GIMA
Geographical Information Management and Applications

MSc. GIMA Master's Thesis

A geo-computational workflow for automatically classifying urban land cover using deep learning and street view imagery

Robin Rutten (6633072)

Supervisor: Hamed Mehdipoor
Professor: Raul Zurita-Mila
Date: 28-02-2020

Universiteit Utrecht
TU Delft
University of Twente.
Wageningen UR

ABSTRACT

Due to urbanization, there is a growing demand for knowledge on urban patterns and their dynamics at multiple spatial scales. Reliable information on urban land cover is increasingly important when dealing with this growing demand. The existing methods, such as analysis of remotely sensed imagery, do not help to accurately classify urban land cover. The aim of this research is to accurately classify urban land cover of the Netherlands. This is done by developing a novel geo-computational workflow that is able to automatically classify urban land cover on a large scale using high-resolution street view imagery and deep learning. In addition, this research looks at what the key parameters and pre-processing strategies are for training a deep learning model for classifying urban land cover. Furthermore, the performance difference between a pre-trained and newly trained deep learning model for classifying urban land cover were compared and analyzed. Finally, the effect of the image distance on the performance of urban land cover classification was investigated.

For the development of the workflow, street view images of the Dutch city Bergen op Zoom were used. In order to achieve this, a script to automatically extract street view images of the image capture locations was developed. Next, the extracted street view images were cropped and an urban land cover label was assigned. The labelled images were used to train a convolutional neural network to recognize urban land cover. Subsequently, the best performing models were used to classify urban land cover in a real-world application. The urban land cover classifications as found in the Dutch land cover map Basisregistratie Grootschalige Topografie, were used as a reference source.

The results of this research showed that the larger the images dataset, the better the performance of the deep learning model. The performance of the trained models was measured in terms of accuracy, recall, precision and F1-score. The application of the workflow resulted in an overall accuracy of 52%. After applying data augmentation techniques, this accuracy increased to 54%. In addition, comparing the predictions for different image distances resulted in the highest accuracy of 70% for images within a range of three-meters from an image capture location.

The performances achieved by the developed geo-computational workflow appeared to be not sufficient enough for urban land cover classification in a real-world application. This can be explained by the fact that the model was only trained on land cover located on a three-meter distance. In conclusion, future research should focus on improving the model that is developed in this study by increasing the number of training images for all used distances. This improvement could result in a higher performance of urban land cover classification and the workflow could be applied to the whole of the Netherlands.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACRONYMS

CNN	Convolutional neural network
NN	Neural Network
BGT	Basisregistratie Grootschalige Topografie
RELU	Rectified Linear Unit

1. INTRODUCTION

Over the last decades, most cities have been experiencing a huge population growth and a fast expansion of their urban areas (Jia et al., 2018). Due to urbanization, there is a growing demand for knowledge on urban patterns and their dynamics at multiple spatial scales (Puissant, Zhang, & Skupinski, 2012). This knowledge is used for the planning, monitoring and modelling tasks of urban planners and decision-makers (e.g. municipalities and network operators), during ongoing urbanization processes (Walde, Hese, Berger, & Schmullius, 2014). According to Hecht, Herold, Meinel, & Buchroithner (2013), knowledge about urban areas varies from information about urban structures to urban landcover. Stefanov, Ramsey, & Christensen (2001) describe urban land cover types and their spatial distribution as fundamental information required for a wide range of studies in the physical and social sciences, as well as for spatial planning. In the domain of land cover classification, urban land cover is defined as a combination of a mosaic of buildings, paved surfaces, vegetated patches, and water spaces, including, but not limited to, paved roads, buildings, bridges, vehicles, fences, railways, trees, and grass cover (Yan, Shaker, & El-Ashmawy, 2015). In cities, the distribution of urban land cover has often of a heterogeneous character, compared to non-urban landscapes. (Walde et al., 2014).

Several methods exist for capturing information on land cover, such as a field survey and analyzing remotely sensed data. The use of remotely sensed imagery is considered to be the most efficient and accurate approach (Hecht et al., 2013; Srivastava, Lobry, Tuia, & Vargas-Muñoz, 2018; Xu, Zhu, Fu, Dong, & Xiao, 2017). This imagery is mainly beneficial due to the wide-area coverage of satellites (Walde et al., 2014). In general, the classification of various land cover was conducted with moderate to high spatial resolution (e.g. Landsat ETM+) remotely sensed imagery (Shahtahmassebi et al., 2016; Yan et al., 2015). In spite of many years of experience, the quality of the resulting urban land cover maps is considered to be too low for operational applications (Beekhuizen & Clarke, 2010). Moreover, the classification of urban land cover still appears to be one of the most challenging areas to remote sensing analysis (Carleer & Wolff, 2006). This is because the conventional satellite image resolution is not well enough suited for the classification of urban land cover (Cao et al., 2018; Jia et al., 2018). This can be explained by the high spatial and spectral diversity of the surface in urban areas, which makes it hard to identify the difference (Carleer & Wolff, 2006; Jia et al., 2018). Furthermore, according to Xu et al. (2017), remote sensing-based classification always requires ground referencing data for training and validation. The collection of this reference data is considered to be a labour-intensive and time-consuming process (Xu et al., 2017).

To address the problems associated with remotely sensed imagery, research has been conducted on land cover classification based on ground-based imagery and deep learning. Deep learning is a subfield within artificial intelligence, which is the science and technology of making intelligent machines and especially intelligent computer programs (Thakur, 2019). As an example of such research, Tracewski, Bastin, & Fonte (2017) successfully repurposed a pre-trained model to filter and classify volunteered photographs for land cover and land use characterization. A pre-trained model that has already learned to extract information from images and can be used as a starting point to learn a new task. An important finding of their study was that ground-based acquired photographs, interpreted by a neural network, have a significant added value by identifying features that can never be distinguished with the help of satellite imagery. Furthermore, in a recent study by Srivastava et al. (2018), land use was characterized based on Google Street View imagery and OpenStreetMap. Both used data sources that have a large amount of information available and their approaches are scalable to cities around the globe.

The previously discussed researches are examples of integration between geosciences and computer sciences. Integration of computer science and geosciences has led to geo-computational approaches, which help to process and integrate large amounts of geographical information to solve a variety of problems (Liu, Xue, Palmer-Brown, Chen, & He, 2015; Mehdi Poor, 2019). A geo-computational workflow can be described as the process of developing a relevant tool with the use of different types of geographical data within the general context of a computational scientific approach (Xue, Hoffman, & Liu, 2009). The use of a geo-computational workflow helps scientists by improving the reliability and reproducibility of evidence from data (Atkinson et al., 2017).

After examining the currently available literature, there is no extensive research on developing a geo-computational workflow for automatic classification of urban land cover with the use of deep learning at a large spatial scale. Most studies have only focused on large-scale land cover classes such as croplands, grasslands, urban and built-up, water bodies, etc. Little attention has been given to accurately classifying more detailed urban land cover classes such as street cover types (e.g. tiles, pavers, asphalt). Hence, this research aims to contribute to the scientific problem by proposing a novel geo-computational workflow that is able to classify urban land cover types, and optimally differentiate between street cover types, by using high-resolution street view imagery and deep learning. Moreover, the proposed workflow contributes to society as well. For example, the Netherlands consists of a considerable large amount of urban areas. There are lots of institutions and companies that are greatly interested in such urban areas, for instance, spatial planners, network operators, and contractors. An automated workflow for the classification of urban land cover types can be beneficial to companies due to having more information for better decision-making. For example, such companies are interested to know what kind of land cover is present at a location before digging activities are carried out.

1.1 RESEARCH OBJECTIVES AND QUESTIONS

The main objective of this research is to develop a novel geo-computational workflow to automatically classify urban land cover in the Netherlands using high-resolution street view imagery and deep learning. The main objective is operationalized by splitting it into three research sub-objectives, which are achieved by answering four research questions:

- Sub-objective 1: To explore existing street view images for analyzing and modelling urban land cover in the Netherlands.

Q1: What are the key parameters and pre-processing strategies for training a deep learning model for classifying urban land cover?

Q2: What are the performance differences between a pre-trained and newly trained deep learning model for classifying urban land cover?

- Sub-objective 2: To develop a geo-computational workflow that automatically trains and operationalizes the classification of urban land cover in the Netherlands.

Q3: What level of accuracy, precision and recall can be reached in classifying urban land cover using a geo-computational workflow?

- Sub-objective 3: To analyze the effect of different image distances on the performance of the geo-computational workflow.

Q4: What is the influence of different images distances on the performance of the geo-computational workflow?

1.2 RESEARCH SCOPE

The scope of this research is defined in the previous formulated research goals and questions. Due to the time scope of this research, a geo-computational workflow for urban land cover classification is investigated, but is limited to investigating four classes. This research focuses on classifying urban land cover classes asphalt, pavers, tiles and vegetation only. Furthermore, the model is only used to predict land cover in the study area. The classification in other areas in the Netherlands requires different input data, which takes time to collect. For this reason, no other areas are used in this research. Finally, this research compares two types of deep learning models: a pre-trained model and a model trained from scratch. There exist more pre-trained models that can be used, but this research only focuses on one pre-trained model.

2. THEORETICAL FRAMEWORK

This chapter presents a theoretical framework that assesses the relevant techniques and methods implemented for urban land cover classification based on deep learning. In particular, this chapter elaborates on the definitions of deep learning, convolutional neural network, model training, transfer learning and pre-processing of imagery. Finally, studies that are somehow similar to this current study are considered.

2.1 DEEP LEARNING

As previously described, deep learning belongs to the field of artificial intelligence. Within the field of artificial intelligence, machine learning is an approach that automatically trains and improves a system from experience without being explicitly programmed through access to data. In addition, deep learning is a subset of machine learning and uses algorithms, inspired by the neural network functioning and structure of the brain, to let programs learn through data analysis (Thakur, 2019). Lecun, Bengio, & Hinton (2015) describe deep learning as computational models that are composed of multiple processing layers to learn representations of data with multiple levels of abstraction. Deep learning appears to be useful for multiple applications. For example, deep learning has been utilized for medical image analysis (Ker, Wang, Rao, & Lim, 2017), natural language processing (Krizhevsky, Sutskever, & Hilton, 2012), big data analysis (Najafabadi et al., 2015) and human voice recognition (Lecun et al., 2015).

Deep learning models are mainly based on neural networks (NN). Schmidhuber (2015) describes a normal NN as a network that consists of many simple connected processors called neurons. Simplified, the neurons in a neural network are arranged in a layered fashion, in which the input and output layers are separated by a group of intermediate layers. (Aggarwal, 2018). Those intermediate layers are referred to as hidden layers because the calculations performed are not visible to the user. An example of a NN is shown in Figure 1. In the architecture of a NN, every neuron is connected to at least one neuron. Moreover, each connection is evaluated by a real number, called the weight coefficient, which reflects the degree of importance of the given connection in the NN (Svozil, Kvasnicka, & Pospichal, 1997).

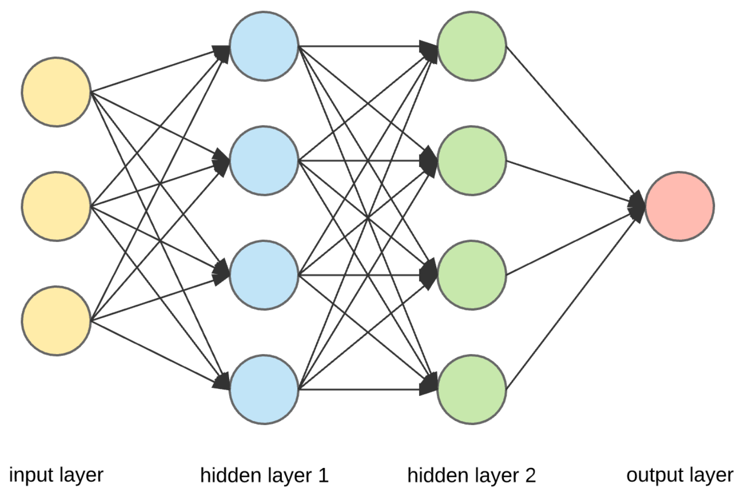

Figure 1. Schematic example of a simple neural network (Sorokina, 2018).

