrix with a pooling kernel of the size 2 and a stride of 2.

Max pooling passes the maximum value within the kernel window on, while ignoring the others. This operation downsamples the spatial resolution of the previous layer, which leads to less data for processing of the subsequent layers. It therefore reduces the runtime of the overall network, while keeping the most relevant features for the given prediction task (see Figure 2).

The activation function is usually applied after the pooling operation. In a multi layerd CNN, the activation function will determine which neuron is activated, and their value passed on to the next layer. The activation function introduces non-linearity into the algorithm. Through this mix of linear and non-linear operations, it is possible to build a network which can represent many problem sets [Li et al., 2022]. The network could only predict linear algorithms if a linear activation function or no activation function at all would be used [He et al., 2015]. Different common activation functions used in convolutional neural networks are the sigmoid, tanh, Leaky Rectified Linear Unit or the Rectified Linear Unit [Agarap, 2018].

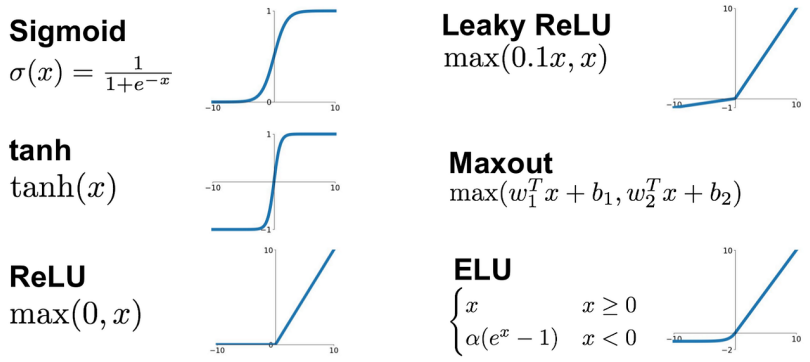

FIGURE 3: Different non-linear activation functions commonly used in and convolutional neural network. Author: [Udofia, 2019]

All the previous mentioned functions have in common that they are non-linear, which is essential for their role in the convolutional neural network.

Another part of a convolutional neural network is how the loss of the network is calculated. A loss function will calculate the difference between the predicted value and target value. Its role in the neural network is usually the measure of how much the model has learned. A small loss means that the model predicts similar to the label data, while a high loss means the opposite. A few popular loss functions encompass the cross entropy, binary focal loss, mean squared error and mean absolute error among others.

In order to minimize the loss, an optimizer is used. The optimizer minimizes the loss by backpropagation. Backpropagation in a neural network is the step of updating previous weights. If the weights used by the network produced a high loss score, the optimizer will change the weights in order to lower the loss score. Common optimizers are the Root Mean Square Prop, and adaptive moment estimation, among others.

2.3 Deep Learning in Remote Sensing

In recent years, deep learning algorithms have been on the rise in all areas of remote sensing.

Deep learning is used in image preprocessing tasks like image fusion. [Shao & Cai, 2018] propose an image fusion method using a deep convolutional neural network to adequately extract spectral and spatial features from multispectral and panchromatic images.

[Xing et al., 2018] developed a deep learning improved pan sharpening method using geometric manifold-based embeddings to estimate final high resolution and multi spectral image patches. A two branched CNN is used by [Yang et al., 2018] in order to fuse hyperspectral and multispectral images. One branch will extract features from pixels in the low resolution hyperspectral image, while the other does the same in the corresponding spatial neighbourhood of the multispectral image. Finally, the branches are concatenated to produce a fused output image.

Other applications in the area of image preprocessing can be found in image registration tasks. [Merkle et al., 2018] use a deep learning approach to match optical and synthetic aperture radar images in order to produce precise and accurate ground control points for improving geolocation accuracies. [Li et al., 2021] make use of a deep convolutional neural network to map the distribution probability of the template image pixel, resulting in higher image registration accuracy.

A large sector in remote sensing is the detection of scenes or the detection of objects in scenes. Scene detection is the identification of an image's overall scene like forest, industrial area, pasture or parking lot. Object detection tries to detect objects inside scenes. These objects could anything from cars, airplanes, oil tanks to playgrounds.

Different uses of deep learning in the field of scene classification are briefly mentioned below.

[Xiaowei Xu & Manickam, 2021] propose a novel approach to improve the scene classification accuracy through a two-way approach. One side gives a preliminary classification of the scene using a convolutional neural network. The other side learns the spatial representation of objects inside the scene through a graph neural network, in order to make a more accurate final predictions.

Another two-stream architecture is developed by [Wang et al., 2022]. Their global-local two-stream architecture aims at joining local and global feature representations, in order to increase the discrimination of images.

Object classification tasks have also benefited from advances in deep neural networks.

[Lang et al., 2021] developed a lightweight object detection framework based on a deep convolutional neural network similar to the YOLO-v4 [Bochkovskiy et al., 2020] one-stage object detection model. Their approach balances algorithm complexity and accuracy to achieve real world applicability, while still beating state-of-the-art lightweigth-models on the RSOD [Xiao, 2017] and DIOR [Li et al., 2020] dataset.

[Kellenberger et al., 2018] used the extremely high pixel resolutions of UAV images in order to detect mammals in the Namibian Kalahari Desert with a convolutional neural network.

Object based image analysis is a field where deep learning algorithms recently help to improve classification accuracies. In OBIA methodologies change since the spatial unit expands from pixels to polygons. These polygons should represent meaningful image objects [Blaschke et al., 2014].

[Ghorbanzadeh et al., 2022] compared a pixel based deep learning classification using a ResU-Net model, a rule based OBIA classification and a combined approach with each other. The combination of a heatmap generated by the ResU-Net model with the rule based OBIA provided a better classification accuracy than the others.

[Detka et al., 2023] used a combination of a convolutional neural network and multi resolution segmentation in order to classify different tree scrubs from very high resolution UAV data. Their model outperformed more traditional classifiers such as the random forest and support vector machine.

Semantic segmentation is a traditional computer vision task where every pixel is assigned a label. Similar to this is the instance segmentation, where also every pixel is assign a label, but additionally the boundaries between objects are detected, grouping pixels in unique instances.

This work chooses a semantic segmentation approach in order to classify sealed areas. Common use cases for semantic segmentation in remote sensing range from land use and

land cover classifications [Onim et al., 2020] [Tzepkenlis et al., 2023], crop cover and type analysis [Mortensen et al., 2016] [Radhika Kamath & Maheshwari, 2022] [Li et al., 2023], tree identification [Lobo Torres et al., 2020] [Wang et al., 2021] [Ulku et al., 2022] or environmental monitoring [Sulla-Menashe et al., 2019] [Alzu'bi & Alsmadi, 2022].

This overview shows how deep learning is at the forefront of research in all areas of remote sensing. Which also led to the decision to use a deep learning based classification in this master thesis.

Other research papers utilizing deep learning in order to classify impervious and pervious areas on image data and how the approach in this thesis differs will be discussed in the chapter 2.4.

2.4 Deep Learning in Impervious Surface Detection

In recent years the use of deep learning algorithms, to solve impervious surface detection tasks, have become more popular since their success in semantic image segmentation [Guo et al., 2018] [Kemker et al., 2018].

[Núñez, 2015] proposed an approach to use convolutional neural network segmentation with the biophysical composition index to better assess impervious surfaces than with traditional single threshold-based segmentation.

[Yin et al., 2022] computed the urban growth of a city through its rise of impervious surfaces. The detection of impervious surfaces was done through training a deep learning model on temporal data.

OSM data were used to create labels and then train different convolutional neural networks on Landsat 8 data [Parekh et al., 2021]. [McGlinchy et al., 2019] use a convolutional neural network on World view 2 imagery, achieving results which outperform the Built-Up Extent and Automated Land Use Land Cover tasks from the Digital Globe's Geospatial Big Data Platform [Potapov, 2022].

In comparison with other studies, the focus of this research is to assess if an automatic

label generation from open-source data is sufficient to detect impervious surfaces or if manual labelling (better labels) are needed for an accurate classification. Furthermore, very high resolution multi spectral imagery (red, green, blue, nir-infrared) is combined with a normalized digital elevation model in order to help the convolutional neural network to better delineate between impervious and pervious surfaces.

3 Materials and Methods

This chapter delves into the necessary materials for training the convolutional neural network. An overview of the workflow can be seen in figure [4].

The materials utilized, and their acquisition, are addressed in chapter 3.1. Preprocessing of the data is explained in chapter 3.2 before the focus is laid on the deep learning model to detect impervious surfaces in chapter 3.3. Finally, merging and georeferencing of the tile predictions is explained in chapter 3.5

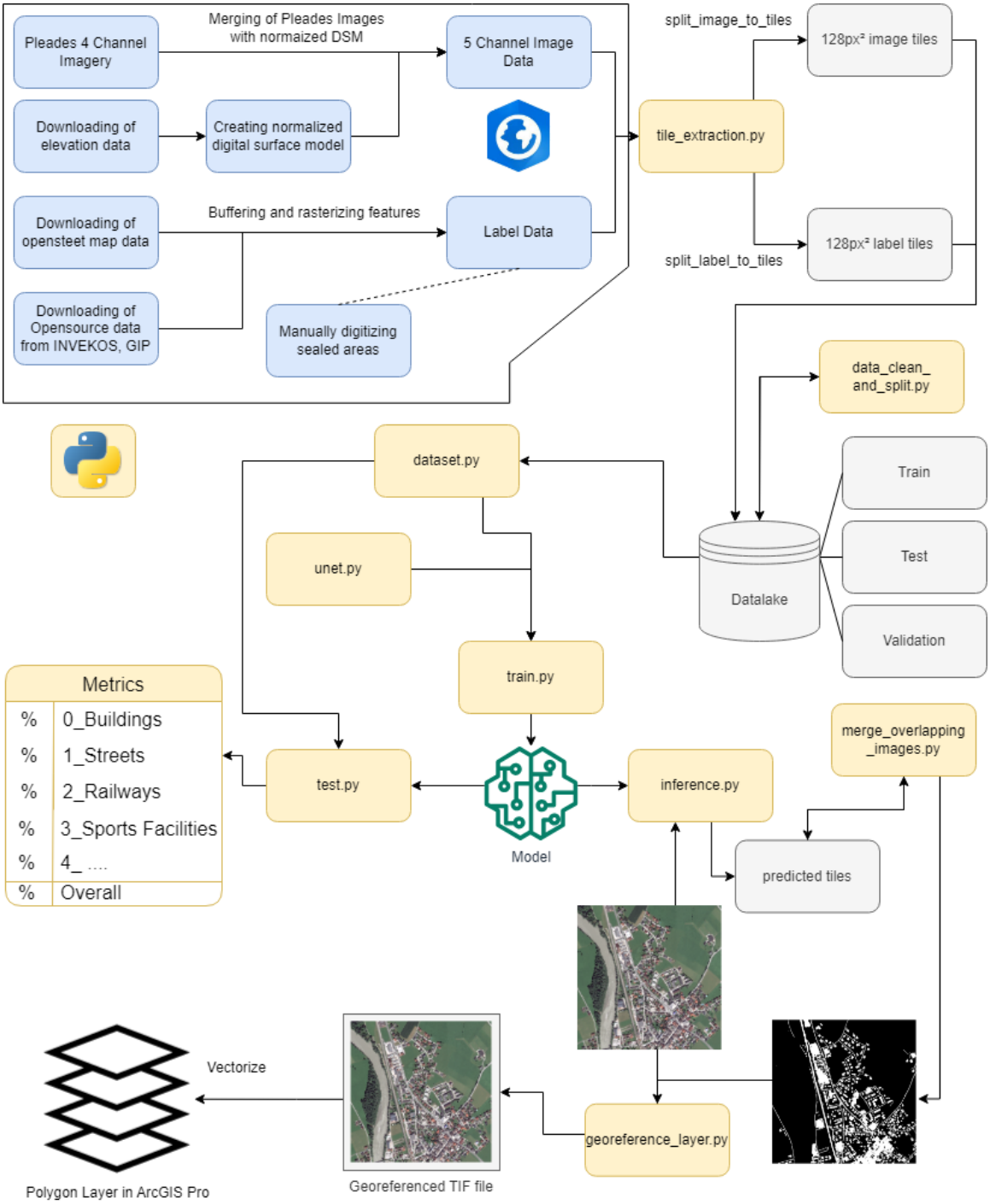

FIGURE 4: Visualization of the workflow.

3.1 Datasources

The subchapter Datasources provides information on the image data and its enhancement, as well as the generation of label data to later detect impervious surfaces. The chapter is divided into imagery and label data.

3.1.1 Imagery

The imagery was acquired from the Pléiades 1-A and Pléiades 1-B twin satellites, which are designed for both civilian and military use. The image product in use is the pansharpened multispectral image with a 50 cm^2 pixel resolution. This very high ground resolution helps to accurately delineate greenery planted between two roads or house boundaries from their gardens, making it suitable for a fine scale analysis. Furthermore, the satellite images have been orthorectified. The multispectral bands consist of Red (600-720 nm), Green (490-610 nm), Blue (430-550 nm) and Near-Infrared (750-950 nm) bands [Airbus, 2023]. An excerpt of the Pléiades image data, in R,G,B and Nir,R,B can be seen in Figure 5. To further enhance the imagery, digital elevation data was downloaded. The Austrian government collects elevation data through airborne laser scanning. The flights over Salzburg and Hallein were tasked in 2019. The digital surface model (DSM) as well as digital terrain model are served per municipality. The mosaics are made freely available to download under https://www.salzburg.gv.at/api5/datalinq/report/vektorwork@alsdownload@alsdownload?.