2.2 Deep learning in land cover classification

In recent years, more and more studies have been using deep learning for classification of land cover. Since 2014, the remote-sensing community has shifted its attention to deep learning (Ma et al., 2019). According to Kussul, Lavreniuk, Skakun, & Shelestov (2017), deep learning has proven to be an efficient and powerful method for the processing of images to extract land cover types such as buildings and roads. Ma et al. (2019) reviewed and analyzed recent publications in the field of remote sensing and deep learning. The research stated that deep learning is used in a wide variety of study areas. According to Ma et al. (2019), especially urban remote-sensing mapping has received much attention. Although a considerable amount of research has been conducted into urban land cover classification, the accuracy and efficiency of the used methods are often too small in practice. The application of deep learning methods has shown high precision performance compared with traditional classification methods. However, Ma et al. (2019) also note that the performance of deep learning in land cover classification is still inferior compared to other land cover classification methods such as scene classification and object detection. Therefore, it is expected that deep learning models develop more strongly because of the diversity of available imagery, which results in further application of land cover classification (Ma et al., 2019). This imagery consists not only of remotely sensed images, but also of ground-based imagery.

Recently, a number of studies have looked into the application of ground-based imagery and deep learning for land cover classification. Ground-based imagery can be described as imagery that is collected at street level. For example, Tracewski et al. (2017) looked into the usefulness of volunteered photos for land cover classification. In their research, a deep learning network was used based on volunteered geo-tagged imagery gained from sources such as Flickr, OpenStreetMap, and Instagram. Transfer learning was applied to train the deep learning model. Their approach was fairly successful in characterizing human influence within a scene when classifying the land cover type (Tracewski et al., 2017).

Multiple studies have been conducted combining deep learning and street view imagery. For example, Cao et al. (2018) integrated aerial and street view images for urban land use classification. They presented experimental studies showing that street view images add more value when the resolutions of the aerial images are lower. In addition, Kang, Körner, Wang, Taubenböck, & Zhu (2018) used street view imagery and CNN for the classification of the functionality of individual buildings. With their approach, relatively high accuracies were achieved for the classification of individual buildings.

2.3 Convolutional neural network

Where deep learning is a subfield of AI, is Convolution Neural Networks (CNN) a subfield of deep learning. Convolutional neural networks, also called ConvNets, are useful for solving visual tasks that rely on recognition and classification. They are designed to process data that come in the form of multiple arrays (Chellapilla, Puri, & Simard, 2006). According to Lecun et al. (2015), CNN has been used with great success in multiple applications, such as traffic sign recognition (Cireşan, Meier, Masci, & Schmidhuber, 2012), segmentation of biological images (Ning et al., 2005) and the detection of faces (Garcia & Delakis, 2002). For this reason, CNN is used for urban land cover recognition in street view imagery.

ConvNets are designed to process data that come in the form of multiple arrays, such as a 2D colour image (Lecun et al., 2015). A CNN takes in an input image, assigns learnable weights and biases (importance) to different aspects/objects in the image and a score is computed for each image class as an output (Lecun et al., 2015; Saha, 2018). ConvNets have, as previously described, an input layer, hidden layers and an output layer. In a CNN, the input layer is the image dataset that is used for training the model. Lecun et al. (2015) state that ConvNets are designed to process data of the input layer that come in the form of multiple arrays, which is in this research a colour image composed of three 2D arrays containing pixel intensities in the three colour channels (red, blue, green).

The input layer is always followed by a set of hidden layers, of which several types of layers exist. The first hidden layer that is discussed is a convolutional layer. The role of the convolutional layer in a CNN is to detect local patterns of features from the previous layer (Lecun et al., 2015). A convolutional layer is, as described by Xing & Yang (2016), a set of convolutional filters. Those filters have a specified height and a width and are smaller than the size of the input image. The filters extend through the three channels of the input layer and can perform different kinds of detections, such as edge, curve, colour and texture detection ("Different Kinds of Convolutional Filters," 2017). The output of a filter after a convolution operation is called a feature map. By stacking those maps on top of each other, more abstract and in-depth information can be received ("Different Kinds of Convolutional Filters," 2017). In general, the more convolutional layers added to the model architecture, the more details are recognized by the model. However, adding too many layers can cause the model to see details that do not exist or are not relevant.

Another hidden layer type is a pooling layer, which is similar to a convolutional layer because of the usage of filters. The task of the pooling layers is to merge semantically similar features into one (Lecun et al., 2015). This is important because of two reasons. First of all, pooling layers decrease the computational power required to process the data (Saha, 2018). This is done through dimensionality reduction. Secondly, pooling layers are useful for the extraction of dominant features in the input layer. This contributes to maintaining the process of effectively training the model (Saha, 2018). There are two types of pooling layers: max pooling and average pooling (Figure 2). Xing & Yang (2016) describe max-pooling layers as layers that summarize the activities and picks up the max values over a neighbourhood region in each feature. In comparison, the average pooling layer returns the average of all the values over a neighbourhood region.

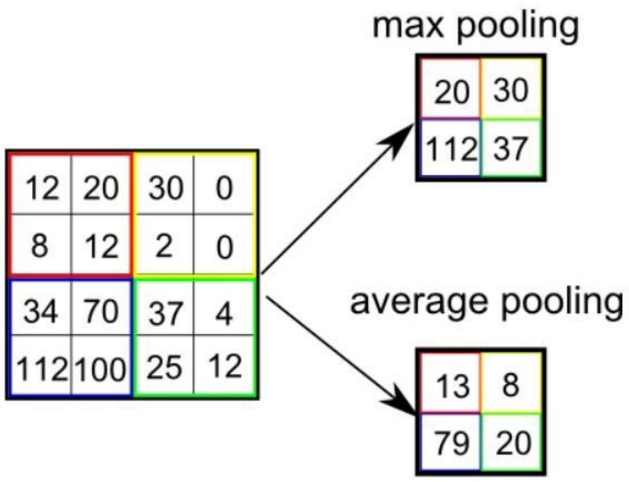

Figure 2. Visualization of the two types of pooling layers. The top right filter shows a max pooling, and the bottom right filter an average pooling (Saha, 2018).

The last hidden layer that is discussed is a fully connected layer. This layer, in contrast to a convolutional layer, connect every neuron in one layer to every neuron in the next layer. Convolutional layers do not cover the entire spatial dimension of the image which make fully connected layers mandatory to use in a CNN (Basha, Dubey, Pulabaigari, & Mukherjee, 2019). In a typical CNN, the fully connected layers comprise most of the parameters of the network.

Finally, a CNN ends with an output layer. This layer decides to which class an image belongs to. The output layer is a fully connected layer, flattens the input from the previous layers, and transforms the output into the number of classes that are defined in the model (Gupta, 2017). In this research, the SoftMax function is used as the last layer. The SoftMax function species a probability of the distribution of the used classes (Agarap, 2018). In other words, this function calculates the probabilities of each target class over all possible target classes (Polamuri, 2017). In addition, the sum of all the probabilities is equal to one. After each epoch, the output generated by the SoftMax layer is compared to the true data for error generation and a gradient of error (model loss) is created. This error rate is used to update the weights of the model and updates the filters and can be used for a new training cycle. Figure 3 presents an example of how a CNN can look like, including all three main layers.

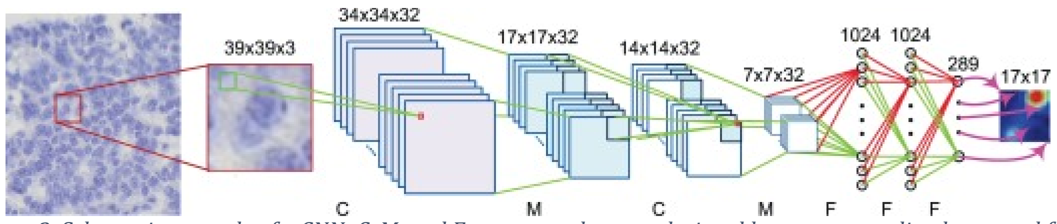

Figure 3. Schematic example of a CNN: C, M, and F represent the convolutional layer, max-pooling layer, and fully connected layer, respectively (Xing & Yang, 2016).

2.4 TRAINING A CNN-BASED MODEL

In order to let a deep learning model predict data, the model has to be trained. There are several parameters that influence the performance of a model. The main parameters of a model are the optimizer, the learning rate, the batch size and the number of epochs. This section discusses those parameters individually. First of all, the optimization parameter is one of the most important parameters of a deep learning algorithm. An optimizer starts with defining a loss function and ends with minimizing the optimization routine. The choice of optimizer can make the difference between getting good accuracy in hours or days (Parmar, 2018). The loss value indicates how well or poorly a model is behaving after every epoch. The model loss is basically a summation of the errors made for each image in training or validation datasets. In general, the lower the model loss, the better the performance of the model. The learning rate is the rate at which a function move through the search space and defines how much parameters should change each iteration (Amara, Bouaziz, & Algergawy, 2017; Parmar, 2018). In general, when the learning rate decreases, the loss decreases and the accuracy increases. It helps to prevent overfitting of the model. However, a model can be trained too accurate when is learning rate is too low. This is because, the model cannot generalize the characteristics of the training samples well enough which decreases the performance of the model (Tang, Mhamdi, McLernon, Zaidi, & Ghogho, 2016). The general rule, according to Kamilaris & Prenafeta-Boldú (2018), is to start with a high learning rate and lower it during the training process.

Furthermore, another important model parameter is the batch size setting. The batch size is the number of training samples in a single batch. At the end of every batch, a prediction is made which is compared with the expected output variables. This results in a calculated error which is used to improve the model by updating the algorithm. A dataset is divided into batches because it is often not possible to train a model on the whole dataset. Brownlee (2019) describes three types of batches: 1) A batch that consists of the whole training dataset called batch gradient descent; 2) A batch that consists of one sample is called stochastic gradient descent; 3) When the batch size is more than one sample and less than the training dataset it is called mini-batch gradient descent. Keskar, Nocedal, Tang, Mudigere, & Smelyanskiy (2019) suggest that the performance of the model is often worse when the model is trained with a large batch size compared to the small batch size. However, large batch sizes can be parallelized across many machines and reduce the training time of the model (Smith, Kindermans, Ying, & Le, 2018). The last parameter is the number of epochs. The number of epochs determines how many times an image is used for training. The number of epochs differs per model, which is difficult to predict beforehand. During the training process, it should be determined what the optimal parameter settings are for each model.

2.5 Transfer learning

Training a convolutional neural network is a difficult, computation expensive and time-consuming process. As a result, many researchers choose to make use of transfer learning. Goodfellow (2016) defines transfer learning as follow:

"Transfer learning and domain adaptation refer to the situation where what has been learned in one setting is exploited to improve generalization in another setting." (p.427)

Tracewski et al. (2017) state that the reason for this choice is because building a model from scratch is a computationally expensive process, it requires a huge set of labelled samples for training and expertise in computer vision and machine learning. When applying transfer learning, an existing deep learning model that is trained on often millions of images is used as a start and retrained with a new dataset in order to solve a problem at hand (Pan & Yang, 2010). There exist several pre-trained networks that can be used for transfer learning. VGG-16, AlexNet, ResNet, and GoogleNet are examples of pre-trained models.

There are two types of transfer learning: fine-tuning and feature extraction. To start with, the feature extraction method uses the weights of the pre-trained network, and results in a pre-specified layer and taking the outputs of that layer as the output features. This method is suitable when the image data is similar to the original training data of the pre-trained model (Xu, Zhu, Fu, Dong, & Xiao, 2017).