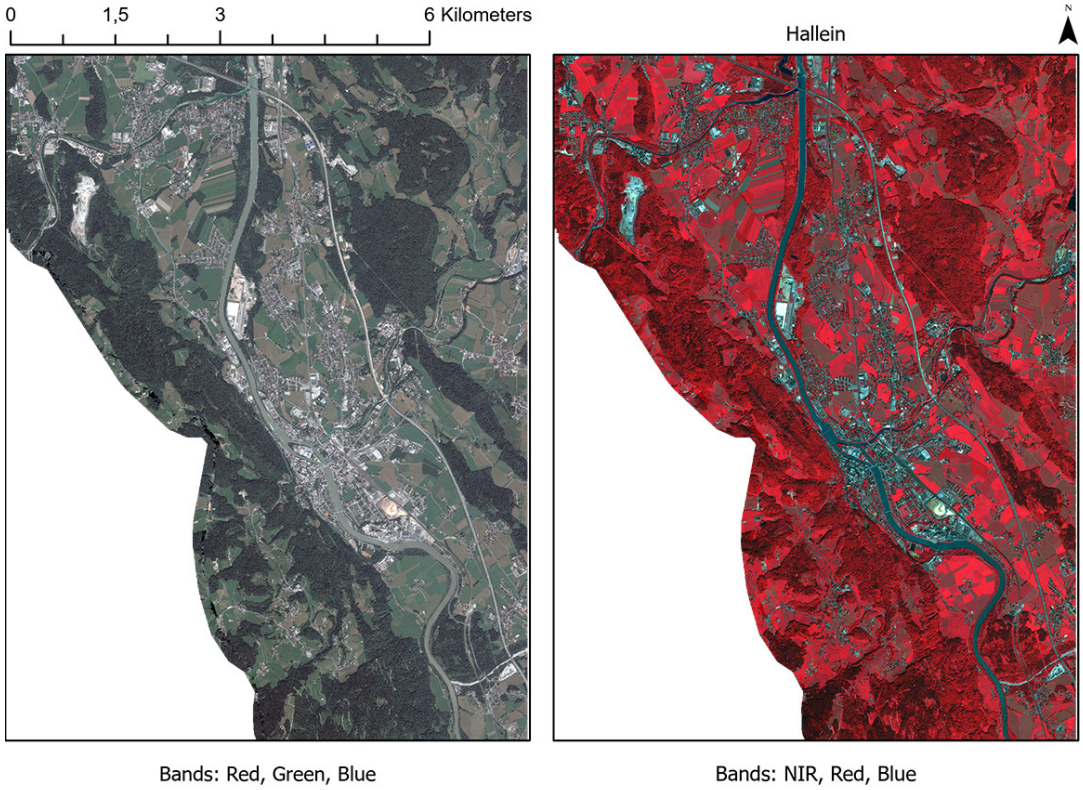

FIGURE 5: Excerpt of the Pleiades Image over Hallein.

To fill the area of interest over Hallein, 5 different overlapping tiles were needed. The elevation data resolution is 1 m^2, making them four times coarser than the Pléiades imagery. Although the resolution is lower, the information might still be beneficial to better delineate houses and roads from features next to them. Figure 6 shows a visualization of the elevation.

Houses, trees and landlines as well as natural elevation are clearly distinguishable. The elevation data provides the neural network with additional input data, which should help the model in finding ways to better delineate pervious from impervious surfaces. All DSM tiles had to be pre-processed to properly stitch them together. Looking at the statistics of the DSM raster, tiles can contain values ranging from -37233 to 9999. The unit of measurement is meters. The extremes indicate null values in a numerical representation. This is problematic for the fusion of multiple tiles in ArcGIS, since the "Mosaic" function will either use the minimum, maximum, mean, horizontally weighted calculated, first or

last value of the overlapping raster cells depending on the "Mosaic Operator" used.

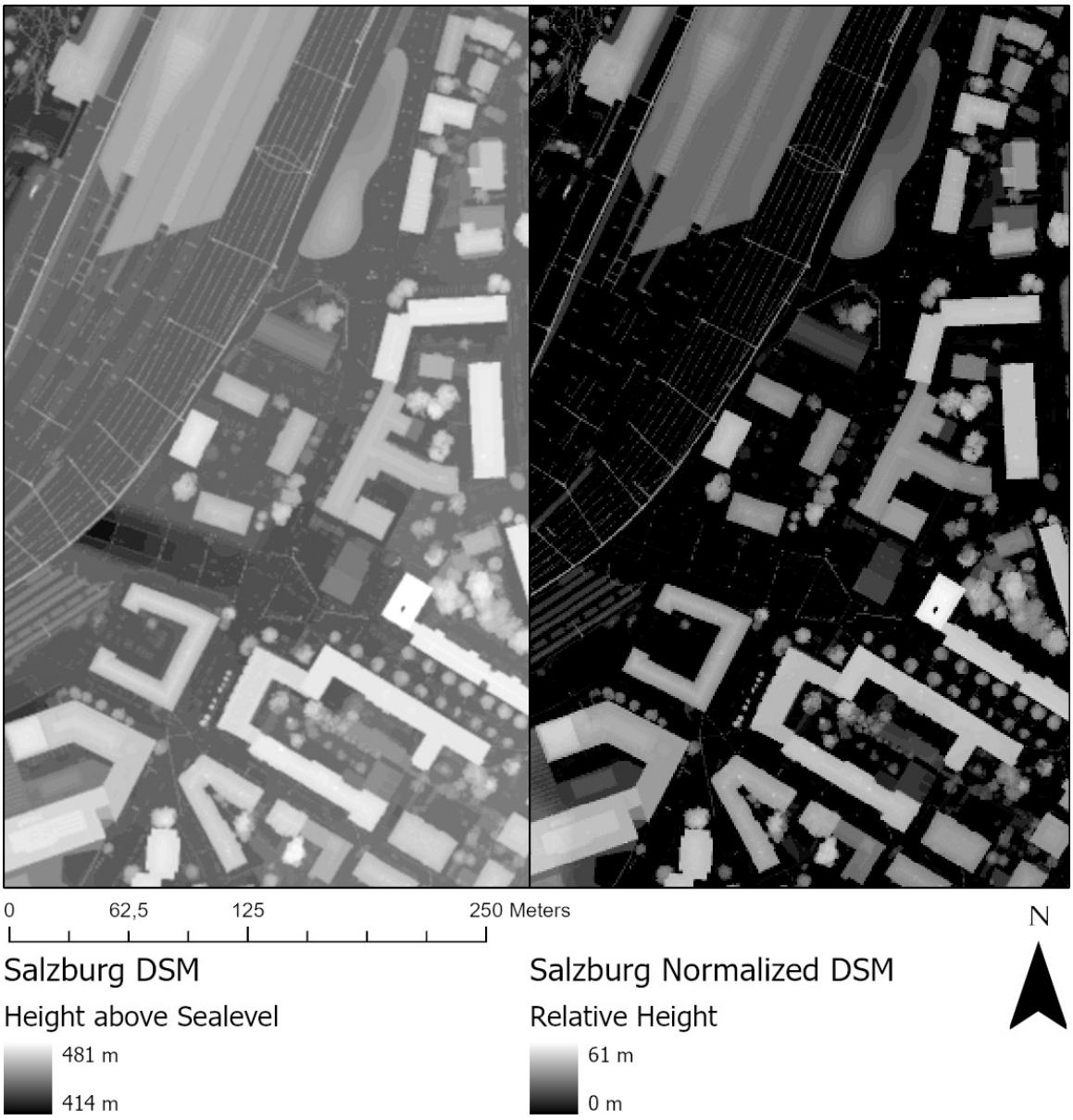

FIGURE 6: Visualization of the normalized DSM in ArcGIS.

To clean up the rasters, the "Raster Calculator" was used. With the SetNull("raster_name.tif" < 0, "raster_name.tif") expression, each cell was either set to "Null", or if "raster > 0" evaluated false, to the original value. This was repeated with values above 4000. The result is cleaned-up raster tiles, which can subsequently be used as input values for the "Mosaic" function. The target of the "Mosaic" function was the respective four channel Pléiades

image, resulting in one raster dataset with five channels (Red, Green, Blue, Nir-Infrared, nDSM).

In order to get the normalized digital surface model, the digital terrain model is subtracted from the digital surface model. The normalized digital surface model shows the relative height to the underlying ground surface. Using the normalized digital surface model ensures uniformity in the elevation data.

Upon visual inspection of the created nDSM, three gaps in the data were found. In the north of Salzburg, one row of pixels was flagged as null value data. A user error could be ruled out since the whole of Salzburg was delivered as one tile. The other two gaps were located in Hallein. These areas will later confuse the learning algorithm of the CNN. Since no official data was available, the area containing "noData" values was not exported for further use.

The resulting raster datasets were exported as 16 bit TIF files, containing 5 channels (Red, Green, Blue, Nir-Infrared, nDSM).

3.1.2 Label Data

In order to train the convolutional neural network, labels for the image data are needed. The generation of labels poses a downside in today's deep learning endeavours [Hoermann et al., 2018] and is often the most time-consuming part [Najafabadi et al., 2015]. Manual labelling is time/cost intensive, but can generate precise labels. While Automatic label generation is comparatively fast and can generate large numbers of label sets, it poses its own problems. If no fitting data for the problem set is available, data which approximates the problem best, has to be chosen. Also, systematic errors will be introduced if label data is not available for all areas which it should represent.

First, the semi-automatic labelling approach is developed. Later, the semi-automatically generated labels are enhanced through manual digitalization of sealed areas. The generated label sets from each approach will later be used as input for a convolutional neural network,

in order to see which approach performs better, and if manual digitization is needed to achieve reasonable results.The manual digitization is expected to reduce systematic errors in the dataset and improve the model's ability to correctly classify sealed surfaces.

In order to query large amounts of open data, regarding the perviousness of surfaces, Open Street map was used. Open street map is a mapping service which relies on prosumers to map objects, like buildings, forests or streets visible from satellite imagery. The data quality is at the will of each user, who adds to the open street map database. Since there is no reward for doing so, it is expected that the users have an intrinsic motivation to further improve and enrich the mapping service, hopefully resulting in accurately placed objects. The amount of objects placed in the map is varying between regions. A study of comparing OSM buildings to buildings from official sources in Germany and Austria showed, that OSM buildings had an intersection of 97 % with their official counterpart [Dorn et al., 2015]. The study also showed, that the classes of buildings and streets had a far higher completeness compared to greenery or forests. This suggests that OSM is a good starting point to extract open source data for impervious surfaces.

Additional challenges regarding the data quality can be a temporal difference between the image data and the Open Street Map database. Since the image data is dated to the year 2019, construction might have changed the landscape, resulting in either features which are in the OSM database but not on the satellite imagery or the other way around. This problem should only be marginal and is likely to be mitigated by the amount of data. With enough label data, these false positives will not confuse the neural network.

In order to query the OSM database, overpass turbo was used. Overpass turbo is a web based exploratory data querying tool for the OSM database. Two queries were created. One to get all features which represent sealed surfaces and the second one to get green spaces which are not sealed.

The first query can be seen in listing 1.

```
[out:json][timeout:25];
(   way["building"]({{bbox}});
```

```
    way["highway"]({{bbox}});
    way["aeroway"="runway"]({{bbox}});
    way["aeroway"="taxiway"]({{bbox}});
    way["surface"="concrete"]({{bbox}});
    way["amenity"="parking"]({{bbox}});
    way["aeroway"="apron"]({{bbox}});
    way["aeroway"="hangar"]({{bbox}});
    way["railway"="rail"]({{bbox}});
    way["landuse"="railway"]({{bbox}});
);
out body;>;out skel qt;
```

LISTING 1: Overpass turbo API call to query roads, buildings, airport structures, railways and concrete surfaces.

The API call selects all ways, which can be line or polygon features, which contain the attribute mentioned in the square brackets. If the attribute is set equal to a specific string, only features with that string will be returned. The API call tries to get all surfaces which are impermeable. Resulting in buildings, all streets, railways, airport buildings and runways, parking spaces as well as all surfaces tagged as concrete.

Through visual analysis, it seems that most buildings have been extracted by the query. Although, a systematic gap in the sealed features becomes evident. Driveways into residential homes, as well as the sealed parking spots in front of the homes, are not available through OSM. OSM does not have them as features. The systematic error could be a large problem for the correctness of the classification. This problem can be addressed later through manual labelling.

The OSM streets seem to be very comprehensive, with the drawback that they are only saved as line features. In order to create a raster dataset of sealed and unsealed surfaces, 2D features are needed. The line features need to be buffered with a suitable width to create an approximation of the street's width. The streets layer consists of small footpaths, highways and residential streets. Since these roads all vary in size and only about 5% of them have a width information, a python function [2] was created which uses their tag and number of

lanes to calculate an approximation of the width divided by two. The approximated width was divided by two, since it will be used in the ArcGIS Pro buffer tool, buffering the streets by the value on both sides.

```
def Calc_width(highway,railway,lanes):
    width = 0
    if (highway == "bridleway" or highway == "footway"
        or highway =="steps"):
        width = 2
    elif (highway == "cycleway" or highway == "path"):
        width = 3
    elif (highway == "service" or highway == "platform"
        or highway == "bus_stop"):
        width = 4
    elif (highway == "living_street" or highway == "motorway_link"
        or highway == "primary" or highway == "primary_link" or highway
        == "proposed" or highway == "residential" or highway ==
        "secondary" or highway == "secondary_link" or highway ==
        "tertiary" or highway == "tertiary_link" or highway ==
        "track" or highway == "unclassified" or highway ==
        "pedestrian" or highway == "construction" or highway ==
        "corridor" or highway == "trunk" or highway == "trunk_link"):
        width = 5
    elif (highway == "motorway"):
        width = 10
    elif(railway == "rail" or railway == "razed"):
        width = 4
    else:
        raise ValueError("Type unknown : " + highway)
    if(lanes != None):
        if(lanes == "0"):
            width = width
        else:
            width = width + (3.5  * int(lanes) -3.5)
    return width / 2
```

LISTING 2: Python code to calculate the width of line features according to their tag and amount of lanes.

Buffering the line features results in mostly fitting street widths. With an automatic approach, some overfitting and under fitting scenarios can not be avoided. In order to mitigate the under fitting scenarios, a second street dataset was used.

The second data source containing streets was the Graph Integration Platform (GIP). The GIP is the digital transport graph of Austria. Openly available through the open government data initiative, the GIP becomes a great source for open source data regarding streets. A benefit of the GIP data is, that it is maintained by the government and that each feature has an attribute containing a value for its width. When using the street width information to buffer the features, it was noticed that it fits well in the region of Hallein, and is often under fitting in Salzburg. This is due to the densely sealed city environment in Salzburg, which often has asphalted parking places, bike ways and walkways on both sides of a street, effectively doubling its sealed area. This led to the decision to treat the street data differently in Salzburg than in Hallein.

In Salzburg, the width attribute which was available in the GIP dataset was not divided by two, resulting in streets with double the width. In Hallein the width was halved, so that the original width of the street is created. This results in good fitting streets in the rural parts of Hallein, with some under fitting in the city center. In Salzburg it is the other way around. The city center has good fitting streets, while in the rural parts there is overfitting.

The benefit from using both datasets becomes apparent when they are overlaid with each other. Slightly different edge points lead to a more accurate depiction of street intersections. Furthermore, OSM data was more comprehensive regarding service roads or small roads leading up to residential buildings (Figure [8]).

In order to mitigate the overfitting, known green spaces can be deducted from the buffered street lines.