The second type of transfer learning is fine-tuning. Similar to feature extraction, with fine-tuning the weights of the pre-trained network are used as the starting point of the training on new data. In this case, the last layer from the pre-trained model is removed, and a new layer fitting to the number of classes is added to the model architecture. The advantage of using the fine-tuning method is that this method is less time-consuming. This is because the training starts with the weights of the pre-trained models (Xu et al., 2017).

2.6 Image pre-processing

2.6.1 Image classification

There are two main types of deep learning classification: supervised classification and unsupervised classification. Supervised classification is used when the training set consists of input features and an associated class label (Augusta, Deardon, & Taylor, 2019). It enables a model to predict on never seen data, based on labelled input training data (Sen, Hajra, & Ghosh, 2020). Supervised classification has proven to achieve state-of-the-art results on computer vision classification tasks (Augusta et al., 2019). However, this type of classifications also has limitations. First of all, the model is bound by the biases in which class it is classified. Although the model teaches itself, is still bases its decisions on the supervised classes it is trained on. Secondly, supervised classification often requires a huge manual effort in creating the labels. This manual effort could cause a smaller train dataset (Shaikh, 2018). Moreover, fully supervised methods could introduce biases during the classification.

In contrast to supervised methods, unsupervised classification does not require manual effort. With this type of learning, all the given data is unlabeled. The goal of unsupervised learning is to find underlying structures or distributions in the data (Brownlee, 2019c). Unsupervised learning can result in finding patterns that humans normally would not find. The decision to use either supervised or unsupervised learning depends on the characteristics of the dataset at hand, such as the structure and volume of data and the use case of issue (Yin Low, 2020).

2.6.2 Data augmentation

As mentioned before, deep learning modelling requires a large dataset consisting of a large number of training samples from which the model can learn. Perez & Wang (2017) state that the more data a deep learning algorithm has access to, the more effective it can be. This is emphasized by Ma et al. (2019), who state that a supervised deep learning model normally requires a great number of training samples. As stated before, acquiring labelled data with the supervised classification method is often a costly- and/or time-intensive task. In order to overcome this problem, data augmentation techniques are used.

Data augmentation is a method that increases the amount of training data. Examples of augmentation techniques are the following: rotating, mirroring, cropping, colour adjustment, brightness adjustment, and contrast adjustment of the image dataset. The application of those techniques creates an extensive training dataset, which includes not only the original data, but also the augmented data. The application of data augmentation has shown to produce promising ways to increase the performance of the model (Kamilaris & Prenafeta-Boldú, 2018). Data augmentation is an especially useful method for models that consists of a small number of input images (Sørensen et al., 2017).

3. METHOD

This section covers the various methods and techniques that were implemented to conduct the proposed geo-computational workflow in this research. The workflow developed in this research consisted of three phases: data preparation, model training, and workflow application. The main steps are discussed in more detail in the general workflow description. After the general workflow description, the method of each step is discussed separately.

3.1 GEO-COMPUTATIONAL WORKFLOW

This research proposed a geo-computational workflow that is able to automatically recognize urban land cover classes with the use of deep learning on street view imagery. Figure 4 shows a conceptual model of this workflow and its steps. The workflow started with the data preparation step. This step was used for the extraction of the street view imagery. After the collection, the street view images were cropped using cropping frames. Next, the data was labelled into four different urban land cover classes: tiles, pavers, asphalt, and vegetation. To increase the image dataset, the images were preprocessed with the use of data augmentation. The last step was to resize the images to a size of 224 by 224 pixels, which is the required input size of the VGG-16 model.

The next step was the training of two training models: a model from scratch and a pre-trained VGG-16 model. Before training the models, the images of the dataset were divided into three separate datasets: training, validation, and test. This step was repeated several times in order to train and test the models on different kinds of training input and hyper-parameters. Subsequently, the pre-trained VGG-16 model was loaded and both model configurations were set. After the training of the models, the model performances were tested on the previously created test dataset. The performance was quantified with the use of the statistical measurements: accuracy, recall, precision, and F1- score (see section 3.5.3). The model that had the best performance was used for urban land cover classification, which was the last phase of this research. In the last phase, the developed workflow was applied by classifying the urban land cover of the area of interest. The images used for this application were collected with the same method as the model training images. For the land cover classification, only the images that included the area of interest were considered and used for classification. After the classification of the images, the predictions were validated on reference data. The reference dataset that was used is the Basisregistratie Grootschalige Topografie (BGT).

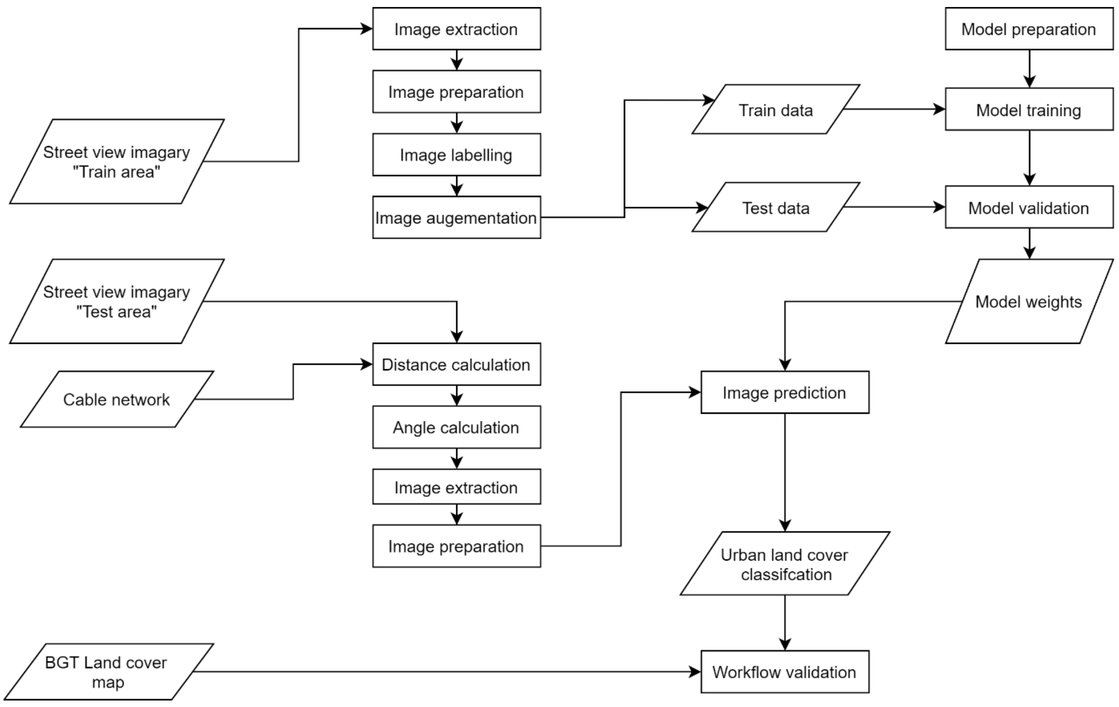

Figure 4. Conceptual model of the urban land cover classification workflow.

3.2 Study area

The area that was used for this research is the Dutch city Bergen op Zoom. This city is located in the south-west of the Netherlands in the province Noord-Brabant, close to the border with Belgium (Figure 5). The study area of this research was divided into three neighbourhoods in Bergen op Zoom: two in the Zeekant district and one in the city centre. Those neighbourhoods were chosen because the areas consist of multiple street cover types. This made those areas suitable for this research.

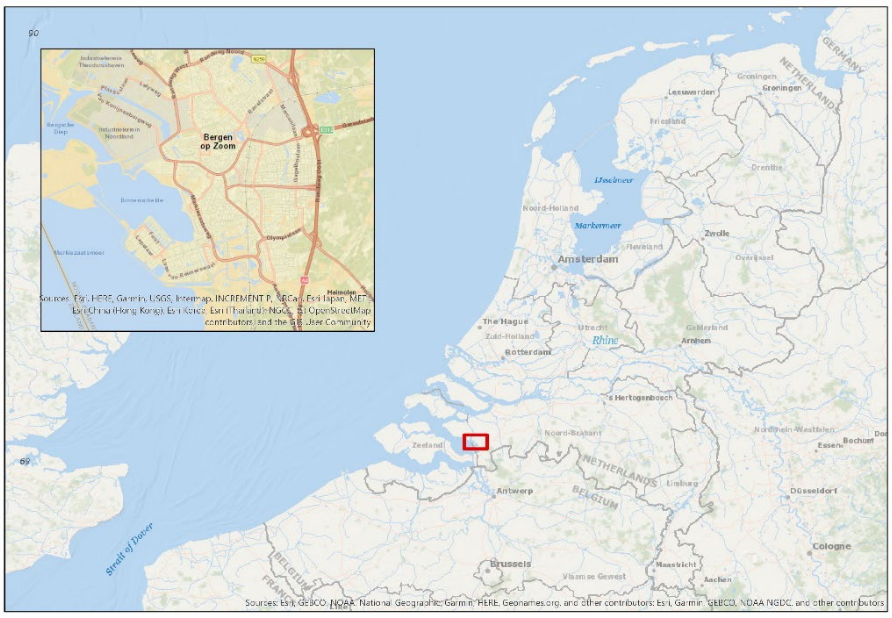

Figure 5. The geographical location of the study area Bergen op Zoom.

3.3 Research tools

In this research, several tools were used for the development of the workflow. ArcGIS Pro was used for the exploration, preprocessing, analyzing, and visualization of the spatial data. Within ArcGIS Pro, the add-in Globespotter, provided by the company Cyclomedia, was used. This add-in enabled to open, visualize, and store 360° street view imagery. Moreover, with Globespotter was it possible to display spatial data within the street view imagery. In addition, the programming tool Python was mainly used for the automation of processes, data preparation and deep learning processes. The main Python modules used were Keras, Arcpy, Pynput and Sklearn. Within Python, Keras was used for deep learning tasks. Keras is a high-level neural networks API, which is written in Python ("Keras Documentation," n.d.). Furthermore, Arcpy is a Python package of ArcGIS that enables Python to work with geographic data. Also, Pynput was used for controlling the positions and actions of the mouse of the computer. Finally, the Python library Sklearn was used for the interpretation of the deep learning model results. The used tools and software are summarized in Table 1.

Table 1. Software and tools used to implement the proposed workflow in this research.

Software	Implementation	Notes
ArcGIS Pro	Used for exploration, preprocessing, analyzing, and visualization of data	License required
Python	General programming, data preparation, deep-learning processes	Main modules used: - Arcpy (Licensed) - Keras (Open-source) - Pynput (Open-source) - Sklearn (Open-source)
Globespotter	Visualization of Cyclomedia street view imagery in ArcGIS Pro	License required

3.4 Data preparation

3.4.1 Data collection

For this research, high-resolution street view imagery was used, which was provided by the company Cyclomedia. The street view images, collected by Cyclomedia, consisted of 360° panoramas including accurate location and orientation information. The images are annually captured with an interval of five meters for every road accessible by car in the Netherlands ("CycloMedia," n.d.). The study area of this thesis included 1.100 image capture locations on which a 360° image was extracted. The study area was divided into three areas (Figure 6). Two areas were used for the training of the deep learning model and one area was used for the land cover classification. Images in those areas are captured between the 20th of March and 24th of April 2019.