Green spaces from OSM were acquired by the second overpass turbo API call, which

FIGURE 7: Left: Streets buffered by their original size. | Right: Streets buffered by double their size.

can be seen in listing 3.

```
[out:json][timeout:25];
(
  way["leisure"="park"]({{bbox}});
  way["landuse"="forest"]({{bbox}});
  way["landuse"="village_green"]({{bbox}});
  way["landuse"="grass"]({{bbox}});
  way["landuse"="meadow"]({{bbox}});
  way["leisure"="pitch"]["surface"="grass"]({{bbox}});
  way["landuse"="farmland"]({{bbox}});
  way["leisure"="playground"]({{bbox}});
); out body;>; out skel qt;
```

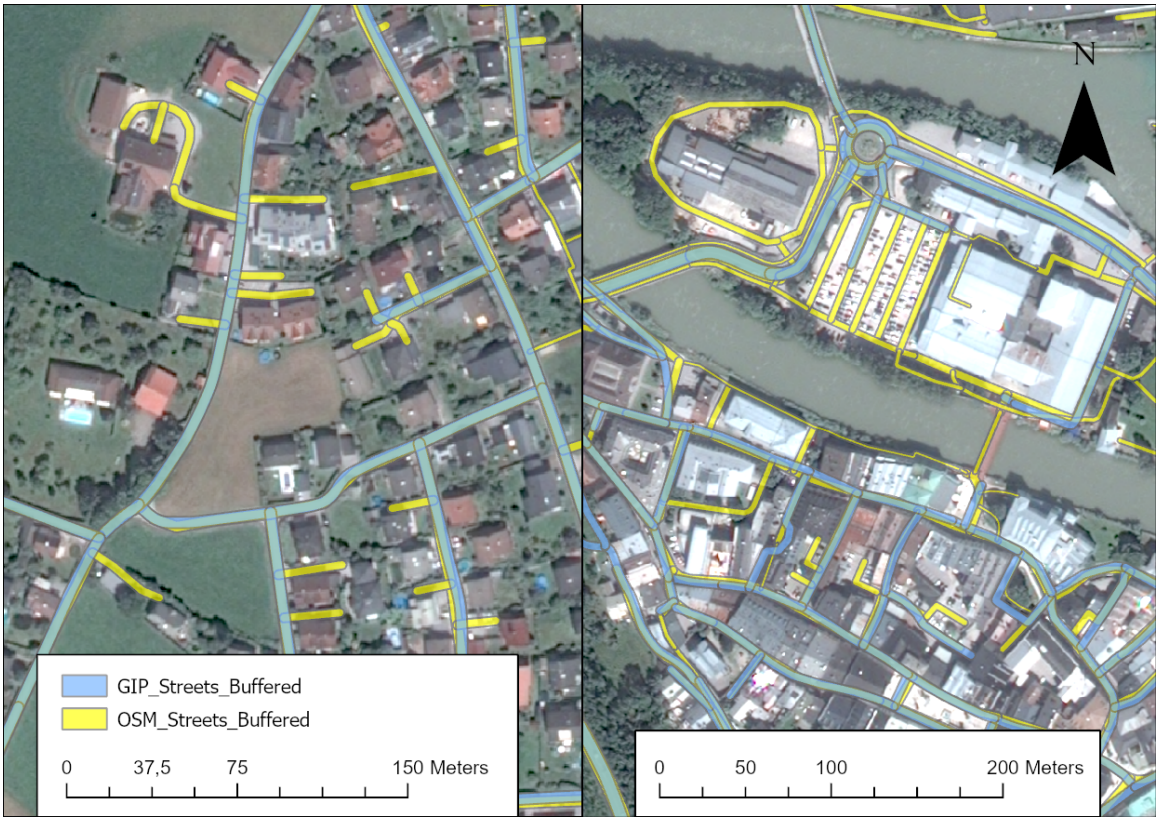

FIGURE 8: Buffered OSM streets in yellow and GIP streets in blue overlaid with each other.

LISTING 3: Overpass turbo API call to query parks, forest, grass, meadows, greenery, farmland and playgrounds.

Additionally, parcels from the Integrated Administration and Control System (IACS) were used. The IACS is a system to control payments to farmers. Part of this system is a European wide database of parcels, which are used for agricultural purposes [Burtscher, 2023]. The deleting of overlapping streets into greenery was done in ArcGIS Pro. With the use of the geoprocessing tool "Erase Features" greenery parcels are subtracted from the buffered streets. The subtraction of green spaces reduced the area labelled as sealed in Salzburg from ca. 15.5 km^2 to 15 km^2. In Hallein, the same process erased about 0.9 square kilometres from 10.6 km^2 to 9.7 km^2

After the all streets which overlap into greenery parcels are adjusted, the polygons, marking impervious surfaces, are merged together. Finally, an additional attribute is added

Sealed Area Labels			
Area of Interest	Roads and Buildings	Greenery Subtracted	Manually Enhanced
Salzburg	15.540 km²	15.016 km² (-3.37 %)	16.856 km² (+12.26 %)
Hallein	10.587 km²	9.711 km² (-8.27 %)	11.282 km² (+16.17 %)

TABLE 1: Table showing the square meters of sealed areas in Hallein and Salzburg, as well as the square meter change from the left neighbouring cell.

to the merged feature class. The added attribute has a default value of 255. The value differentiates between sealed (255) and unsealed (null) areas. It will later be used while converting the feature class into a raster file.

This concludes the steps needed to generate automatic label data as a polygon format.

The manually labelled dataset only added to the automatically generated data, while taking none of the spaces away. This was done due to a few reasons. The automatic approach largely underfitted the sealed areas in the imagery, eg. no driveways being available, missing industrial sites and parking spots. So the main improvements could be made focusing on these areas. Secondly, there is an over-representation of unsealed areas in the data, due to the nature of the landscapes. The area of interest has many meadows, fields, parks and forests. This is especially visible in Hallein, but also in Salzburg. This should allow the model to be confident in the detection of these areas.

The satellite image was used as a background to be digitized on. The image bands were set to near-infrared red and green in order to highlight vegetation. After manually drawing boundaries of sealed features, the resulting layer of Salzburg had a 12 % increase in sealed surface area, than compared to the automatic approach. In Hallein, the sealed area increased by 16 %. This shows that parking spaces, driveways and industrial sites being left out in the automatic approach are quite significant.

With both, the automatically generated base labels and the manually digitized enhance-

ments available as polygons, the feature classes can be merged together in ArcGIS Pro.

Finally, the resulting layers will be rasterized to the same resolution as the image data. A 1 bit raster was chosen as a raster type, since only a distinction between sealed (True) and unsealed (False) was needed. Choosing the appropriate data format keeps the load on the hard drive minimal and speeds up the training process. The resulting Raster data set was then exported as a TIF file for further processing.

3.1.3 Random Sample Points

In order to see which model performs the best, and where each model struggles to correctly classify sealed or unsealed surfaces, a stratified random sampling approach is chosen.

500 points are distributed among 10 groups. Each group represents distinct areas in the imagery. The 10 groups are as follows: streets, buildings, railways, sports facilities, driveways, mineral/dumping sites, forests, pastures, arable land and water. The first 6 groups represent sealed areas. The last 4 groups represent areas which are unsealed. Sport facilities fall under the sealed category since tennis courts, running tracks or basketball fields, all of which are sealed, were selected as possible areas for random sample points.

The points per group are split between Salzburg and Hallein. Out of the 50 points, 30 lie in Hallein and 20 in Salzburg. This was done due to the size of the training areas. There is one exception to this rule. The mineral/dumping sites group could only be found in Hallein, therefore all 50 Points lie in Hallein. This splits the 500 points of the test data set between the regions, with 180 points in Salzburg and 320 Points in Hallein.

In order to generate random points, polygon layers, containing valid point locations, are created for each group. The Copernicus Urban Atlas 2018 dataset was used as a base for some of the classes. The Urban Atlas dataset contains 27 classes, ranging from different densities of urban fabric to multiple nature classes. The dataset is created from 2m to 4m pixel resolution input data [European Environment Agency, 2023]. This results in a more coarse resolution than the image data that the CNN model is working with. In order not to

have pixels which belong to a different class, each point was manually looked at. Points that showed a different class than the class they were created for, were moved to the next clear pixel of that class. The classes railways, mineral/dumping sites, forests, pastures, arable land and water were extracted from the dataset and required minimal adjusting.

The polygon layers for the streets and buildings group were constructed using the streets and buildings label data.

The class sports facilities is created from manually labelled polygons of the facilities.

The class driveways is manually set across the image data. This had to be done since it was not feasible to manually digitize each driveway across the whole dataset.

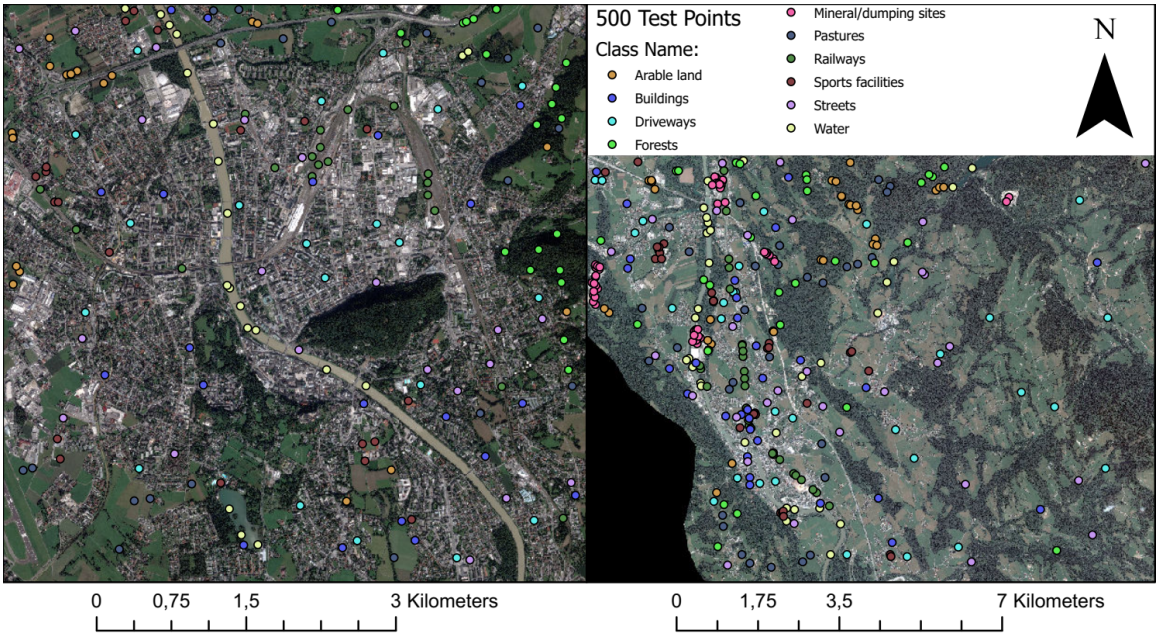

FIGURE 9: Visualization of the 500 Points in ArcGIS Pro. The left map shows the AOI of Salzburg, the right over Hallein.

After all possible point regions have been defined, the Create Random Sample tool is used with the regions as a limiting boundary, to create the points.

Each point needs to have its local image coordinates as an attribute, in order to later remove the tiles, containing test points from the training's dataset via python. The pixel coordinates are also needed when an interference is run on an image tile, surrounding

the specified point. The calculation of the pixel coordinate can be seen in listing [4]. The geographic coordinate system MGI / Austria GK M31 with the EPSG code of 31258 eases the conversion from geographic coordinates to pixel coordinates, since the unit of measurement is 1 meter [Klokan Technologies, 2007]. This, combined with the half meter pixel resolution, provides a simple formula for conversion.

```
X = ((X Coordinate in EPSG:31258) - (Top Left X Coordinate of AOI)) * 2
Y = ((Y Coordinate in EPSG:31258) - (Top Left Y Coordinate of AOI)) * -2
```

LISTING 4: Arcgis Mock-Code to calculate the image coordinate.

The points are finally exported as a .csv file for further processing.

3.2 Data preprocessing

The exported image and label TIF files have to be further processed in order to be used as an input for a neural network.

The neural network takes tiles of the whole image as an input. The input tile size for the network was configured to 128 px^2 and a five channel depth. In order to use the images in Python, they were loaded using the tiffile library and converted to a NumPy array. A function [5] to split the images and labels was written. The function takes a NumPy array representing the image and iterates over the whole array with a step of 128 px.

```
def split_image_to_tiles(image:np.ndarray,path:str,
                         offset_width:int=0,offset_height:int=0):
    img = image
    im_h, im_w = img.shape[0],img.shape[1]
    ti_h, ti_w = 128, 128


    for row in np.arange(im_h - ti_h + 1, step=ti_h):
        for col in np.arange(im_w - ti_w + 1, step=ti_w):
```

```
tile = img[row:row + ti_h, col:col + ti_w]
tifffile.imwrite("datalake/train/"+ path +"/images/image_tile_"+
str(col + offset_width) +"_"+ str(row + offset_height) +
".tif",tile.astype(np.uint16))
```

LISTING 5: Python function to split the Pléiades image into 16bit 128px * 128px tiles.

Each image tile is then saved with its local array coordinates of the top left corner in the name. The naming scheme allows removing the tiles which contain the test points, as well as to combine the image tiles back together.

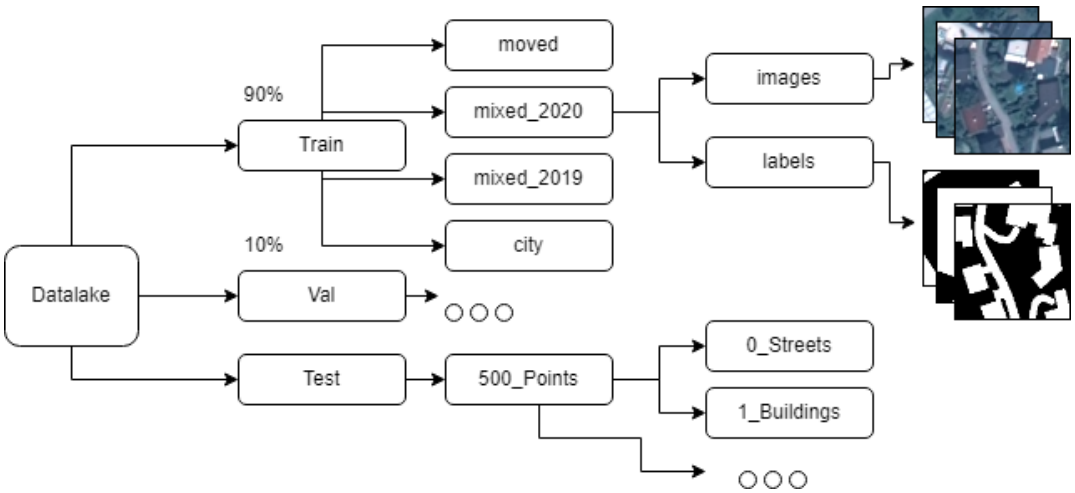

FIGURE 10: Image data structure and percentages of training, validation and testing data.