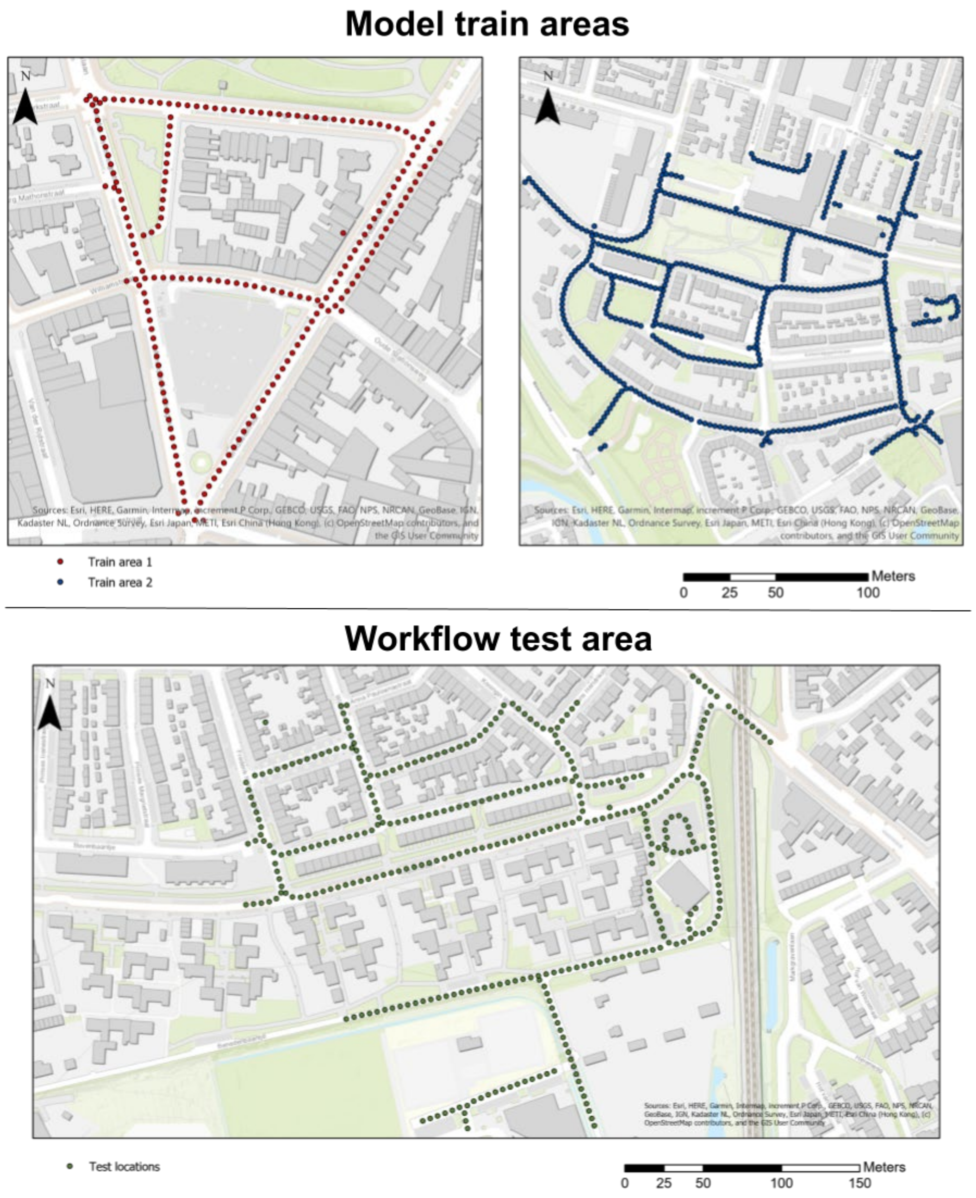

Figure 6. Street view image capture locations of the training area (top) and the testing area (bottom).

The Globespotter add-in (Table 1) lacks functionality to retrieve all available street view image locations inside the study area imagery at once. This means that the data of Cyclomedia was not directly available to use for this research. However, it was possible to open, view, and download the imagery at every capture location in ArcGIS. To efficiently extract the required image data, an automated script was created used within the ArcGIS Pro environment. In addition, the Python library Pynput that allows to control and monitor the position of the mouse of the computer was used. First, the script used the Python module Arcpy to select an image capture location. Next, the corresponding 360° street view image was displayed by clicking on the image location. Subsequently, a standard downward pitch and the zoom level of the camera was set. Next, the camera vision was set into the direction of one of the eight cardinal directions which were North, Northwest, West, Southwest, South, Southeast, East and Northeast. Finally, the image was saved in the corresponding directory. This process was repeated for every image location within the study area and for all cardinal directions with the ID of the image as the filename. This process was done for 752 capture locations in the three areas and resulted in a dataset of 6.016 images. The size of the collected street view imagery was 3.216 by 1.772 pixels.

3.4.2 Cropping frames

Before the extracted imagery was usable for training a deep learning model, pre-processing techniques had to be applied. First of all, after extracting the images, it could occur that multiple land cover classes existed within one image. In order to lower the chance of multiple classes and to be able to classify land cover for different distances, image cropping frames were created. The cropping frames cut out parts of the original images and resulted in smaller cropped images. In this research, the size of the cropping frames was one by two meters. Several steps were required to create those cropping frames. The first step was to choose a calibration location within the study area. This calibration location was used to set the camera to the pitch and zoom level the same as used for the image extraction. The next step was to create multiple squared buffers around the calibration location. With the use of the Globespotter add-in, it was possible to visualize the created buffers within the street view imagery (Figure 7). The buffers were created in a range of one to seven meters and act as a reference for the cropping frames. During the extraction of the street view imagery, the same pitch and zoom level were used. Because of this, the cropping frames were always located at the position for every extracted street view image. Figure 8 presents an example of the result of this method.

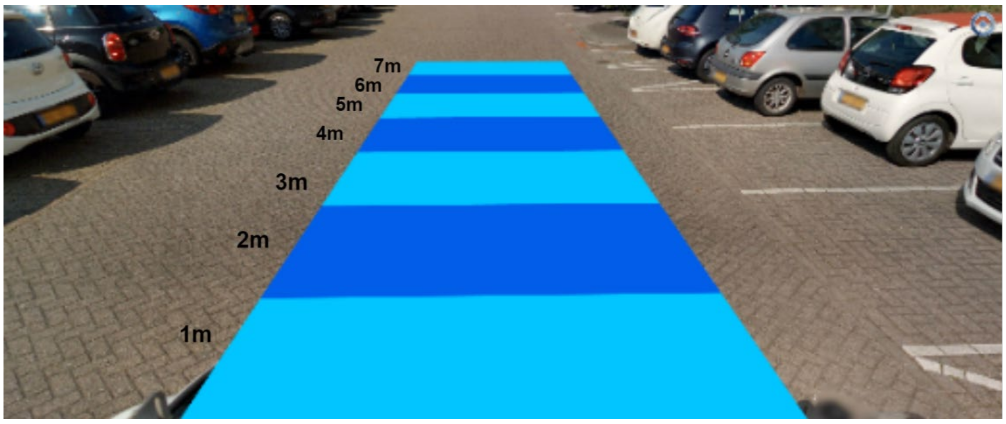

Figure 7. Example of multiple squared buffers ranging from one to seven meters.

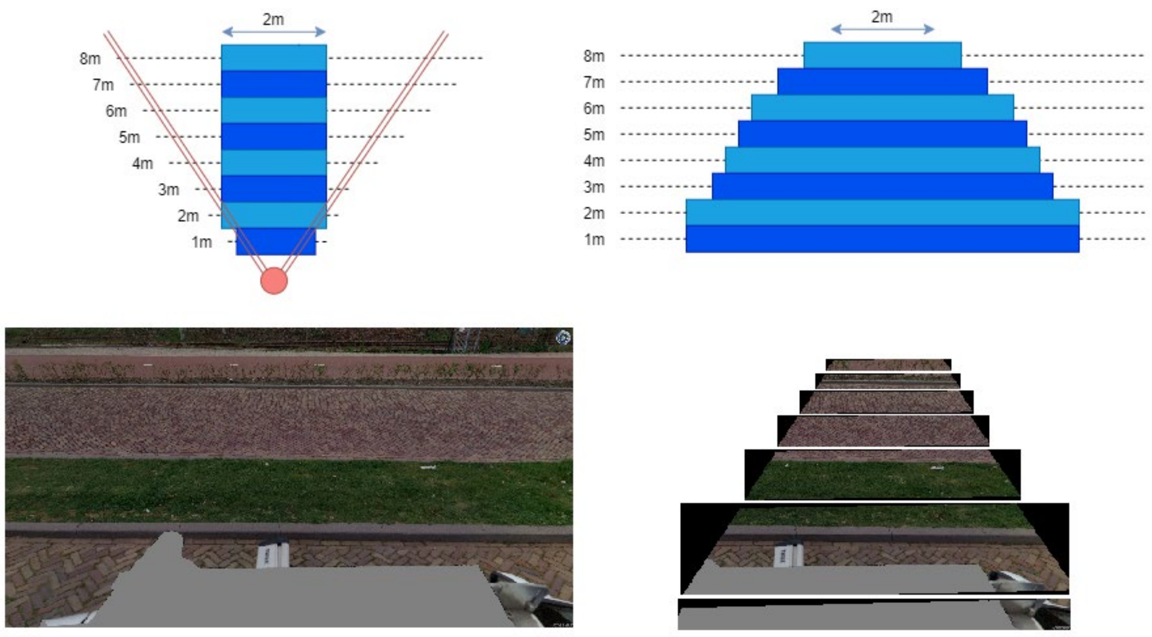

Figure 8. Visualization of the created cropping frames: schematic top-view visualization of the squared buffers (top left), schematic ground visualization of the squared buffers (top right), original street view image (bottom left), visualization of the result after applying the cropping frames (bottom right).

3.4.3 Image labelling

The previous extraction and cropping led to a street view image dataset consisting of 6.016 unique images. Cropping the images using the seven different cropping frames resulted in 42.112 unique images. Due to time limitation, it was chosen to only use the images that are cropped by the three-meter cropping frame. Only those images were used for training the model. After extracting and the preprocessing the images, the three-meter cropped images were manually labelled based on human expertise using Python. In this manual method, each individual image was displayed and a specific land cover class was given. The labels that were given consisted out of three main classes, which were open pavement, closed pavement and vegetation. Open pavement can be described as a road surface that is made of separate elements such as pavers or tiles. In the Netherlands, tiles are commonly made of concrete and used for sidewalk pavement ("Geostandaarden - Tegels," n.d.). The second class was pavers. In the Netherlands, many different types of pavers exist. In general, pavers are smaller than tiles and have an average size of 10 by 25 centimetres. Closed pavement is defined as asphalt what is made from asphalt concrete or other materials bound with bitumen ("Geostandaarden - Asphalt," n.d.). At last, the class vegetation was given for every type of greenery that exists. In this thesis, no distinction was made between different types of vegetation. Figure 9 shows examples of images of the classes asphalt, pavers, tiles, and vegetation that were used in this research.

The labelling of the images resulted in a dataset of 3.650 images. However, the created dataset appeared to be unbalanced: the class pavers was over-presented. Hensman & Masko (2015) found a relationship between the large imbalances and low performances when some classes were over-represented. To avoid the problem of an unbalanced dataset, it was chosen to only use a maximum of 500 images for each class for the training of the CNN. In addition, multiple image datasets consisting of different amount of images were created. This was done to see how the performance of the trained models responded to different image dataset sizes. Table 2 shows the number of images per class for every used dataset.