In order to sort and store the image files, the folder structure seen in figure [10] was set.

The *mv_training_tiles* function extracts the point location from the points CSV file and moves the tiles which contain the respective point. This ensures that the point which is used to infer the model's accuracy has not been used in the training data. Then a python function randomly chooses images to relocate them according to two possible use cases, training and validation. Out of the whole image pool, 90 % of the images were used for training. 10 % were used to continually validate the model's performance during the training process.

3.3 Deep learning for impervious surface detection

In this work, the TensorFlow framework was used to train multiple convolutional neural network models. TensorFlow is a machine learning framework developed by the Google Brain team, which can be used for many applications, but its main focus lies on working with deep neural networks [Abadi et al., 2016].

A commonly used network architecture for semantic segmentation is the UNet architecture [Ronneberger et al., 2015]. The ternaus-v2 network, which is also a UNet, made the second place in the DeepGlobe Building Extraction Challenge [Iglovikov et al., 2018]. Therefore, the details of the used architecture, as seen in Figure 11, are inspired by the ternaus-v2 network.

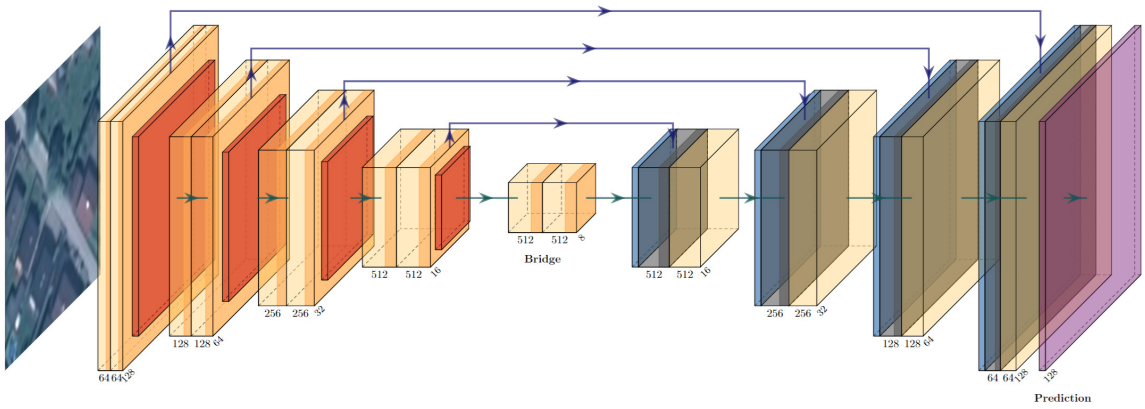

FIGURE 11: The UNet model architecture used.

The model can be split into four parts, the encoder, bridge, decoder and output (see listing 7).

The encoder takes the input image and encodes it to a smaller representation, containing the main features of the input data. It consists of several parts. The 'conv_block', see in listing 6, performs two times a 3x3 convolution followed by a batch normalization layer [Ioffe & Szegedy, 2015] and the rectified linear unit activation function [Agarap, 2018]. The ReLU is the most common activation function in use today. For negative values, the function evaluates to 0 while for 0 and positive values, the function evaluates to the

given value [see Figure 3]. One additional training benefit of this function is that the computational burden of the network is lowered. Some neurons will be deactivated (set to null), thus decreasing the computational load.

Finally, after each 'conv_block' a max pooling layer is introduced that reduces the spatial resolution by a factor of two, while the number of filters is increased by the same factor. This sequence of operations is performed four times until the resolution is reduced from 128x128x4 to 8x8x512.

The bridge repeats the last convolution block. The same process, as described in the encoding, is repeated in the decoding, with the difference being, up sampling instead of max pooling. Each encoder and decoder level is connected through skip connections [PapersWithCode, 2022] which concatenate the encoder output with the decoder input. This allows the model to directly retain information of previous layers and learn better. Through the process of decoding, the resolution from 8x8x512 is again upsampled to the full image resolution of 128x128x64.

A final convolution with a 1x1 kernel and the sigmoid activation function is applied to receive an output of 128x128x2. The two channels correspond to the impervious surface class and the pervious surface class. The sigmoid activation function ensures that our output is mapped between 0 and 1, which is optimal for a binary classification.

As a loss function, the binary focal loss was chosen. The binary focal loss addresses the foreground background imbalance, which is significant in our dataset, since most areas are unsealed. Through down-weighting the plentiful unsealed areas, the detector is forced to focus more on sealed areas [Lin et al., 2017]. Therefore, predicting a sealed area wrong, will produce a higher loss than predicting an unsealed area as sealed. This hinders the network to not predict everything as unsealed to reach a low loss score.

Adam was used as an optimizer algorithm for the network, which has been proven to work well compared to other stochastic optimizers [Kingma & Ba, 2014]. The learning rate was set to 0.001 and the batch size, which is the number of images that are provided to the

network at the same time, is set to 8. For the model evaluation metric, the Jaccard-Index which is commonly known as the intersection over union was regularly calculated to monitor the training.

```
def conv_block(x, num_filters):
    x = Conv2D(num_filters, (3, 3), padding="same",
    data_format="channels_first")(x)
    x = BatchNormalization()(x)
    x = Activation("relu")(x)
    x = Conv2D(num_filters, (3, 3), padding="same",
    data_format="channels_first")(x)
    x = BatchNormalization()(x)
    x = Activation("relu")(x)
    return x
```

LISTING 6: Python function to perform the convolution block twice.

The models are trained locally on a GeForce RTX 3060 GPU with 12 GB of onboard memory and a TensorFlow version of 2.9. 32 GB of random access memory allows storing the whole training set in the memory. It is run with 5 channels (R,G,B,Nir,Normalized_DSM) as input.

First, three models are trained on all datasets, with three different sizes. The depth of the model will range from 16x16x256, 8x8x512 to 4x4x1024. Smaller models will be able to train faster, while the larger models, which go deeper, should be able to retain more information from the problem set. If in this case, larger models perform better and if they are worth their additional resources will be seen.

The UNet model which performed the best is chosen to train the following CNN.

First a model with automatically generated labels and the previously determined UNet architecture is trained. Then, the manually enhanced dataset is used to train a second model. Both models will be tested on the 500 random test points. Each point will be in the middle

```python
def build_model(channels=4, size=128, classes=2):
    num_filters = [64, 128, 256, 512]
    inputs = Input((channels, size, size))

    skip_x = []
    x = inputs

    ## Encoder
    for f in num_filters:
        x = conv_block(x, f)
        skip_x.append(x)
        x = MaxPool2D((2, 2), data_format="channels_first")(x)

    ## Bridge
    x = conv_block(x, num_filters[-1])

    num_filters.reverse()
    skip_x.reverse()

    ## Decoder
    for i, f in enumerate(num_filters):
        x = UpSampling2D((2, 2), data_format="channels_first")(x)
        xs = skip_x[i]
        x = Concatenate(axis=1)([x, xs])
        x = conv_block(x, f)

    ## Output
    x = Conv2D(classes, (1, 1), padding="same", data_format="channels_first")(x)
    x = Activation("sigmoid")(x)

    return Model(inputs, x)
```

Listing 7: Python function to build the UNet model.

of a 128px^2 image tile. The predicted tiles, and if the middle was predicted as pervious or impervious, will be saved and compared. This should show if the manually generated labels are needed in order to improve the model's performance to a sufficient level, or if the automatic approach alone is enough.

A look at the confidence of the better model in regard to its predictions is made. Mapping the confidence of the predictions, should show in which regions the model has difficulties

to make confident predictions.

Finally, the model is used to predict soil sealing on larger, previously unseen images. One image is in the south of Hallein and was tasked in 2020. The other is an image from the Großarl region, which was tasked in 2019. This is done to test the real world applicability of the model and to complete the workflow. The predicted layers will be overlaid with the images in ArcGis Pro and visually analysed. The output soil sealing maps are added in the appendix.

The complete python project with all helper functions and scripts can be accessed under https://github.com/MxBeckm/soilsealingCNN.

3.4 Inference and testing of the model

In order to use trained models to generate predictions, the inference script was written. The inference script consists of three main functions. The functions *inference*, *learining_series* and *inference_with_overlapp*. The function *inference* [Listing 8] is used to infer over one tile.

```
def inference(model, image):
    return model(image, training=False)
```

LISTING 8: inference function.

This function is a wrapper to use the model on an image and return its prediction.

A function *learning_series* which is used to visualize the learning series of one tile over all saved training steps. Its inputs are the model, image tile and number of iterations.

The function inference_with_overlap [Listing 9] which is used to split on large input image into tiles which overlap by a certain factor.

An inference is done on every tile and the tiles are saved in order to be stitched together later.

The testing script is used, after training the model. It tests the model's performs on the test dataset and extracts relevant information about the model.

```
def inference_with_overlap(model, full_image_path, factor):
    os.makedirs(f"{OUTPUT_FOLDER}", exist_ok=True)
    full_image = load_tiff(full_image_path).astype(np.float32)
    im_h, im_w = full_image.shape[1],full_image.shape[2]
    ti_h, ti_w = 128, 128
    for row in np.arange(im_h - ti_h + 1, step=int(ti_h/factor)):
        for col in np.arange(im_w - ti_w + 1, step=int(ti_w/factor)):
            tile = full_image[:,int(row):int(row + ti_h)
                   , int(col):int(col + ti_w)]
            tile = tf.expand_dims((tile), axis=0)
            pred = inference(model,tile)
            # plot prediction
            output_image = np.transpose(pred*255, [0, 2, 3, 1])[..., 1]
            Image.fromarray(output_image[0,:, :].astype(np.uint8)).save
                (f"OUTPUT_FOLDER}img_tile_pred_{int(col)}_{int(row)}.png")

```

LISTING 9: Inference with overlap function. This function will iterate over an image and predict soil sealing. The overlapping factor, image and model can be set as parameters.

The main functions are the *setup* function, which prepares the input variables for the *test* script. The dataset is created, Unet is loaded and the logger initialized.

The *test* function will take these inputs and test the model.

Every image from the test dataset will be inferred, the middle of each tile is checked to see if the model predicted the right class for the image. The image, a NIR infrared visualization, a raw prediction, a thresholded prediction and a confidence visualization will be saved for every image. After all images of the test dataset are predicted, a table showing the hitrate per group is plotted to the console, as well as the confusion matrix. Additionally, the specificity, precision, accuracy and error rate of the test are printed out.

The testing setup allows to quickly compare different models with each other.

3.5 Merging and georeferencing of predictions

The predictions of the neural network are being made on adjacent tiles. At the border of the tile, a distinct edge effect can be noticed. The edge effect occurs due to the network not

```python
def test(datasets, model, logger,ROC_Calc=False):
    @tf.function
    def inference(images):
        pred = model(images, training=False)
        return pred
    # evaluate each dataset:
    metrics = {}, y_true = [], y_predicted = [], thresholds = [0.5]
    score_by_class = {class_id:[] for class_id in datasets.keys()}
    for class_id, image_paths in datasets.items():
        correct_classified = 0
        for image_path in tqdm.tqdm(image_paths):
            # load image
            image = load_tiff(image_path)
            image = tf.expand_dims(tf.convert_to_tensor(image), axis=0)
            prediction = inference(image)
            for threshold in thresholds:
                threshold_prediction = np.where(prediction < threshold, 0, 1)
            # calculate if prediction matches the class label
            score_by_class[class_id].append(prediction[0, :, 64,64])
            pred_class_id = np.argmax(threshold_prediction[0, :, 64,64])
            gt_class_id = class_id
            if pred_class_id == class_id_mapping[gt_class_id]:
                correct_classified += 1
            # calculate confussion matrix
            y_predicted.append(pred_class_id)
            y_true.append(class_id_mapping[gt_class_id])

            # plot image, # plot nir, # plot prediction
            # plot threshold_prediction # plot confidence
        metrics[f"{class_id}_hitrate"] = correct_classified / len(image_paths)
        with logger.as_default():
            tf.summary.scalar(f"{class_id}_hitrate",
                              metrics[f"{class_id}_hitrate"], step=1)
    avg = 0
    for class_id, image_paths in datasets.items():
        hitrate = metrics[f"{class_id}_hitrate"]
        table.add_row([class_name_mapping[class_id], f"{hitrate*100.0}%"])
        avg += hitrate
    avg /= len(datasets.items())

```

Listing 10: A shortened version of the test function.

having full neighbourhood information for bordering pixels. In order to fix the edge effect, a function for merging overlapping tiles is written.

The *save_labels_as_layer* function has a boolean flag named overlap, which can be set to true if predictions have been calculated with and overlap. For each 32px^2 quadrant of the prediction area, all filenames for each possible image tile which could overlap into that quadrant will be constructed. Then it is checked if these positions are valid, through an *os.isvalidfile* check to see if the files indeed are stored in the datastorage. The final values for the quadrant are of the sum of all overlapping values divided by the number of overlapping tiles.

This approach ensures that every quadrant, except for the first three quadrants from every border, has been predicted 16 times with different neighbourhoods.

The final image, which is stitched together, will be geocoded using GDAL. GDAL is an open-source geospatial abstraction library and is implemented in this project using its python API [GDAL/OGR contributors, 2023]. The coordinates of the original input image, which is used by the *inference_with_overlapp* function, will be extracted and used to georeference the newly created label layer. This ensures a fit between the image and label data.

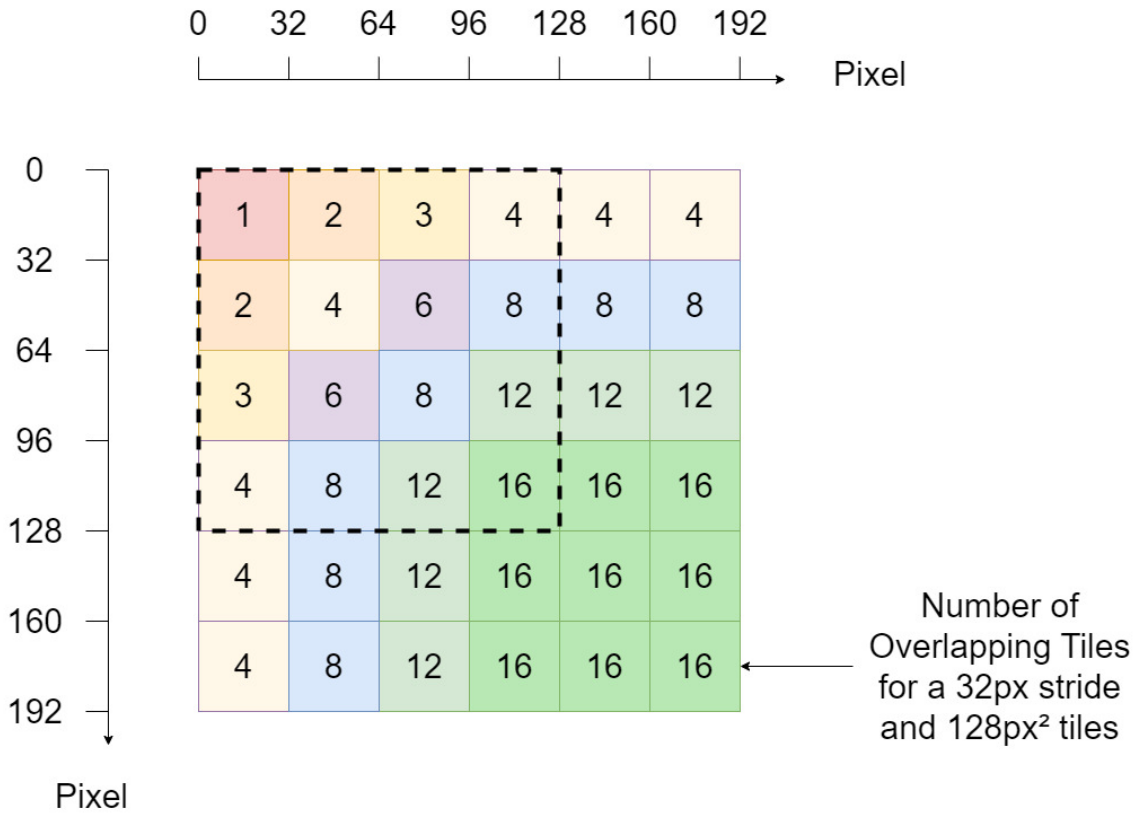

FIGURE 12: Number of overlapping tiles in each 32px wide square of the prediction area.

4 Results

The Intersection over Union from the first three models which determined the model architecture of the next models can be seen in Figure 13. The different model sizes were one with 256, 512 and 1024 filters on the respective smallest resolutions. The first model with 256 maximum filters completed 175 training epochs in 1 day and 17 hours. It achieved a maximum intersection over union of 0.89. The second model with a maximum of 512 filters completed 175 training epochs in 2 days and 5 hours. It achieved a maximum IoU of 0.927. The third model with 1024 filters completed its 175 training epochs in 2 days and 23 hours. It achieved a maximum IoU of 0.922. The model using 512 filters achieved the highest IoU out of the three trained models and trained faster than the largest model.

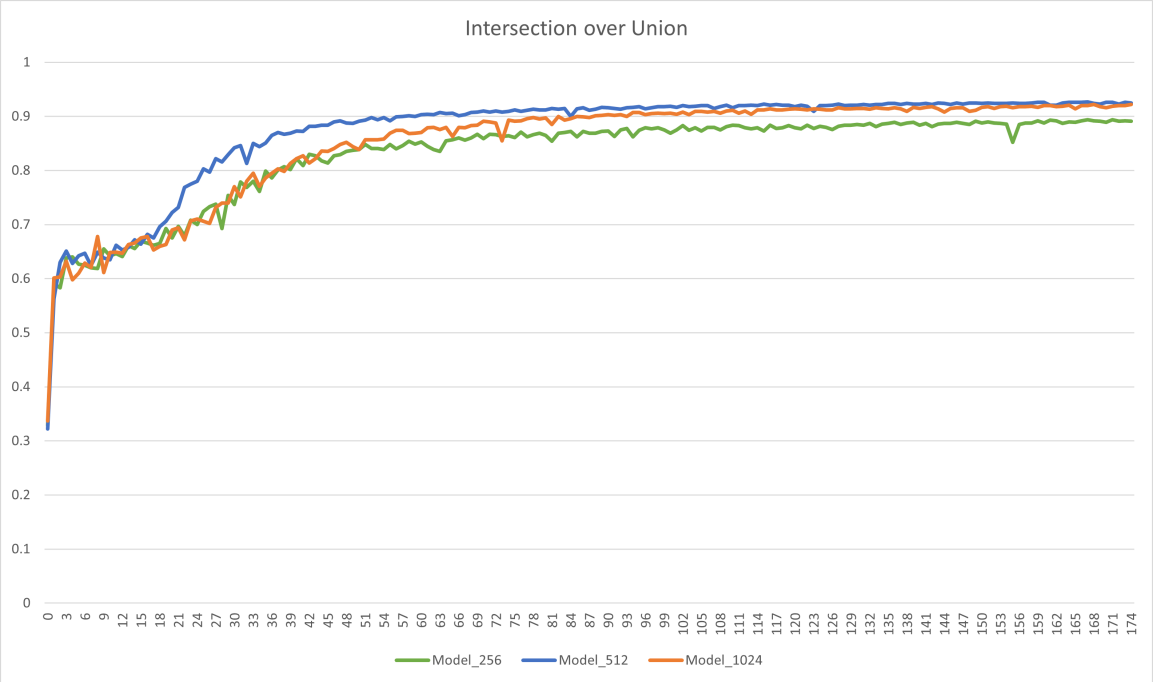

FIGURE 13: Training curves of three models with the automated dataset and different number of filters.

Although, the differences in IoU between the second and third model are only marginal, the faster training time as well as the better test score decided that the following models will be trained using the model architecture of the model with a maximum of 512 filters. In this case, using a deeper model with more parameters did not yield a better performance.

The following trainings use a model architecture with a maximum of 512 filters. Two models were trained with the same parameters, except for different label sets.

Model 1 (Automated) was trained using the semi-automatically generated labels. Model 2 (Handmade) was trained using the manually enhanced labels.

Both were tested on the random sample points dataset. A threshold for prediction was set by plotting the accuracy of the model with thresholds between 0 and 1 using a 0.05 increment. Both models performed best on the test dataset when a threshold of 0.5 was used. A threshold of 0 would classify everything as unsealed, while a threshold of 1 would classify everything as sealed.

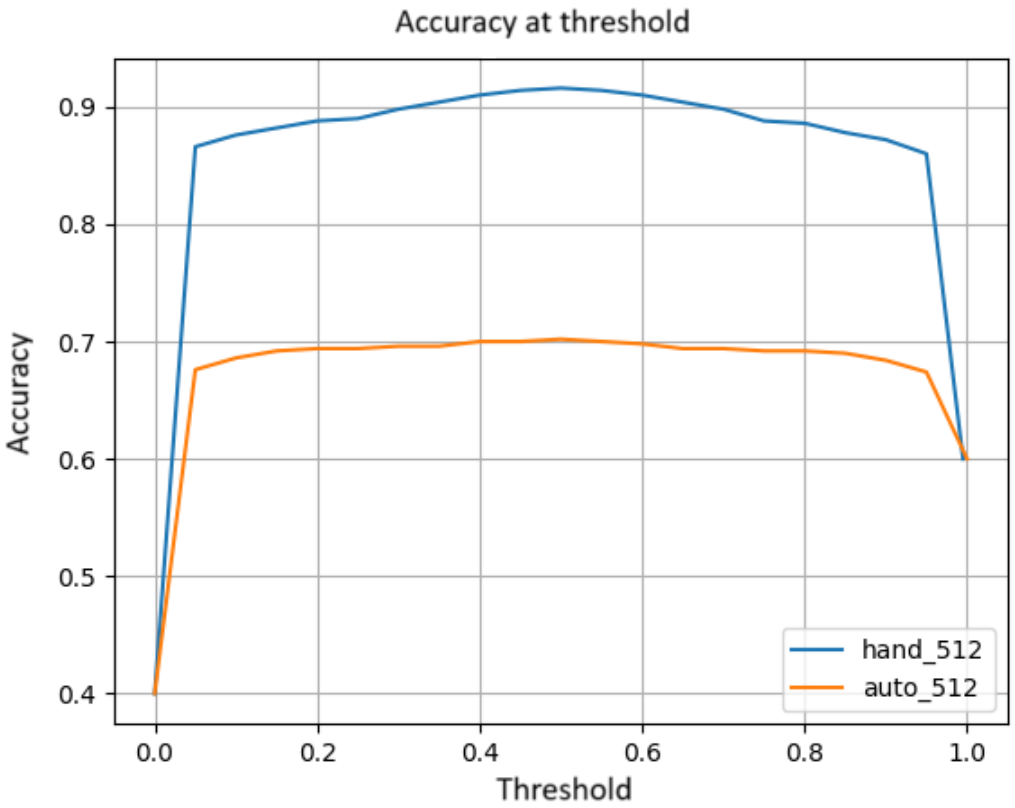

FIGURE 14: Accuracy of the models with thresholds between 0 and 1. The automated model is coloured orange and the handmade one is coloured blue

Model 1 achieved a maximum intersection over union in training of *0.927*. Model 2 achieved a maximum intersection over union of *0.925*. Both models were then tested on the 500-point dataset. Model 1 achieved a hit rate of *70.2%* while Model 2 achieved a hit rate of *91.6%*. A confusion matrix for both models can be seen in Figure 15.

CNN Models		
Point Group	Model 1 (Auto)	Model 2 (Hand)
0 Streets	96.0%	98.0%
1 Buildings	88.0%	92.0%
2 Railways	92.0%	98.0%
3 Sports Facilities	0.0 %	76.0%
4 Driveways	20.0%	66.0%
5 Industrial Sites/Dumping	8.0%	88.0%
6 Forest	100.0%	100.0%
7 Pasture	98.0%	98.0%
8 Arable Land	100.0%	100.0%
9 Water	100.0%	100.0 %
- Overall	70.2%	91.6 %

TABLE 2: Table showing the test results of model 1 and 2.

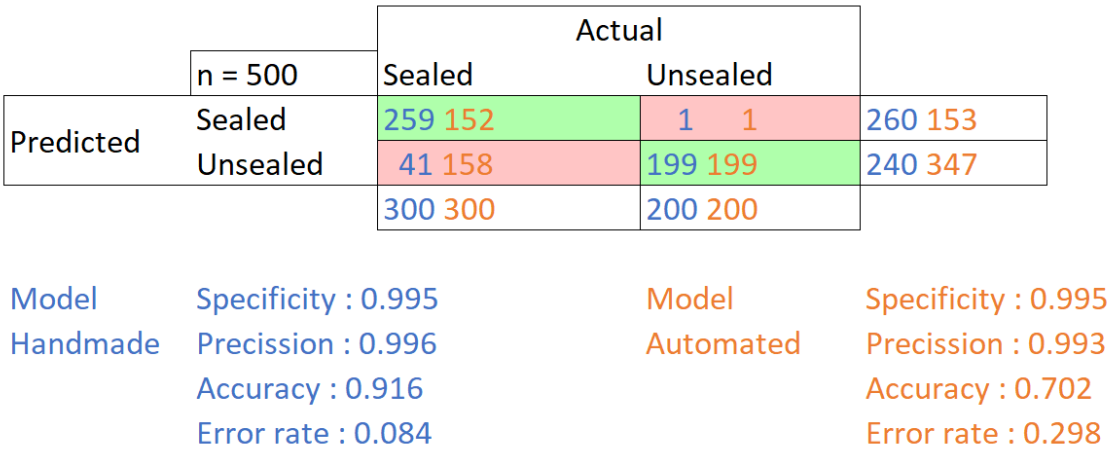

FIGURE 15: Confusion matrix of the two final models.

The confusion matrix shows that the overall accuracy of the models is 21.4% higher for the model with manually enhanced data. The second model performed better than the first model in the classes of sports facilities, driveways and industrial sites/ dumping. The classes

forest, pasture, arable land and water performed the same between both model. Streets, buildings and railways saw a minor performance gain through the manually enhanced labels. All sealed point groups saw an increase in accuracy, while unsealed groups stayed the same.

Differences of the models predictions as well as a visualization of manually enhanced models prediction can be seen in Figure 16.

5 Discussion

Figure 16 shows a selection of the 500 sample point tiles can be seen. The image tile, a NIR visualization of the automated model's prediction, then the handcrafted model's prediction, and finally a confidence visualization of the handcrafted model's prediction are placed side by side.

The first row of the street's dataset is a tile from the city centre of Hallein. The differences in the prediction of sealed areas between the two models are apparent. The handmade model predicts more pixels as sealed. The automated model is still accurate in predicting buildings, indicating that the automatically generated label data is useful for detecting houses. Roads are predicted less accurately, missing sections entirely and generally predicting too little in this case. Sealed areas between roads and buildings are misclassified.

The second row of the streets group shows a scene with houses, roads, an unpaved parking lot and trees with large canopy's. Both models seem to have problems to accurately classify the sealed areas in the scene. It seems that both models have trouble classifying pixels that are obscured by shadows. This could be due to the fact that shaded areas are mostly classified as unsealed. This is because label information is only available for a building's outline. The shadow thrown by the house will most of the time be classified as unsealed, except for the cases where it falls on a classified street or partially on another house.