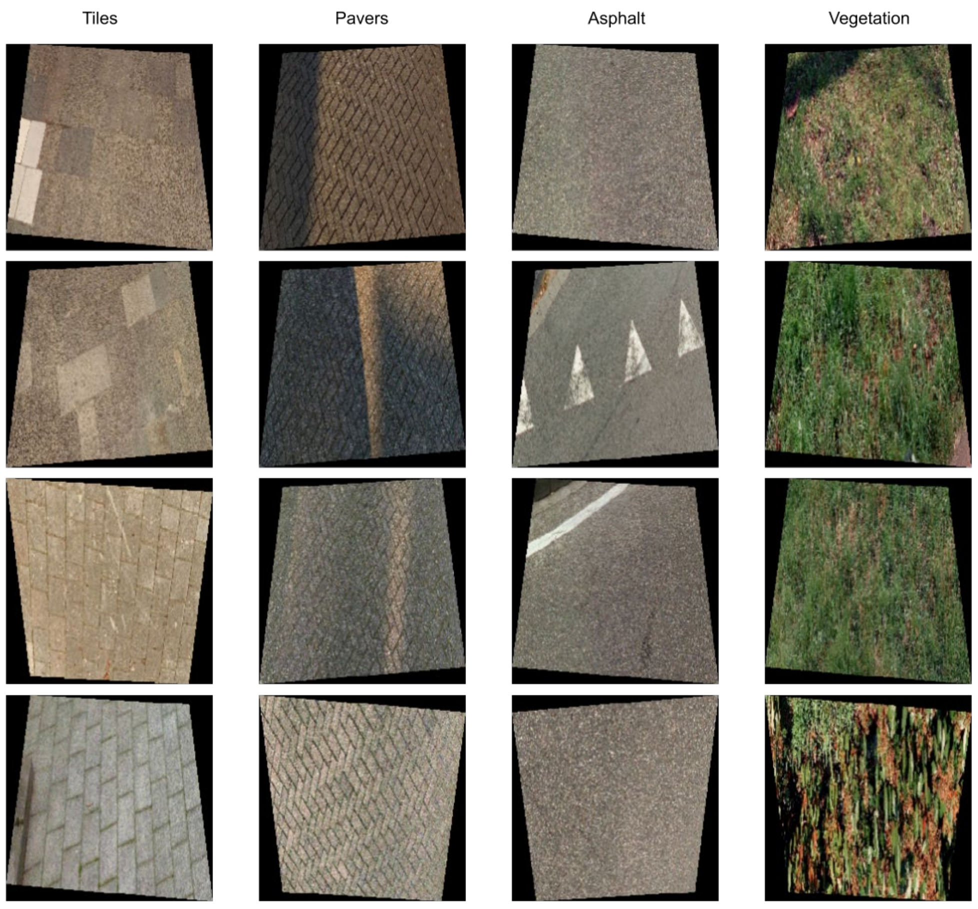

Figure 9. Examples of the labelled image dataset. It totally contains of 3.500 images of the urban land cover classes tiles, pavers, asphalt and vegetation.

Table 2. Overview of the labelled image datasets.

Dataset		Tiles	Pavers	Asphalt	Vegetation
Original dataset	(N = 3.500)	635	2025	498	492
Dataset 1	(N= 400)	100	100	100	100
Dataset 2	(N= 800)	200	200	200	200
Dataset 3	(N= 1.200)	300	300	300	300
Dataset 4	(N= 1.600)	400	400	400	400
Dataset 5	(N= 1.992)	500	500	498	492

3.4.4 Data augmentation

Due to the large size and complexity of the used CNN, the need existed to artificially expand the dataset in order to both maximize the benefits of fine-tuning and to minimize the risk of overfitting. In order to expand the images of the dataset, data augmentation techniques were applied. The goal of data augmentation is to make the model more robust.

For the initial training of the model, the previously collected images were used. Hereafter, several data augmentation methods were used to increase the image dataset in order to improve the performance of the models. Data augmentation adds value to the base dataset by providing extra information derived from the base dataset. The horizontal and vertical flipping of the image was done to increase the number of images in the initial dataset. Moreover, brightness and colour changes were applied. This helped to train the model in such a way that it could predict on images that are slightly different than the original training data, due to shadowing effects for example. Figure 10 shows examples of the result of data augmentation. After data augmentation, the training dataset was increased to 9.960 images.

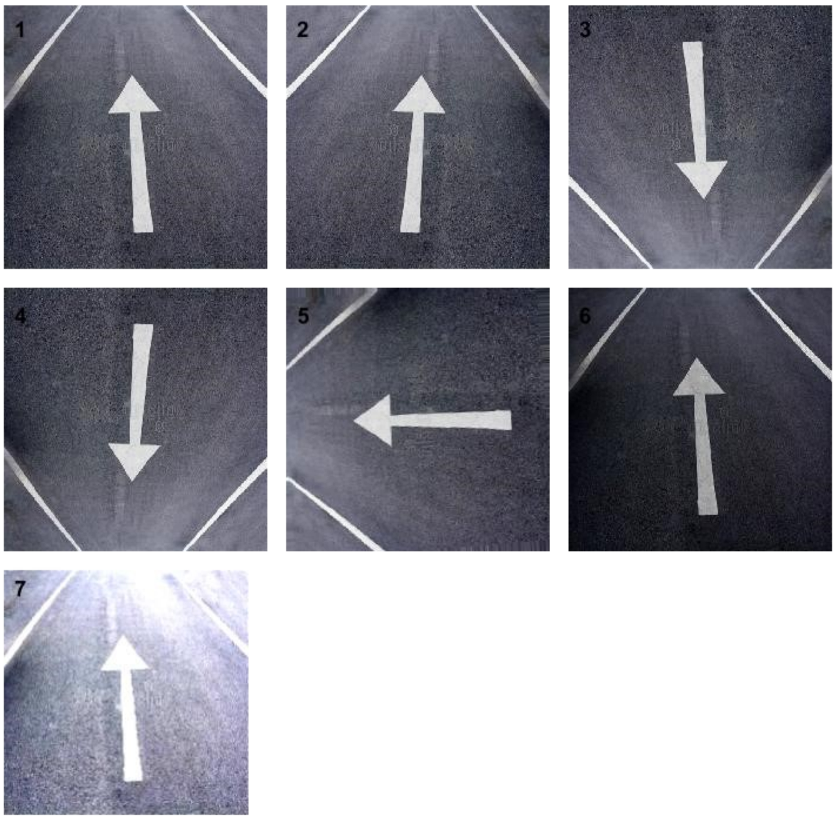

Figure 10. Examples of data augmentation: 1. Original; 2. Horizontal flip; 3. Vertical flip; 4. Horizontal and vertical flip; 5. Rotation 90; 6. Brightness decrease 50%; 7. Brightness increase by 50%.*

3.5 Model training

3.5.1 Model preparation

The previous steps have led to a large labelled image dataset. This created image dataset was used as input of the model training. A CNN requires three different datasets: a training dataset, a validation dataset, and a test dataset. A training dataset consists of images that are used to train (fit) the model. The model tries to find patterns in the training dataset and it validates itself on the validation dataset. A test dataset consists of data that the model has never seen before. The weights of the trained model are used to make a prediction on the test dataset. For this research, the image dataset was split into 90% training dataset and 10% as test dataset. The training dataset was split into 80% training data and 20% validation data.

3.5.2 Model architecture

In this research, the pre-trained VGG-16 model was fine-tuned. This pre-trained model is trained on more than a million images and is able to classify on 1.000 different scene categories of the ImageNet database (MathWorks, 2019). VGG-16 has already proved to provide satisfactory performances in multiple cases. This is substantiated by Yu et al. (2016) who compared the VGG-16 network with AlexNet. Yu et al. (2016) conclude that unlike VGG-16, AlexNet retains more unrelated background information in last convolutional layer. This retaining of unrelated background information often disturbs the final prediction. However, there is also a drawback of using the VGG-16 model. The computational power that is required for this model is higher than other networks such as AlexNet and GoogLeNet (Siegmund, Prajapati, Kirchbuchner, & Kuijper, 2018). This due to the fact that VGG-16 has a greater number of parameters which is more expensive to evaluate and requires a large amount of memory in optimizing the learning parameters.

The VGG-16 network consists of sixteen weighed layers: thirteen convolutional layers, five max-pooling layers, and three fully-connected layers (Table 3). The main contribution of Siegmund et al. (2018), is the usage of small convolutional filters (3x3) which proved remarkable improvement on the prior-art configurations. Also, the model applies the rectified linear unit (RELU) as the activation function for all convolutional layers and uses group regularization in the fully-connected layers file (Siegmund et al., 2018). Table 3 presents an overview of the architecture of the pre-trained VGG-16 network. To classify images of a more limited dataset, VGG-16 can be used as a starting point from which transfer learning can be applied. The required input size of VGG-16 is 224x224x3. In this case, the 224 stands for the number of pixels and the 3 for the number of bands (RGB) in the image. The last layer of the VGG-16 was replaced for a SoftMax layer. This output layer results in a probability number for each class and sums up to a total of 1.

Table 3. Overview of the architecture of the fine-tuned VGG-16 network.

Layer (type)	Output Shape	Param
Conv2D	224,224,32	1792
Conv2D	224,224,64	36928
MaxPooling	112, 112, 64	0
Conv2D	112, 112, 128	73856
Conv2D	112, 112, 128	147584
MaxPooling	56, 56, 256	0
Conv2D	28, 28, 512	1180160
Conv2D	28, 28, 512	2359808
Conv2D	28, 28, 512	2359808
MaxPooling	14, 14, 512	0
Conv2D	14, 14, 512	2359808
Conv2D	14, 14, 512	2359808
Conv2D	14, 14, 512	2359808
MaxPooling	7, 7, 512	0
Flatten	25088	0
Fully Connected	4096	102764544
Fully Connected	4096	16781312
SoftMax	4	4004

Although several pre-trained image classification models exist, the models are usually not trained on images of urban land cover classes. Due to the previously described disadvantages, a new CNN configuration was developed from scratch. The goal of this was to see what the performance of a non-trained model can be. This new model consisted of two convolutional layers, one max-pooling layer, and three fully-connected layers (see Table 4). In the convolutional layers, as explained before, filters of the size 3x3 slide over each image. The same image input size as the VGG-16 model was used, which is 224x224x3. A flattened layer was included because the convolutional layer is a 2D layer and the fully connected layer requires a 1D layer. The activation layer was RELU and for the last fully connected layer a SoftMax layer was used. The optimizer of this model was Adam, with a learning rate of 0.0001. Finally, to prevent over-fitting from occurring, early stopping was applied. Overfitting occurs, according to Ling (1995), when the error on the testing sample increases if training goes on for too long. Therefore, training was stopped at the most optimal point, rather than allowing to proceed until the training error was as small as possible. Also, to decrease the training time of the model, two dropout layers were included. Those layers consist of settings to zero the output of each hidden neuron with probability 0,1(Krizhevsky et al., 2012).

Table 4: Overview of the architecture of the scratch model.

Layer (type)	Output Shape	Param
Conv2D	222,222,32	896
Conv2D	220,220,64	18496
MaxPooling	110, 110, 64	0
Dropout	110, 110, 64	0
Flatten	774400	0
Fully Connected	256	198246656
Dropout	0,1	0
Fully Connected	128	32896
Dropout	0,1	0
Fully Connected	4	516

3.5.3 Model performance

To evaluate the performance of both modes, an accuracy assessment was conducted. This was done by the statistical parameters accuracy, recall, precision, and F1 score. The previously defined test dataset consisting of 10% of the original train dataset, was used for this evaluation. In order to do so, a confusion matrix was used where a prediction could be either a True Positive (TP), False Positive (FP), True Negative (TN), or False Negative (FN). Where True Positive is the number of images that are correctly classified, False Positive is the number of images that are classified as true while being false. True negative is the number of images that are correctly rejected. False Negative is number of images that are classified false while being true.

First of all, accuracy (A) is the fraction of the prediction of the model that is correct. In other words, the accuracy can be defined as the number of correct predictions divided by the total number of predictions. The overall accuracy is given in percentages and indicates 100% when all images are classified correctly. Accuracy is mathematically defined as follows:

$$(1) \quad A = \frac{TP + TN}{TP + TN + FP + FN}.$$

Accuracy is not the only performance measurement method that needed to be considered. This is due to the fact that the performance differs per class. Three more methods were used to measure the performance of the model which are recall, precision and F1- score. Those measures are computed from the elements of the confusion matrix.