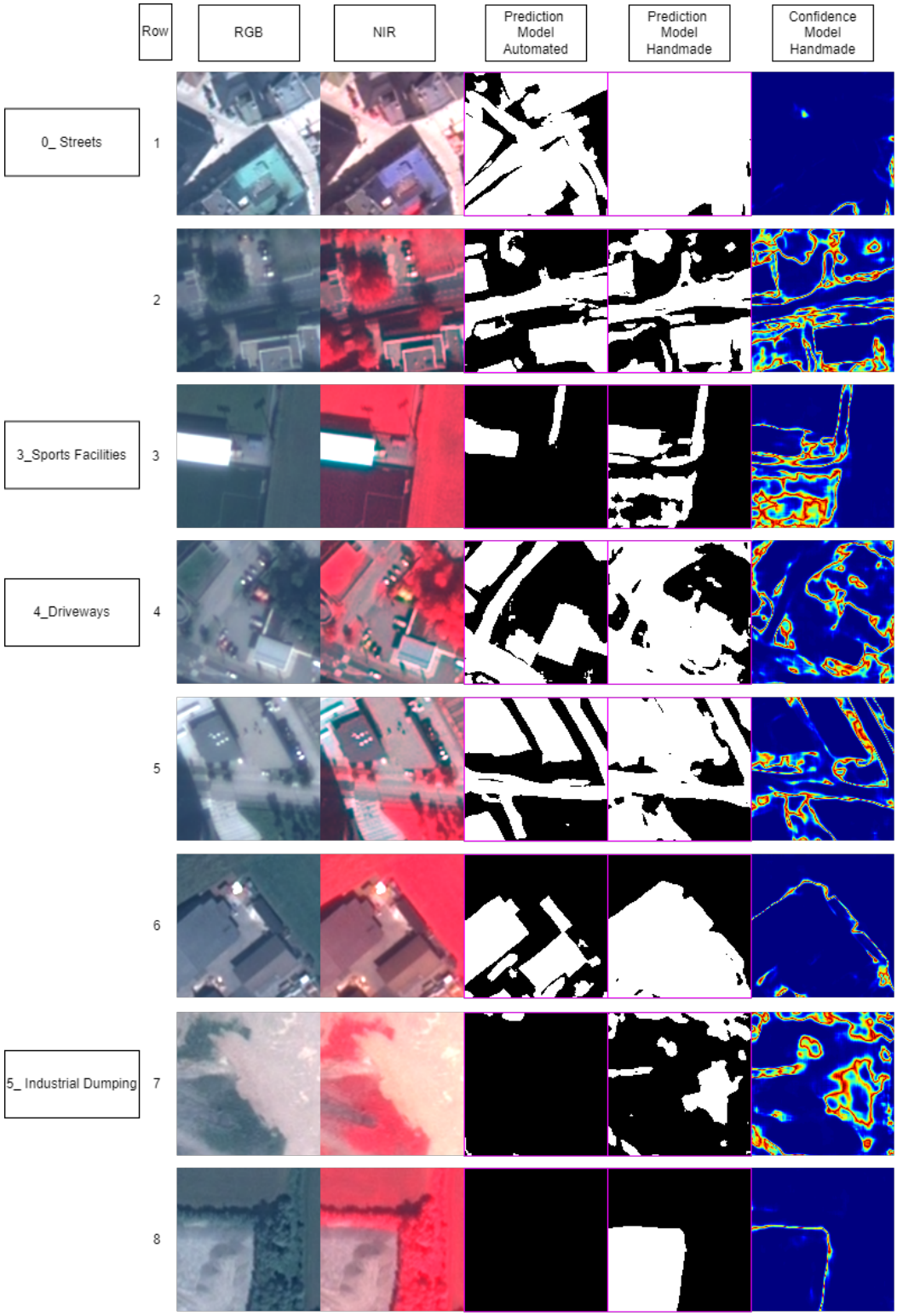

FIGURE 16: Graphic showing the image, NIR image, automated model prediction, handmade model, prediction and handmade model confidence.

Thus, shaded sealed areas are likely to be underrepresented in the dataset and harder

for the model to classify correctly. In addition, the lower spectral range of shaded areas may make it more difficult for the model to discriminate between classes. These areas are an interesting complexity which future work could focus on in order to improve prediction accuracies.

Looking at the image row from the sports facilities class, both models seem to accurately classify the roofed viewing gallery of the facility. The paved or asphalted area next to it is better classified in the hand made model than in the automated one, but it is still not covering all sealed area. The football pitch in the south has artificial grass and should be classified as sealed. The automated model has no chance of classifying it correctly, since all its label data taught it otherwise. In the handmade dataset, some artificial pitches were classified as sealed. The model still struggles to correctly classify the pitch. This suggest that there is a lack of similar input data and or to many artificial pitches which were forgotten by the manual classification.

The three images from the driveways class all paint a similar picture. While the automated model gets the main features that it learned from its label data, like the building footprints and roads, it struggles to fill the spaces between. The handmade model is much better at classifying the asphalted driveways or parking lots. Looking at the first image in the driveways row, it is to note that both models correctly classify the building with rooftop greenery as a building (Top left corner of the image). This suggests that the normalized digital elevation model helps in deciding if these areas are in fact sealed or not.

In order to further analysis, the weaknesses of the better model, a visualization of its confidence in predictions was plotted. Looking at the confidence map of the first row, it's interesting to see that the model is less confident on a tree feature in the top left section of the image. But since the prediction is still higher than the threshold of 0.5, a sealed area is predicted. The second row's confidence tile shows a lot of uncertainty. The mixed scene with trees parking lots and shadow were difficult to classify. Most uncertainty lies in parts which are shaded and bordering regions between sealed and unsealed features. Examples

of this can be seen in row 2, 4 and 5. Mixed pixels will also make an accurate classification harder. This could be mitigated with more correctly labelled data or higher resolution imagery.

FIGURE 17: Graphic showing the vectorized prediction of the handmade model in the south of Hallein.

The predicted image [see Figure 17] in the south of Hallein shows a general viability of the model. The merging of predicted tiles produces a smooth output prediction, with no edge effects. The model visually predicts sealed surfaces well. More output maps can be seen in the appendix [Figures 18,19,20,21,22,23].

The images 24 25 26 showcase different special cases and can be seen in the appendix.

Figure 24 shows a prediction on a cloud covered image. The whole image is overlaid with a thin cloud layer, which negatively impacts the prediction accuracy. It seems that the model is still able to accurately predict structures like houses. Roads on the other hand are poorly classified.

Figure 25 shows a larger building complex in its center. The building complex features two patches of roof greenery, displayed in red color in the upper image half. The model has no problems in correctly classifying the roof greenery as sealed. This shows that the use of a normalized digital surface model can help to classify these cases correctly.

Figure 26 shows an error in the normalized digital elevation model and its effect on the prediction. A row of pixels was set to a null value which evaluated to a pixel value of 65535. This abnormally high value confuses the model and in consequence the model produces artefacts during its prediction.

The differences in model architecture have only a minimal effect. The success of the model seems to be due to its training data.

The automated model works in detecting the input data that it was given, which are mainly buildings and streets. It is not sufficient to detect all sealed surfaces. The handmade model is clearly better than the automated approach when it comes to certain classes.

In order to improve the effectiveness of automated approaches, additional data sources are required. By identifying appropriate data sources, a productive automated approach can be developed. This may result in a more varied dataset that can be generated quickly. Due to the limited availability of open-source data, governments and companies that already collect and manage such data could benefit most from this approach.

This work could be built upon. There is potential in expanding on the hand labelled data in many directions. Future work could use the input data and either enhance it further with more datasets from different areas, timeframes, or satellites. Another route could be to find better openly available training data which would improve the automated approach. It could also be used as a baseline for additional tests to find better suitable model architectures.

6 Conclusion

The focus of this research was to investigate the potential of training a convolutional neural network for impervious surface detection using only open-source data and manual labelling. The comparison between the two approaches was made to determine if a model trained on automatically generated labels using only open-source data is sufficient to accurately classify impervious surfaces, or if manual labelling is needed to improve the final accuracies.

The results show that the automatic approach works well for areas with large coverage of open source soil sealing data. These areas were mainly roads, railways and buildings for the sealed classes. For the unsealed classes, both approaches performed very well. This can be explained by the large number of good training data, which were very similar between the data sets of the two models. The manually augmented dataset approach outperformed the automatic approach in the sealed classes across the board, but the most notable performance gain was in the sports facilities, industrial/dumping sites, and driveways classes. This can be explained by the fact that very little to no open source data was available for these classes.

In conclusion, the automatic approach was not sufficient to accurately classify the sealed surfaces in the test dataset. The additional manual labels helped greatly to increase the overall classification accuracy. Although the automatic approach did not perform well, its potential is still great. With the availability of more open source data, or the clever use of existing open source data sources, an automatic label generation could be developed that requires very little to no manual work to achieve good classification results.

Acknowledgments

The Pléiades imagery provided to the author, by the university of Salzburg, was only used for the soil sealing detection topic and is prohibited to be used in a commercial context or for other use cases.

List of Listings

List of Figures

References

[Abadi et al., 2016] Abadi, M., et al. (2016). TensorFlow: A system for large-scale machine learning. In *12th USENIX Symposium on Operating Systems Design and Implementa-*

tion (OSDI 16) (pp. 265–283). https://www.usenix.org/system/files/conference/osdi16/osdi16-abadi.pdf.

[Agarap, 2018] Agarap, A. F. (2018). Deep Learning using Rectified Linear Units (ReLU) https://arxiv.org/abs/1803.08375.

[Airbus, 2023] Airbus (2023). Pleiades 1A Satellite Brochure. Accessed: 2023-01-23 https://www.intelligence-airbusds.com/files/pmedia/public/r61_9_geo_011_pleiades_en_low.pdf.

[Alzu'bi & Alsmadi, 2022] Alzu'bi, A. & Alsmadi, L. (2022). Monitoring deforestation in Jordan using deep semantic segmentation with satellite imagery. *Ecological Informatics*, *70*, 101745.

[Blaschke et al., 2014] Blaschke, T., et al. (2014). Geographic object-based image analysis–towards a new paradigm. *ISPRS journal of photogrammetry and remote sensing*, *87*, 180–191.

[Bochkovskiy et al., 2020] Bochkovskiy, A., Wang, C.-Y., & Liao, H.-Y. M. (2020). YOLOv4: Optimal Speed and Accuracy of Object Detection.

[Brodrick et al., 2019] Brodrick, P. G., Davies, A. B., & Asner, G. P. (2019). Uncovering Ecological Patterns with Convolutional Neural Networks. *Trends in Ecology Evolution*, *34*(8), 734–745, https://doi.org/https://doi.org/10.1016/j.tree.2019.03.006 https://www.sciencedirect.com/science/article/pii/S0169534719300862.

[Burghardt, 2006a] Burghardt, W. (2006a). Soil Sealing and Soil Properties Related to Sealing. *Geological Society, London, Special Publications*, *266*, 117–124, https://doi.org/10.1144/GSL.SP.2006.266.01.09.

[Burghardt, 2006b] Burghardt, W. (2006b). Soil Sealing and Soil Properties Related to Sealing. *Geological Society, London, Special Publications*, *266*, 117–124, https://doi.org/10.1144/GSL.SP.2006.266.01.09.

[Burtscher, 2023] Burtscher, W. (2023). IACS https://agriculture.ec.europa.eu/common-agricultural-policy/financing-cap/assurance-and-audit/managing-payments_en#elementsofiacs.

[Cheremisinoff, 1997] Cheremisinoff, N. P. (1997). 3 - Principles of Hydrogeology. In N. P. Cheremisinoff (Ed.), *Groundwater Remediation and Treatment Technologies* (pp. 85–126). Westwood, NJ: William Andrew Publishing https://www.sciencedirect.com/science/article/pii/B9780815514114500053.

[Deng & Wu, 2012] Deng, C. & Wu, C. (2012). BCI: A biophysical composition index for remote sensing of urban environments. *Remote Sensing of Environment*, *127*, 247–259, https://doi.org/https://doi.org/10.1016/j.rse.2012.09.009 https://www.sciencedirect.com/science/article/pii/S003442571200363X.

[Detka et al., 2023] Detka, J., Coyle, H., Gomez, M., & Gilbert, G. S. (2023). A Drone-Powered Deep Learning Methodology for High Precision Remote Sensing in Californiarsquo;s Coastal Shrubs. *Drones*, *7*(7), https://doi.org/10.3390/drones7070421 https://www.mdpi.com/2504-446X/7/7/421.

[Dorn et al., 2015] Dorn, H., Törnros, T., & Zipf, A. (2015). Quality Evaluation of VGI Using Authoritative Data—A Comparison with Land Use Data in Southern Germany. *ISPRS International Journal of Geo-Information*, *4*(3), 1657–1671, https://doi.org/10.3390/ijgi4031657 https://www.mdpi.com/2220-9964/4/3/1657.

[Enzinger, 2021] Enzinger, S. (2021). Bodenverbrauch in österreich - umweltbundesamt. Accessed: 2021-07-23 https://www.umweltbundesamt.at/news210624.

[European Environment Agency, 2023] European Environment Agency, E. (2023). Urban atlas 2018 https://land.copernicus.eu/local/urban-atlas/urban-atlas-2018?tab=metadata.

[GDAL/OGR contributors, 2023] GDAL/OGR contributors (2023). *GDAL/OGR Geospatial Data Abstraction software Library*. Open Source Geospatial Foundation https://gdal.org.

[Ghorbanzadeh et al., 2022] Ghorbanzadeh, O., Shahabi, H., Crivellari, A., Homayouni, S., Blaschke, T., & Ghamisi, P. (2022). Landslide detection using deep learning and object-based image analysis. *Landslides*, *19*, https://doi.org/10.1007/s10346-021-01843-x.

[Gu et al., 2018] Gu, J., et al. (2018). Recent advances in convolutional neural networks. *Pattern Recognition*, *77*, 354–377, https://doi.org/https://doi.org/10.1016/j.patcog.2017.10.013 https://www.sciencedirect.com/science/article/pii/S0031320317304120.

[Guo et al., 2015] Guo, W., Lu, D., Wu, Y., & Zhang, J. (2015). Mapping Impervious Surface Distribution with Integration of SNNP VIIRS-DNB and MODIS NDVI Data. *Remote Sensing*, *7*(9), 12459–12477, https://doi.org/10.3390/rs70912459 https://www.mdpi.com/2072-4292/7/9/12459.

[Guo et al., 2020] Guo, X., Zhang, C., Luo, W., Yang, J., & Yang, M. (2020). Urban impervious surface extraction based on multi-features and random forest. *Ieee Access*, *8*, 226609–226623.

[Guo et al., 2018] Guo, Y., Liu, Y., Georgiou, T., & Lew, M. (2018). A review of semantic segmentation using deep neural networks. *International Journal of Multimedia Information Retrieval*, *7*, https://doi.org/10.1007/s13735-017-0141-z.

[He et al., 2015] He, K., Zhang, X., Ren, S., & Sun, J. (2015). Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification. In *2015 IEEE International Conference on Computer Vision (ICCV)* (pp. 1026–1034).

[Hoermann et al., 2018] Hoermann, S., Henzler, P., Bach, M., & Dietmayer, K. (2018). Object Detection on Dynamic Occupancy Grid Maps Using Deep Learning and Automatic Label Generation. In *2018 IEEE Intelligent Vehicles Symposium (IV)* (pp. 826–833).

[Hurd et al., 2004] Hurd, J., Assistant, R., Civco, D., & Professor, A. (2004). Temporal characterization of impervious surfaces for the State of Connecticut.

[Hurd & Civco, 2004] Hurd, J. D. & Civco, D. L. (2004). Temporal characterization of impervious surfaces for the State of Connecticut. In *ASPRS Annual Conference Proceedings, Denver, Colorado*: Citeseer.

[Iglovikov et al., 2018] Iglovikov, V., Seferbekov, S., Buslaev, A., & Shvets, A. (2018). TernausNetV2: Fully Convolutional Network for Instance Segmentation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*.

[Ioffe & Szegedy, 2015] Ioffe, S. & Szegedy, C. (2015). Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift https://arxiv.org/abs/1502.03167.

[Jacobson, 2011] Jacobson, C. R. (2011). Identification and quantification of the hydrological impacts of imperviousness in urban catchments: A review. *Journal of Environmental Management*, *92*(6), 1438–1448, https://doi.org/https://doi.org/10.1016/j.jenvman.2011.01.018 https://www.sciencedirect.com/science/article/pii/S0301479711000259.