Recall (R) can be described as the fraction of the total amount of true positives that are found. The precision (P) of a model can be defined as the fraction of positive instances which was correct. In other words, recall measures how many truly relevant results are predicted, and precision measures the relevancy of the result ("scikit-learn - precision and recall," n.d.). Recall and precision are mathematically defined as follows:

$$(2) \quad R = \frac{TP}{TP + FN} \qquad (3) \quad P = \frac{TP}{TP + FP}.$$

The last measure that was considered is the F1-score. The F1 is the weighted average of recall and precision (Huang, Wang, & Abudureyimu, 2012). This method results in a score between 0 and 1, where 0 is the lowest possible score and 1 the highest possible score F1-score is mathematically defined as follows:

$$(4) \quad F1 - score = \frac{R * P}{R + P} * 2.$$

3.6 Model application

In order to check whether the workflow was able to classify urban land cover, a prediction of the land cover was done with the use of the best performing deep learning models. This section covers the method that was used for the prediction and validation of the land cover classification. For the classification, the area of interest, the image capture location of the test area, and the weights of the best performing CNNs were used. The classification resulted in predicted land cover classes in the test area. After the classification, the urban land cover predictions were validated on reference data.

3.6.1 Classification

In this research, the area of interest was the urban land cover above a hypothetical cable network, which was stored as a shapefile, within the study area. Normally, the cables are located one meter below the surface. This cable network was a feature line and each element of the feature line was split on every two meters. The total length of this network within the study area is 4,3 km and is visualized in Figure 11. In addition, the test area consistst of 504 image capture locations.

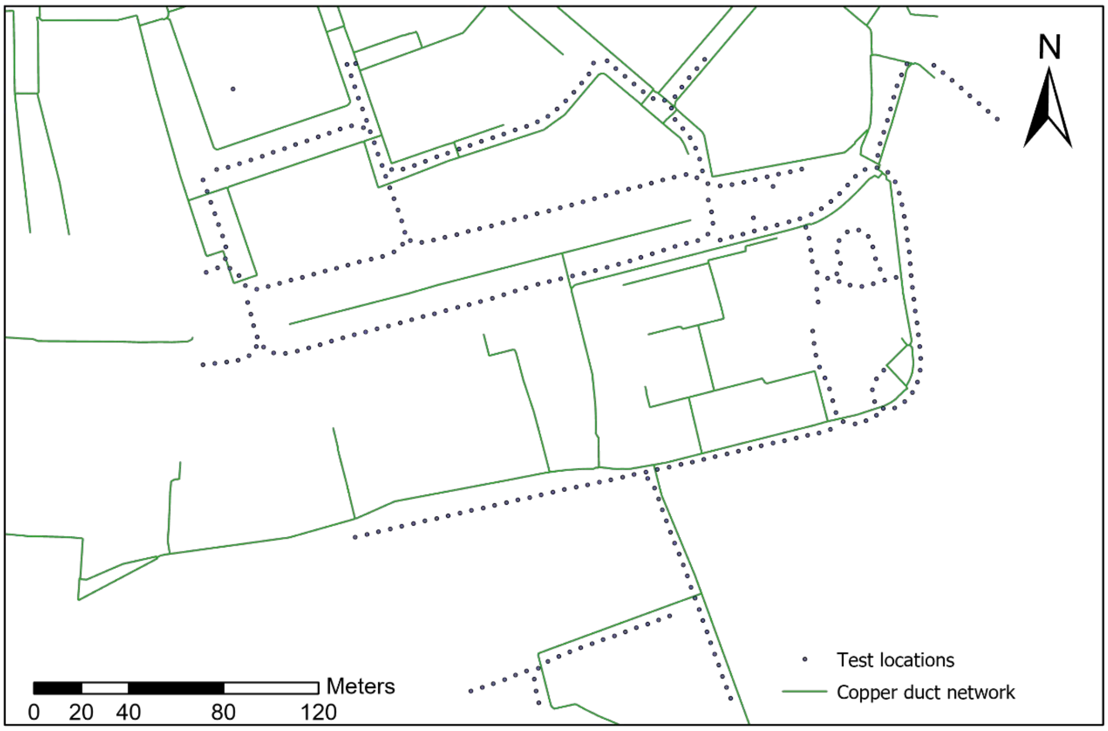

Figure 11. Visualization of the test area including the image capture locations and the used cable network.

The first step of the classification was to calculate both the distance and angle between the capture location and the cable network. By doing this, the street view image that contains the area of interest could be selected. Both distance and angle were calculated with the use of the tool 'Near' in ArcGIS. This tool calculates the distance and additional proximity information between the input features and the closest feature in another layer or feature class ("Pro.ArcGIS.com," n.d.) (Figure 12). Other variables that can be calculated with this tool are the location and the angle of the nearest feature. The location is given by x and y coordinates and the angle is within the range of -180° to 180°, with 0° to the east, 90° to the north, 180° (or -180°) to the west, and -90° to the south ("Pro.ArcGIS.com," n.d.). To prevent incorrect calculations, it is essential that both input and near features are projected in the same coordinate system.

This information was used for the street view image extraction of the test area. For the training images were all eight cardinal directions used. However, for the test area, we were only interested in the image facing the area of interest: the cable network. With the information from the previously calculated angle, it was possible to select only the locations that were facing the area of interest, including the corresponding direction. This method significantly reduced image extraction time because not all eight cardinal directions were required. The result of this method was information about the image capture locations that were closest to the features, the distances between the capture locations and features of interest, and in which the direction the camera should be facing. Combining those three inputs resulted in a cropped image containing the nearest feature.

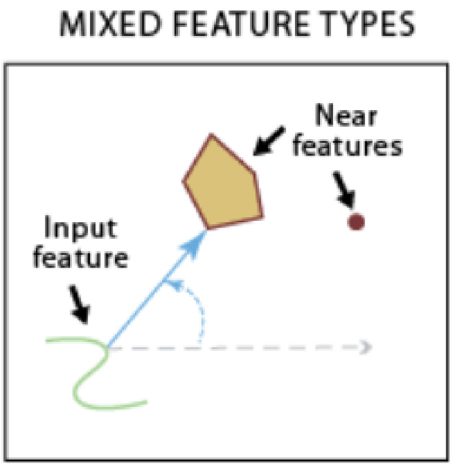

Figure 12. Visualization of the tool Near ("Pro.ArcGIS.com," n.d.).

3.6.2 Validation

The next and final step was to validate the previous predicted urban land cover classes. This was done by comparing the results of the prediction with reference data. This validation was done with the use of an accurate, already existing land cover map. The existing land cover map used in this research was the topological map of the Dutch national 'basic registry' called: Basisregistratie Grootschalige Topografie (BGT). Data providers, such as municipalities and organizations, are responsible for maintaining the topographical information in this registry ("Digitaleoverheid," n.d.). The BGT contains object definitions of roads, water, land use/ land cover, bridges, and tunnels. The BGT was used as the validation source because the registry includes the same classes as used in this research. In this case, the BGT was a suitable validation source because the registry is accurate and up-to-date in the study area. However, this is not always the case for the whole of the Netherlands. By comparing the predicted urban land cover classes with the BGT data, the model could be validated on true data.

In order to be able to validate the predictions, the BGT classes were joined on the cable segments separately. First, the BGT was converted to a raster layer, with a cell size om 10 centimetres. Secondly, the tool zonal statistic was used to calculate statistic on the values of the BGT raster within the zones of the cable network. The calculated median was used for the assignment of the BGT class of the segments of the cable network.

The BGT classes of each cable segment are shown in Figure 13. This figure shows that the cable network consisted of 15 asphalt segments, 884 paver segments, 1.578 tile segments, and 200 vegetation segments.

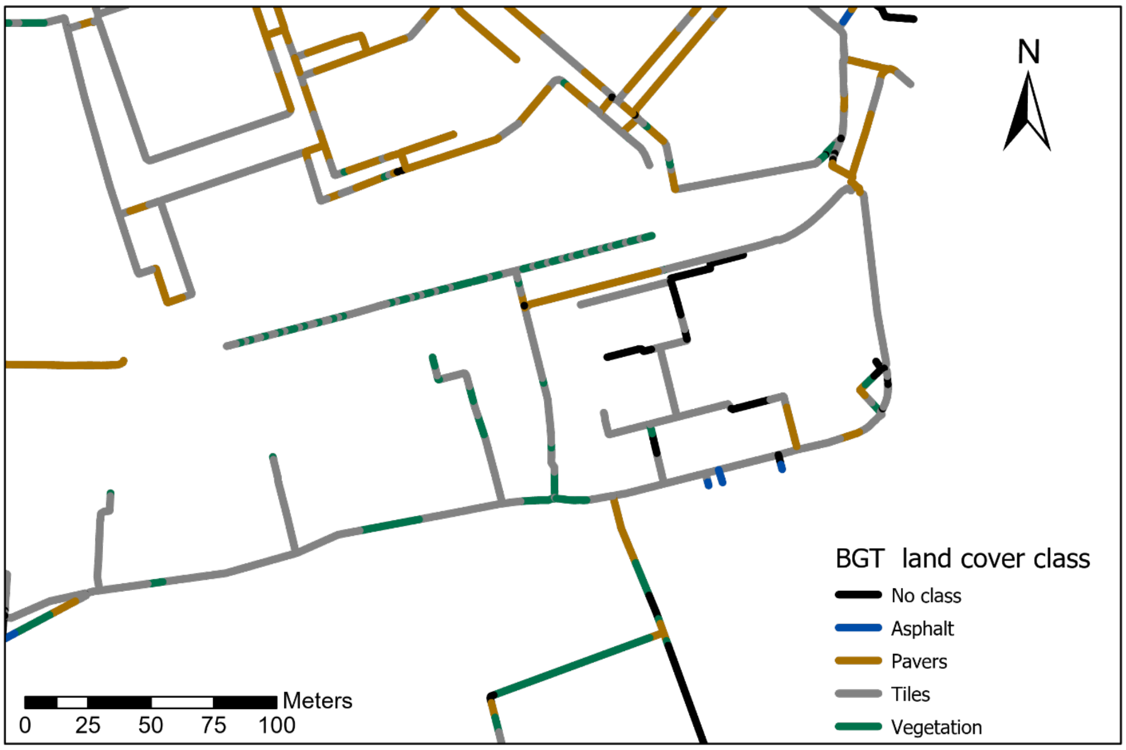

Figure 13. The urban land cover of each segment of the cable network retrieved from the BGT.

3.7 Results presentation

In order to monitor the performance of the learning models, a learning curve was used. According to Anzanello & Fogliatto (2011), learning curves are considered to be effective tools for monitoring the performance of models learning a new task. In addition, a learning model behaviour can be diagnosed using the shape and the dynamics of a learning curve (Brownlee, 2019b). A learning curve is a line-pot with the number of epochs (time) on the x-axis and the learning or improvement on the y-axis. The plot consists of the learning curve of both the training and validation of the model. The train learning curve indicates how well the model is learning on the training dataset. Likewise, the validation learning curve indicates how well the model is generalizing on the validation dataset (Brownlee, 2019b). In addition, the train and validations curves are plotted for both accuracy and loss. The model loss is a number that indicates how the model's prediction was over a single epoch. If the model's predictions are perfect, the loss is equal to zero. The model loss is plotted for both the train and the validation loss.

For the presentation of the results, several methods were used. First of all, a confusion matrix was used to report the number of predictions of the urban land cover classification model (Figure 14). This confusion matrix was based on the amount of True and False Positive and negative predictions. The vertical axis represents the true images labels and the horizontal axis represent the labels of the predicted images. The confusion matrix was used for both testing the performances of the trained models and for the validation of the land cover classification. From the constructed confusion matrices, the validation measures accuracy, precision, recall and F1-scores were calculated and presented in a table.

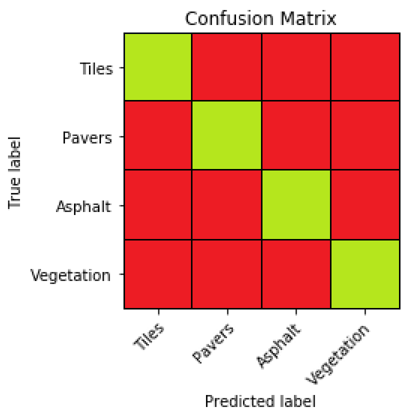

Figure 14. Example of a confusion matrix. The green tiles are either True Positives or True Negatives. The red tiles are either False Positives or False Negatives.