[Kattenborn et al., 2021] Kattenborn, T., Leitloff, J., Schiefer, F., & Hinz, S. (2021). Review on Convolutional Neural Networks (CNN) in vegetation remote sensing. *ISPRS Journal of Photogrammetry and Remote Sensing*, *173*, 24–49, https://doi.org/https://doi.org/10.1016/j.isprsjprs.2020.12.010 https://www.sciencedirect.com/science/article/pii/S0924271620303488.

[Kellenberger et al., 2018] Kellenberger, B., Marcos, D., & Tuia, D. (2018). Detecting mammals in UAV images: Best practices to address a substantially imbalanced dataset

with deep learning. *Remote Sensing of Environment*, *216*, 139–153, https://doi.org/https://doi.org/10.1016/j.rse.2018.06.028 https://www.sciencedirect.com/science/article/pii/S0034425718303067.

[Kemker et al., 2018] Kemker, R., Salvaggio, C., & Kanan, C. (2018). Algorithms for semantic segmentation of multispectral remote sensing imagery using deep learning. *ISPRS journal of photogrammetry and remote sensing*, *145*, 60–77.

[Kingma & Ba, 2014] Kingma, D. P. & Ba, J. (2014). Adam: A Method for Stochastic Optimization https://arxiv.org/abs/1412.6980.

[Klokan Technologies, 2007] Klokan Technologies, G. (2007). EPSG:31258 https://epsg.io/31258-1194.

[Lang et al., 2021] Lang, L., Xu, K., Zhang, Q., & Wang, D. (2021). Fast and Accurate Object Detection in Remote Sensing Images Based on Lightweight Deep Neural Network. *Sensors*, *21*(16), https://doi.org/10.3390/s21165460 https://www.mdpi.com/1424-8220/21/16/5460.

[LeCun et al., 2015] LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep Learning. *Nature*, *521*, 436–44, https://doi.org/10.1038/nature14539.

[Li et al., 2023] Li, G., et al. (2023). Multi-Year Crop Type Mapping Using Sentinel-2 Imagery and Deep Semantic Segmentation Algorithm in the Hetao Irrigation District in China. *Remote Sensing*, *15*(4), 875.

[Li et al., 2020] Li, K., Wan, G., Cheng, G., Meng, L., & Han, J. (2020). Object detection in optical remote sensing images: A survey and a new benchmark. *ISPRS Journal of Photogrammetry and Remote Sensing*, *159*, 296–307, https://doi.org/10.1016/j.isprsjprs.2019.11.023 http://dx.doi.org/10.1016/j.isprsjprs.2019.11.023.

[Li et al., 2021] Li, L., Han, L., Ding, M., Cao, H., & Hu, H. (2021). A deep learning semantic template matching framework for remote sensing image registration. *ISPRS Journal of Photogrammetry and Remote Sensing*, *181*, 205–217, https://doi.org/https://doi.org/10.1016/j.isprsjprs.2021.09.012 https://www.sciencedirect.com/science/article/pii/S0924271621002446.

[Li et al., 2022] Li, Z., Liu, F., Yang, W., Peng, S., & Zhou, J. (2022). A Survey of Convolutional Neural Networks: Analysis, Applications, and Prospects. *IEEE Transactions on Neural Networks and Learning Systems*, *33*(12), 6999–7019, https://doi.org/10.1109/TNNLS.2021.3084827.

[Lin et al., 2017] Lin, T.-Y., Goyal, P., Girshick, R., He, K., & Dollár, P. (2017). Focal Loss for Dense Object Detection https://arxiv.org/abs/1708.02002.

[Lobo Torres et al., 2020] Lobo Torres, D., et al. (2020). Applying fully convolutional architectures for semantic segmentation of a single tree species in urban environment on high resolution UAV optical imagery. *Sensors*, *20*(2), 563.

[McGlinchy et al., 2019] McGlinchy, J., Johnson, B., Muller, B., Joseph, M., & Diaz, J. (2019). Application of UNet Fully Convolutional Neural Network to Impervious Surface Segmentation in Urban Environment from High Resolution Satellite Imagery. In *IGARSS 2019 - 2019 IEEE International Geoscience and Remote Sensing Symposium* (pp. 3915–3918).

[Merkle et al., 2018] Merkle, N., Auer, S., Mueller, R., & Reinartz, P. (2018). Exploring the potential of conditional adversarial networks for optical and SAR image matching. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, *11*(6), 1811–1820.

[Mortensen et al., 2016] Mortensen, A. K., Dyrmann, M., Karstoft, H., Jørgensen, R. N., Gislum, R., et al. (2016). Semantic segmentation of mixed crops using deep convolutional neural network. In *CIGR-AgEng conference* (pp. 26–29).

[Najafabadi et al., 2015] Najafabadi, M., Villanustre, F., Khoshgoftaar, T., Seliya, N., Wald, R., & Muharemagic, E. (2015). Deep learning applications and challenges in big data analytics. *Journal of Big Data*, *2*, 1–2, https://doi.org/10.1186/s40537-014-0007-7.

[Núñez, 2015] Núñez, J. M. (2015). Segmentation of Urban Impervious Surface Using Cellular Neural Networks.

[Onim et al., 2020] Onim, M. S. H., Ehtesham, A. R. B., Anbar, A., Islam, A. N., & Rahman, A. M. (2020). LULC classification by semantic segmentation of satellite images using FastFCN. In *2020 2nd International Conference on Advanced Information and Communication Technology (ICAICT)* (pp. 471–475).: IEEE.

[O'Riordan et al., 2021] O'Riordan, R., Davies, J., Stevens, C., & Quinton, J. N. (2021). The effects of sealing on urban soil carbon and nutrients. *SOIL*, *7*(2), 661–675, https://doi.org/10.5194/soil-7-661-2021 https://soil.copernicus.org/articles/7/661/2021/.

[PapersWithCode, 2022] PapersWithCode (2022). Papers with code - concatenated skip connection explained https://paperswithcode.com/method/concatenated-skip-connection.

[Parekh et al., 2021] Parekh, J. R., Poortinga, A., Bhandari, B., Mayer, T., Saah, D., & Chishtie, F. (2021). Automatic Detection of Impervious Surfaces from Remotely Sensed Data Using Deep Learning. *Remote Sensing*, *13*(16), https://doi.org/10.3390/rs13163166 https://www.mdpi.com/2072-4292/13/16/3166.

[Pettorelli, 2013] Pettorelli, N. (2013). *The normalized difference vegetation index*. Oxford University Press.

[Potapov, 2022] Potapov, P. (2022) https://glad.umd.edu/dataset/GLCLUC2020.

[Radhika Kamath & Maheshwari, 2022] Radhika Kamath, Mamatha Balachandra, A. V. & Maheshwari, U. (2022). Classification of paddy crop and weeds using semantic segmentation. *Cogent Engineering*, *9*(1), 2018791, https://doi.org/10.1080/23311916.2021.2018791 https://doi.org/10.1080/23311916.2021.2018791.

[Ronneberger et al., 2015] Ronneberger, O., Fischer, P., & Brox, T. (2015). U-Net: Convolutional Networks for Biomedical Image Segmentation https://arxiv.org/abs/1505.04597.

[Shao & Cai, 2018] Shao, Z. & Cai, J. (2018). Remote Sensing Image Fusion With Deep Convolutional Neural Network. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, *11*(5), 1656–1669, https://doi.org/10.1109/JSTARS.2018.2805923.

[Shao & Liu, 2014] Shao, Z. & Liu, C. (2014). The Integrated Use of DMSP-OLS Nighttime Light and MODIS Data for Monitoring Large-Scale Impervious Surface Dynamics: A Case Study in the Yangtze River Delta. *Remote Sensing*, *6*(10), 9359–9378, https://doi.org/10.3390/rs6109359 https://www.mdpi.com/2072-4292/6/10/9359.

[Shi et al., 2017] Shi, L., et al. (2017). Impervious Surface Change Mapping with an Uncertainty-Based Spatial-Temporal Consistency Model: A Case Study in Wuhan City Using Landsat Time-Series Datasets from 1987 to 2016. *Remote Sensing*, *9*, 1148, https://doi.org/10.3390/rs9111148.

[Shrestha et al., 2021] Shrestha, B., Stephen, H., & Ahmad, S. (2021). Impervious Surfaces Mapping at City Scale by Fusion of Radar and Optical Data through a Random Forest Classifier. *Remote Sensing*, *13*(15), https://doi.org/10.3390/rs13153040 https://www.mdpi.com/2072-4292/13/15/3040.

[Sulla-Menashe et al., 2019] Sulla-Menashe, D., Gray, J. M., Abercrombie, S. P., & Friedl, M. A. (2019). Hierarchical mapping of annual global land cover 2001 to

present: The MODIS Collection 6 Land Cover product. *Remote Sensing of Environment*, *222*, 183–194, https://doi.org/https://doi.org/10.1016/j.rse.2018.12.013 https://www.sciencedirect.com/science/article/pii/S0034425718305686.

[Tian et al., 2018] Tian, Y., Chen, H., Song, Q., & Zheng, K. (2018). A Novel Index for Impervious Surface Area Mapping: Development and Validation. *Remote Sensing*, *10*(10), https://doi.org/10.3390/rs10101521 https://www.mdpi.com/2072-4292/10/10/1521.

[Tzepkenlis et al., 2023] Tzepkenlis, A., Marthoglou, K., & Grammalidis, N. (2023). Efficient Deep Semantic Segmentation for Land Cover Classification Using Sentinel Imagery. *Remote Sensing*, *15*(8), https://doi.org/10.3390/rs15082027 https://www.mdpi.com/2072-4292/15/8/2027.

[Udofia, 2019] Udofia, U. (2019). Basic overview of Convolutional Neural Network (CNN) https://medium.com/dataseries/basic-overview-of-convolutional-neural-network-cnn-4fcc7dbb4f17.

[Ulku et al., 2022] Ulku, I., Akagündüz, E., & Ghamisi, P. (2022). Deep semantic segmentation of trees using multispectral images. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, *15*, 7589–7604.

[Wang et al., 2021] Wang, L., et al. (2021). Early detection of encroaching woody juniperus virginiana and its classification in multi-species forest using UAS imagery and semantic segmentation algorithms. *Remote Sensing*, *13*(10), 1975.

[Wang et al., 2022] Wang, Q., Huang, W., Xiong, Z., & Li, X. (2022). Looking Closer at the Scene: Multiscale Representation Learning for Remote Sensing Image Scene Classification. *IEEE Transactions on Neural Networks and Learning Systems*, *33*(4), 1414–1428, https://doi.org/10.1109/TNNLS.2020.3042276.

[Weng et al., 2004] Weng, Q., Lu, D., & Schubring, J. (2004). Estimation of land surface temperature–vegetation abundance relationship for urban heat island studies. *Remote Sensing of Environment*, *89*(4), 467–483, https://doi.org/https://doi.org/10.1016/j.rse.2003.11.005 https://www.sciencedirect.com/science/article/pii/S0034425703003390.

[Xiao, 2017] Xiao, Z. (2017). RSIA-LIESMARS-WHU/RSOD-dataset-: An open dataset for object detection in remote sensing images https://github.com/RSIA-LIESMARS-WHU/RSOD-Dataset-.

[Xiaowei Xu & Manickam, 2021] Xiaowei Xu, Yinrong Chen, J. Z. Y. C. P. A. & Manickam, A. (2021). A novel approach for scene classification from remote sensing images using deep learning methods. *European Journal of Remote Sensing*, *54*(sup2), 383–395, https://doi.org/10.1080/22797254.2020.1790995 https://doi.org/10.1080/22797254.2020.1790995.

[Xing et al., 2018] Xing, Y., Wang, M., Yang, S., & Jiao, L. (2018). Pan-sharpening via deep metric learning. *ISPRS Journal of Photogrammetry and Remote Sensing*, *145*, 165–183.

[Yang et al., 2018] Yang, J., Zhao, Y.-Q., & Chan, J. C.-W. (2018). Hyperspectral and multispectral image fusion via deep two-branches convolutional neural network. *Remote Sensing*, *10*(5), 800.

[Yin et al., 2022] Yin, R., He, G., Wang, G., Long, T., Li, H., Zhou, D., & Gong, C. (2022). Automatic Framework of Mapping Impervious Surface Growth With Long-Term Landsat Imagery Based on Temporal Deep Learning Model. *IEEE Geoscience and Remote Sensing Letters*, *19*, 1–5, https://doi.org/10.1109/LGRS.2021.3135869.

[Zha et al., 2003] Zha, Y., Gao, J., & Ni, S. (2003). Use of normalized difference built-up index in automatically mapping urban areas from TM imagery. *International Journal*

of Remote Sensing, *24*(3), 583–594, https://doi.org/10.1080/01431160304987 https://doi.org/10.1080/01431160304987.

[Zhang & Huang, 2018] Zhang, T. & Huang, X. (2018). Monitoring of Urban Impervious Surfaces Using Time Series of High-Resolution Remote Sensing Images in Rapidly Urbanized Areas: A Case Study of Shenzhen. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, *11*(8), 2692–2708, https://doi.org/10.1109/JSTARS.2018.2804440.

A Appendix

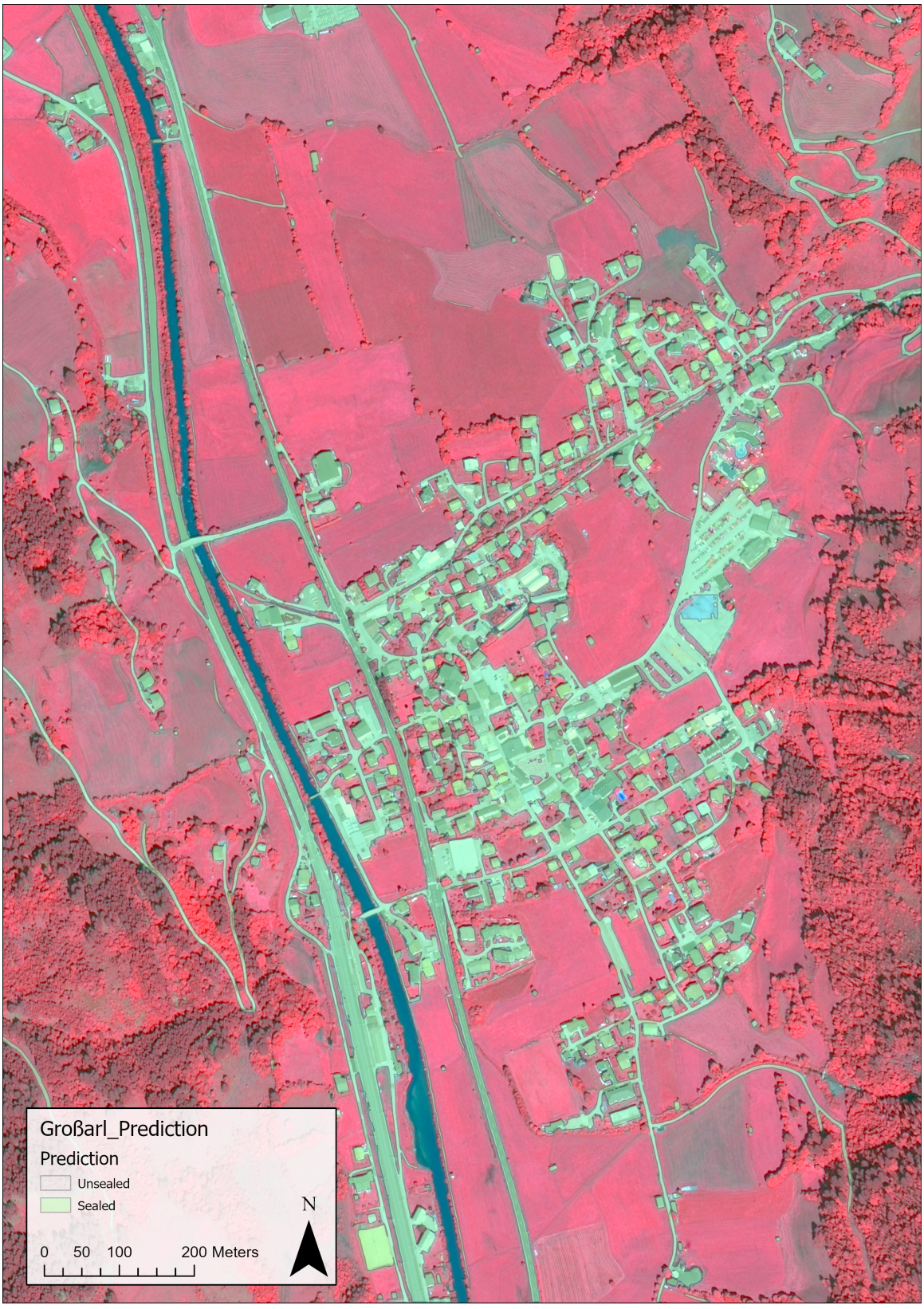

FIGURE 18: Graphic showing the vectorized prediction of the handmade model in the south of Hallein.

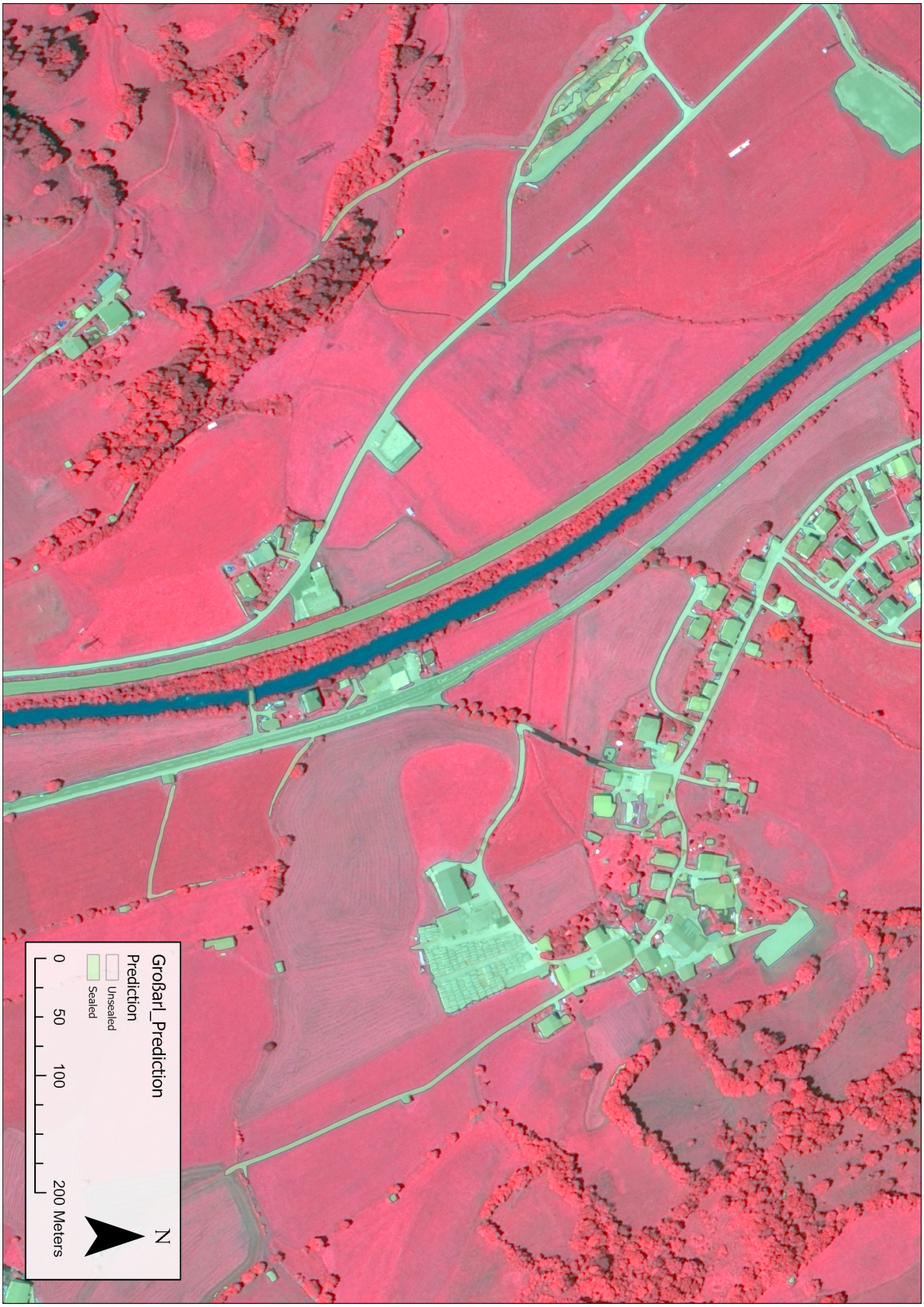

FIGURE 19: Graphic showing the vectorized prediction of the handmade model around Großarl.

FIGURE 20: Graphic showing the vectorized prediction of the handmade model around Großarl.

FIGURE 21: Graphic showing the vectorized prediction of the handmade model around Großarl.

FIGURE 22: Graphic showing the vectorized prediction of the handmade model around Großarl.

FIGURE 23: Graphic showing the vectorized prediction of the handmade model in the south of Hallein.

FIGURE 24: Graphic showing the vectorized prediction of the handmade model around Großarl. Note: The image is obscured by clouds. The clouds affect the classification heavily. Roads seem to be also always missed while the classification of houses still works well.

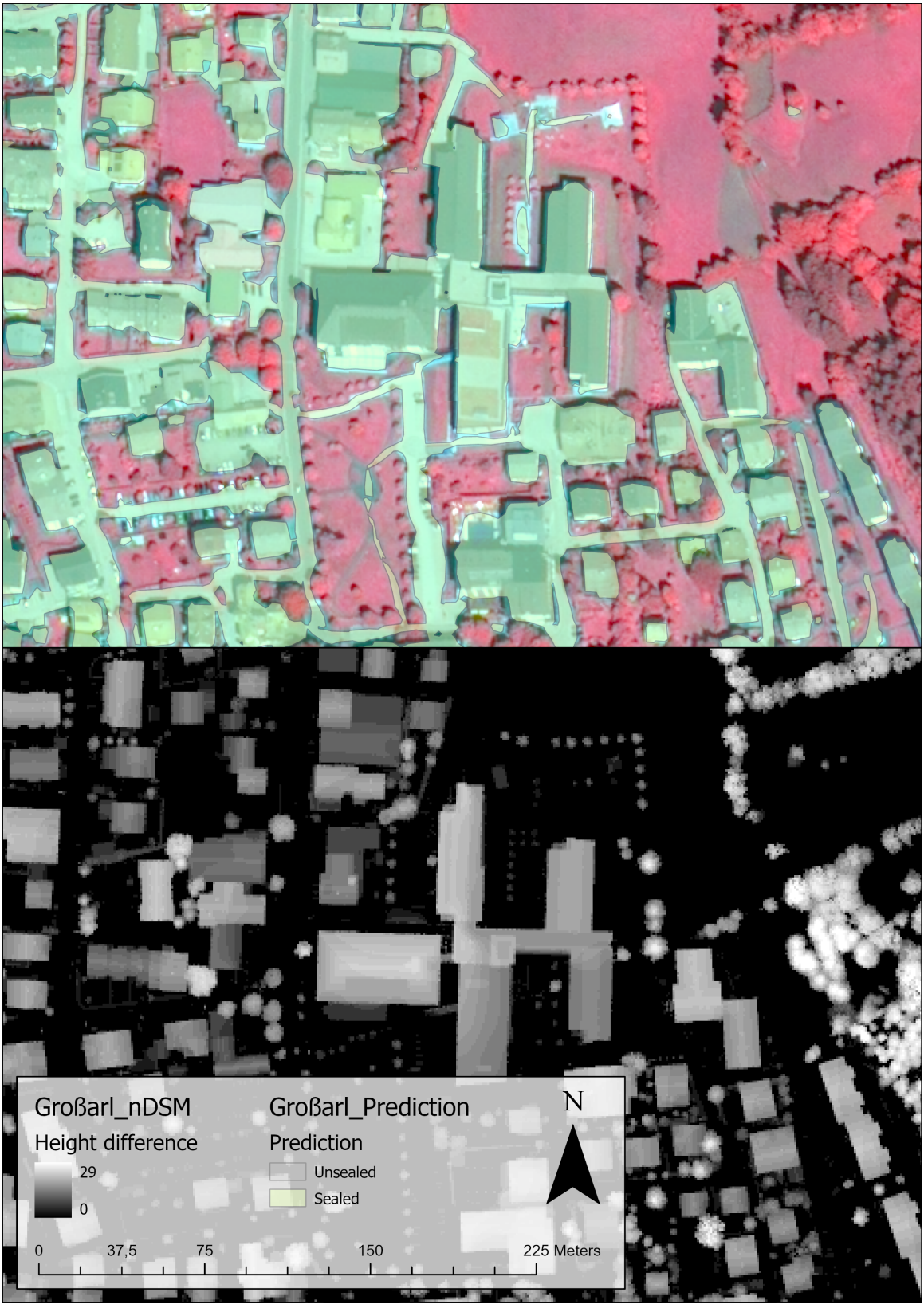

FIGURE 25: Graphic showing the vectorized prediction of the handmade model around Großarl. Note: The building with roof greenery is correctly classified as sealed.

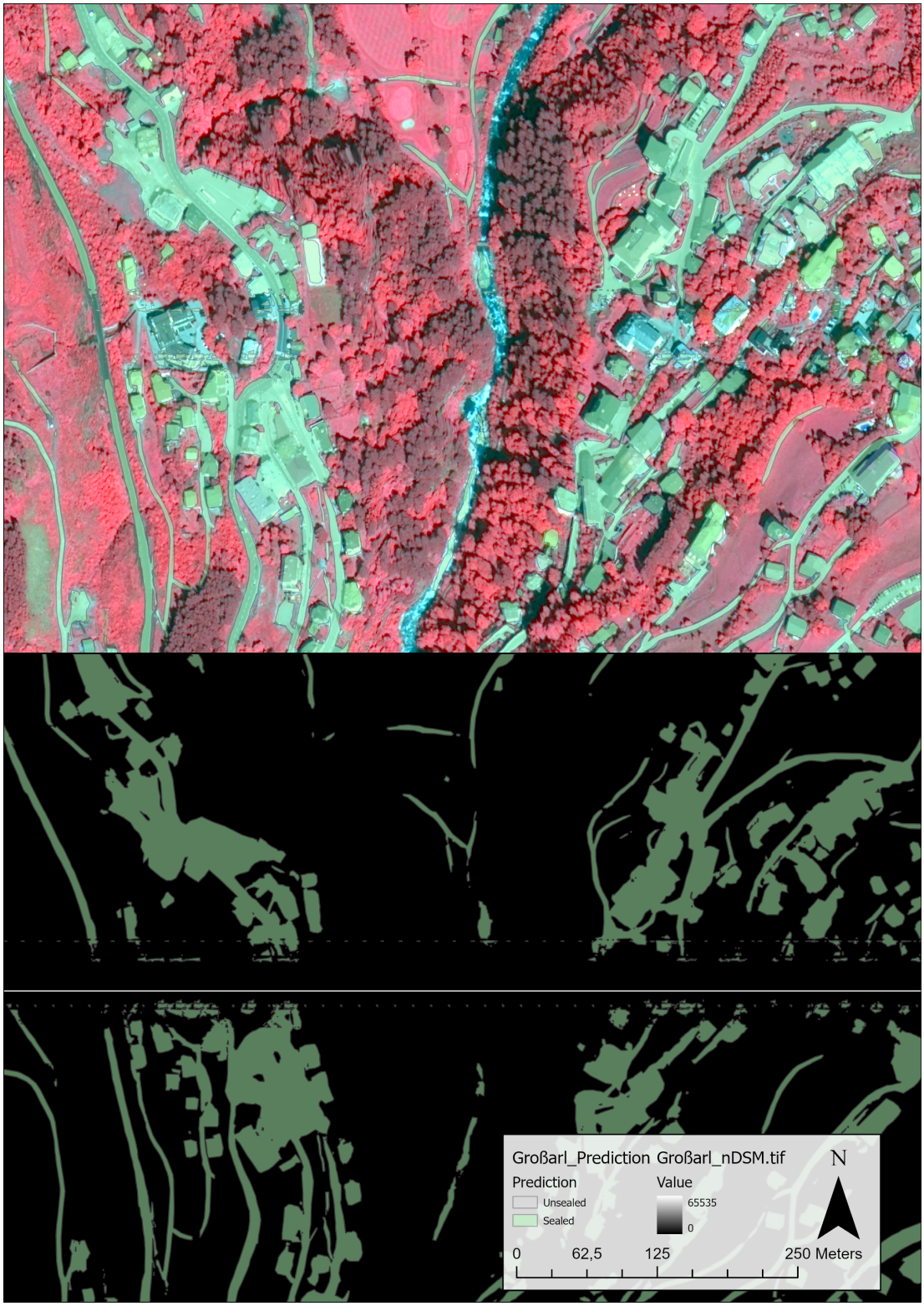

FIGURE 26: Graphic showing the vectorized prediction of the handmade model in the south of Großarl. Note: This image section had an error in the n_DSM. A pixel wide stripe of the n_Dsm layer had Null values in the form of a 65535 pixel value. The prediction cannot handle this anomaly and produces artefacts.