In addition, the results of the urban land cover classification of the test area are visualized with the use of maps. Those maps consist of the cable network segments including the predicted urban land cover. Also, along with the maps, a graph is provided that shows the count of cable segments per classified class. For visualization purposes, only a part of the cable network is presented in the maps. The complete maps are presented in Appendix C, D and E.

4. RESULTS

In this chapter, the results of this research are presented. First, the training process of the models is discussed. Secondly, the performances of the trained models are shown. In addition, accuracy, recall, precision and F1-score are the measures that are discussed. Furthermore, the results of the urban land cover classification are shown. Finally, the performances differences between the different image distances are compared.

4.1 MODEL TRAINING

4.1.1 Scratch model

For visualization purposes, only the learning curves of the model trained on the 500-dataset are shown in Figure 15. The learning curves of the other scratch models are shown in Appendix A.1, where both training and validation accuracy and loss are plotted. The learning curves show that the train accuracies of the models slowly increased to a maximum of 96%. This means that the model was able to generalize of 96% of the training dataset. Furthermore, the validation accuracy of all models fluctuated a lot between 45% and 55 %. Remarkable is that the validation accuracy of dataset-1 appeared to reach the highest of all, namely 67%. The difference between the training and validation accuracy suggests that parts of the learned patterns specific to the training data were not relevant to the validation data.

The accuracy of the train model increased when the dataset size increased. The opposite applied to the train loss. The train loss of the smallest dataset size started at an error of 345, and the train loss of the largest dataset size started at 55. Over the epochs, the train loss of all models decreased slowly until a minimum of 0.3. The validation loss of most models decreased to a value of 0.5, but fluctuated over the epochs between 1.5 and 0.5. The fluctuation of the validation accuracy and loss suggests that over-fitting of the model was occurring. Although the model seems to fit the training data well, the model appeared not to be able to generalize on new data. Lastly, the model runtime increased when the size of the dataset increased. The exact runtime of each model is shown in Table 5.

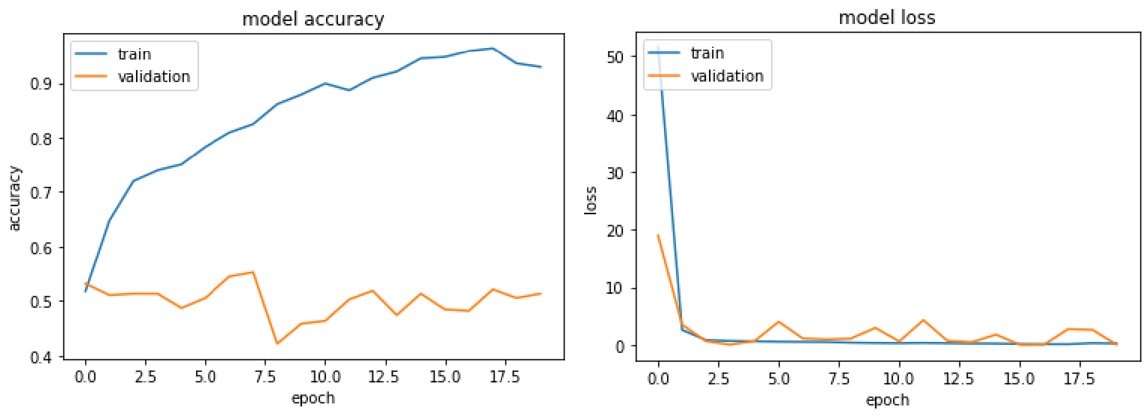

Figure 15. Learning curves model accuracy (left) and model loss (right) of the scratch model trained on the 500-dataset.

4.1.2 VGG-16 model

The second model that was trained is the pre-trained model VGG-16. The learning curves of both model accuracy and model loss are visualized in Figure 16. The models train accuracy slowly increased until a maximum of 58%. The validation accuracy followed the same pattern, but this accuracy did not rise above 47%. Remarkably, the model train loss started at 1.39 and constantly decreased to 1.33 over the epochs. The validation loss, however, fluctuated a lot between a loss of 1.35 and 1.43. The total runtime of this model was 8.500 seconds.

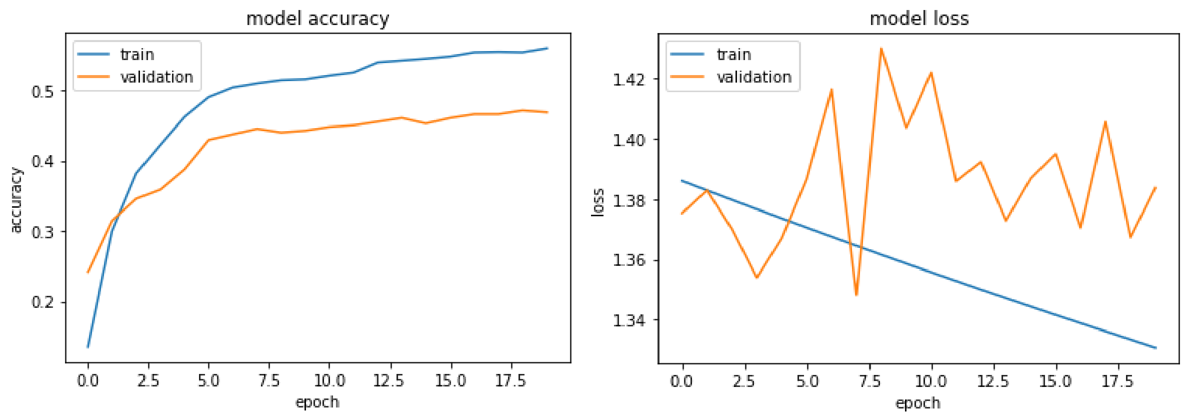

Figure 16. Learning curves model accuracy (left) and model loss (right) of the trained VGG-16 model.

4.1.3 Augmented models

In this section, the learning curves of the models trained on the augmented image dataset are presented. With the use of data augmentation, the largest labelled image dataset was increased from 1.992 to 9.960 images. This was done with the use of horizontally and vertically flipping the images and changing the brightness of the images. Unlike the non-augmented models, the models were trained for 10 epochs. This was done due to the long-runtime of the augmented models. The learning curves of the augmented models of both the model from scratch and the pre-trained VGG-16 are presented in Figure 17 and Figure 18.

First of all, the scratch model train accuracy started at a percentage of fifty-six. This accuracy slowly increased to a maximum of 95% at the end of the tenth epoch. The validation accuracy started at 65%. Hereafter, this accuracy increased to 71% over the next two epochs. After this, the validation accuracy slowly decreased. At the last epoch, the validation accuracy increased to a final 70%. Furthermore, the model train loss started at an error rate of 30. After the first epoch, the loss rapidly decreased to 1 and remained at a constant level over the following epochs.

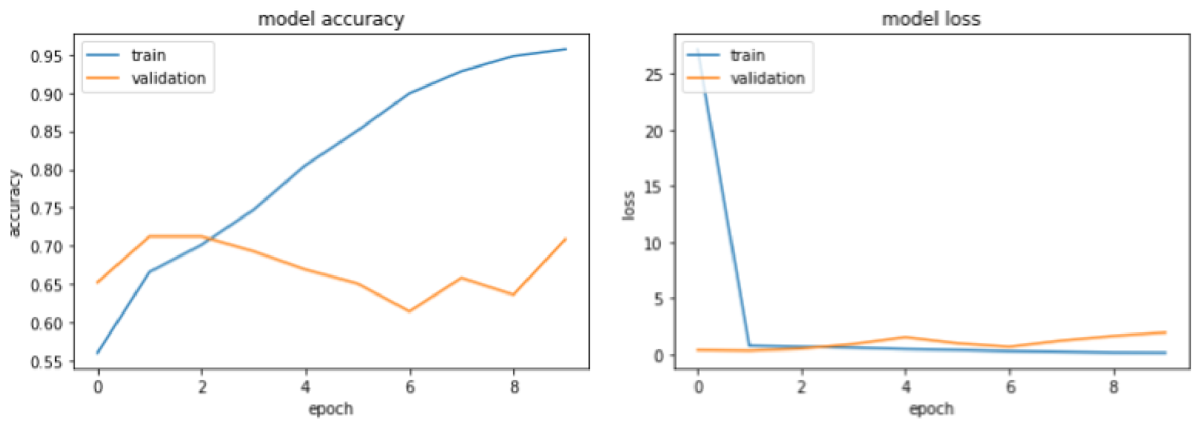

Figure 17. Learning curves model accuracy (left) and model loss (right) of the scratch model trained on augmented images.

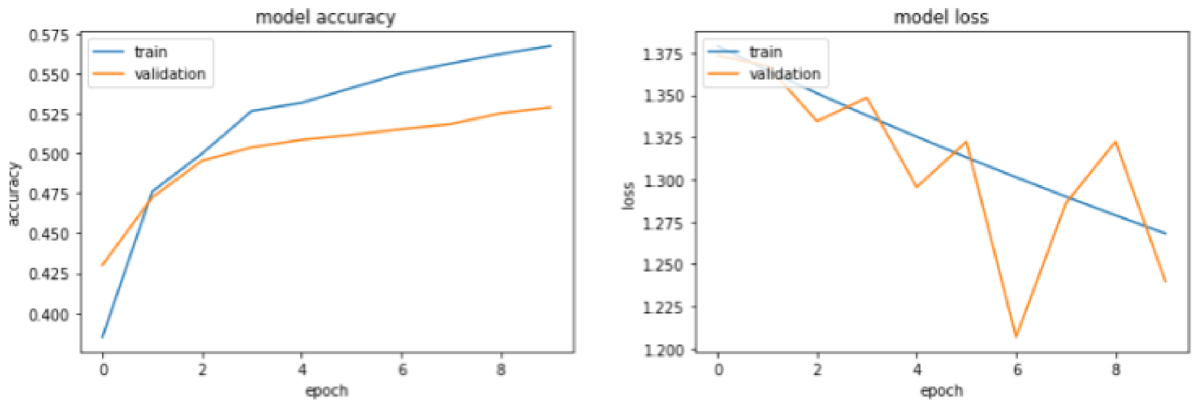

Figure 18. Learning curves model accuracy (left) and model loss (right) of the VGG-16 model trained on augmented images.

Table 5. Summary of the training process of the trained models.

Model	100	200	300	400	500	500-aug	VGG-16	VGG-16 aug
Train accuracy	95%	96%	97%	97%	97%	95%	58%	56%
Validation accuracy	66%	58%	55%	55%	54%	71%	46%	53%
Train loss (error)	0.5	0.5	0.5	0.5	0.5	0.1	1.33	1.27
Validation loss (error)	0.5	0.5	0.5	0.5	0.5	0.3	1.35	1.21
Runtime per epoch (seconds)	400	450	500	550	600	4800	600	3200

4.2 Model performance

After the training of the models was done, the trained models were used to classify the land cover in images of the previously defined test image dataset. In this section, the model performances are shown and discussed. This is done for the scratch models and VGG-16 model separately. Furthermore, the results of the models that are trained on the augmented dataset are shown and compared with the non-augmented results. All models were tested on the same test dataset, containing 200 images (50 per each land cover class). Among those models, the best performing model was used for generating urban land cover maps.

4.2.1 Models from scratch

In order to see the effect on the performance of the models trained on different image dataset sizes, predictions were made on the test image dataset. Figure 19 presents those predictions made by the models separately. In addition, the performance scores of the models are summarized in Table 6. Remarkably, the accuracy appeared to increase when a larger training dataset size was used. Likewise, the F1-score of the trained models showed a higher score when a larger training dataset was used. The best performance can be found in the 500-dataset, where the accuracy resulted in 76%. It should be noted that the accuracy, recall and precision of the 100-dataset were higher than the performance of the 200-dataset. However, the F1-score of both models resulted in 65%. Overall can be stated that the larger the dataset size the better the performance of the model. The model with the best performance was the model that is trained on the 500-dataset. This model is from now on referred to as the 'scratch' model. The details of this best performing scratch model are discussed in more detail in the following paragraph.

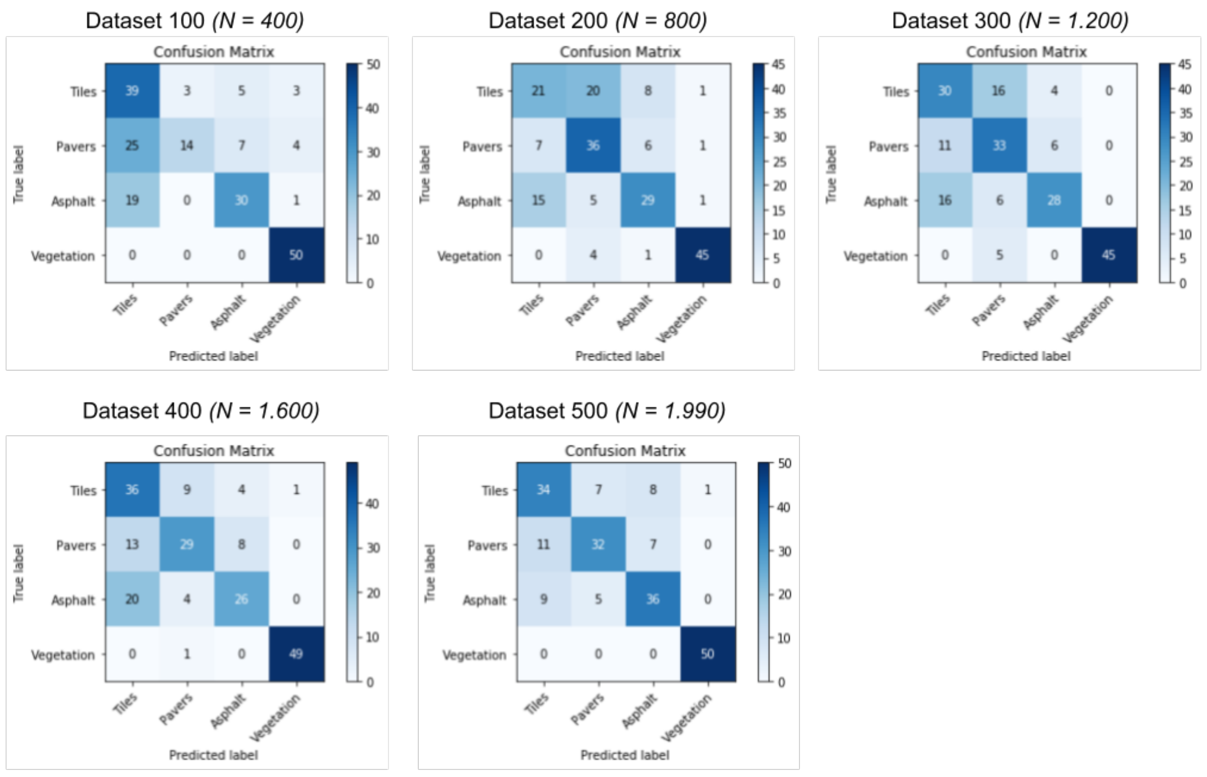

Figure 19. Confusion matrices of the predictions resulting from the models trained on different dataset sizes.

Table 6. Performance scores of the model trained on different dataset sizes.

Measure	100	200	300	400	500
Accuracy	67%	66%	68%	70%	76%
Recall	72%	66%	68%	72%	76%
Precision	67%	66%	70%	70%	76%
F1-Score	65%	65%	69%	70%	76%

The model trained on the 500-dataset had a varying degree of success according to the statistical measures. First of all, the overall accuracy of this model was 76%. This means that 152 out of 200 images were predicted correctly. The results of the model training, presented in Table 7, show a remarkably high performance of the vegetation class, with a precision score of 1 and a recall score of 0,98. This resulted in an F1-score of 99%. This means that the model was able to accurately predict land cover in vegetation images. The model, however, had more difficulties with the other three classes. The second-best performing class was asphalt. For this class, 36 out of 50 images were correctly predicted as asphalt. The recall score of this class was 0,71 and the precision was 0,72. Furthermore, the models appeared to have the most difficulties with classifying the classes pavers and tiles. The tile class had an accuracy of 72%: 34 out of 50 images were correctly classified. Also, this class had the lowest recall score of 0,63 and a precision score of 0,68.

Table 7. Performance scores of the scratch model trained on the 500-dataset.

Class	Recall	Precision	F1-Score
Tiles	0,63	0,68	0,65
Pavers	0,73	0,64	0,68
Asphalt	0,71	0,72	0,71
Vegetation	0,98	1,00	0,99
Accuracy:			**0,76**

4.2.2 Augmented scratch model

In this section, the results of the models that are trained on the augmented image dataset are discussed and compared with the model trained on non-augmented data. To start with, the overall accuracy of the augmented scratch was 73%. In total, 147 out of 200 images were predicted correctly. First of all, the predictions of this model for the vegetation class were similar to the predictions made by the non-augmented scratch model. However, this augmented model classified 4 vegetation images as a different land cover type. The second-best performing class was asphalt. There were 35 asphalt images classified correctly. The recall and precisions scores were respectively 0,76 and 0,70. In contrast to the non-augmented scratch model, the tile class performed better than the paver class. Also, the F1-score of tiles was slightly higher than the score of the pavers. Remarkable is that the recall score of the tiles class was relatively high (0,80), and the precision was relatively low (0,53). The opposite applied to the scores of the paver class: 0,52 for recall and 0,79 for precision. The model appeared to have the most difficulties in classifying the difference between both classes. The predictions of this model are reported in Figure 20 and the performance scores of this model are presented in Table 8.

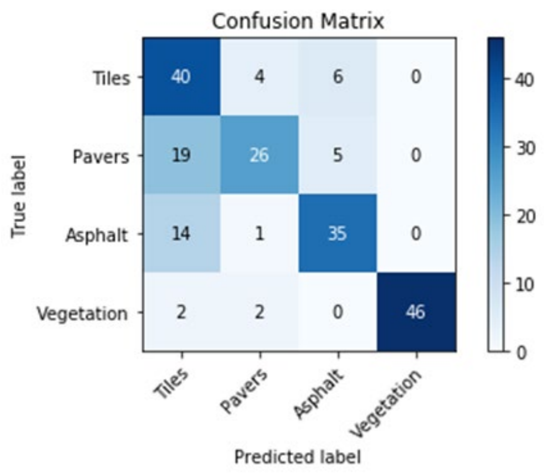

Figure 20. Confusion matrix of the predicted image classes of model trained on augmented images.

Table 8. Performance scores of the scratch model trained on augmented images.

Class	Recall	Precision	F1-Score
Tiles	0,80	0,53	0,64
Pavers	0,52	0,79	0,63
Asphalt	0,70	0,76	0,73
Vegetation	0,92	1,00	0,96
Accuracy:			**0,73**

Table 9 presents the performances of both the non-augmented scratch model and the augmented scratch model. The main difference between the non-augmented model and the augmented model is the overall model accuracy: 76% for the non-augmented model over 73% for the augmented model. In addition, the augmented model resulted in a lower score of the overall accuracy (73%), recall (63%), and F1-score (74%). However, the precision of the augmented model scored 1% higher. This is mainly due to the higher precision of the paver and asphalt classes.

Table 9. Comparison of the performances between the non-augmented and augmented scratch model.

Measure	Non-augmented scratch model	Augmented scratch model
Accuracy	76%	73%
Recall	76%	73%
Precision	76%	77%
F1-Score	76%	74%

4.2.3 VGG-16

The model trained based on the pre-trained VGG-16 is tested on the same images as the scratch models. The results of the predictions made by the VGG 16 are presented in (Figure 21). The overall accuracy of this model is 26%, which is significantly lower than the accuracy of the scratch model. Remarkably, only one image is classified as tiles. Moreover, this classification appears to be a False Positive classification. Apparently, during the training, this model was not able to find patterns in images of tiles. The best performances of this model can be found for the asphalt class, with F1-score of 0,37 (Table 10).

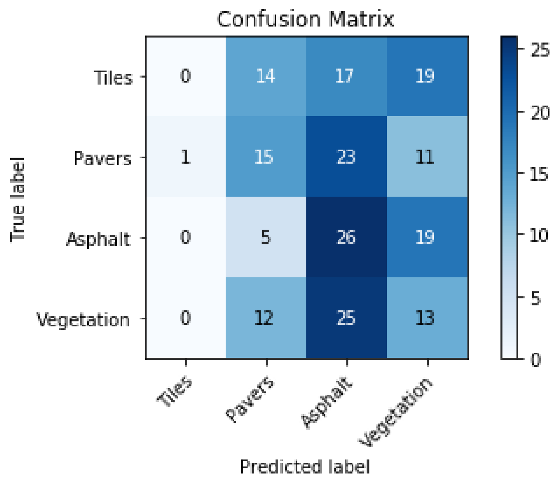

Figure 21. Confusion matrix of the predicted image classes the VGG-16 model.

Table 10. Performance scores of the VGG-16 model trained.

Class	Recall	Precision	F1-Score
Tiles	0,00	0,00	0,00
Pavers	0,30	0,33	0,31
Asphalt	0,52	0,29	0,37
Vegetation	0,26	0,21	0,23
Accuracy:			**0,27**

Training the VGG-16 model on the augmented images had a negative effect on the performance of the pre-trained model. The predictions and performances of this model are presented in Figure 22 and Table 11 respectively. The augmented VGG-16 model resulted in an accuracy of 24%, which is 2% lower than the model trained on non-augmented images. Figure 22 shows that the images were mainly predicted as asphalt and vegetation. Also, compared with the non-augmented VGG-16 model, more images were predicted as tile. This resulted in a F1-score of 0,06 of the tile class. It should be noted that the vegetation class, with a F1-score of 0,33, was the best performing class of this model.

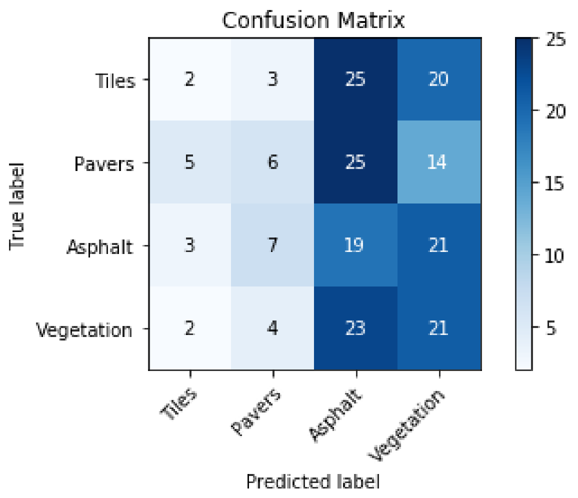

Figure 22. Confusion matrix of the predicted image classes the VGG-16 model trained on augmented images.

Table 11. Performance scores of the VGG-16 model trained on augmented images.

Class	Recall	Precision	F1-Score
Tiles	0,04	0,17	0,06
Pavers	0,12	0,30	0,17
Asphalt	0,38	0,21	0,27
Vegetation	0,42	0,28	0,33
Accuracy:			**0,24**

4.3 Urban land cover classification of test area

4.3.1 Near distance and angle

In this section, the results of the application of the workflow are shown and discussed. Based on the previously trained CNN, the urban land cover in the study area is predicted. This was done for the cable network which was introduced in section 3.6. The cable network of the test area consists of 2.929 segments of two meters. For every cable network segment, the distance to the nearest an image capture location was determined with the method described in section 3.6.1 function. Only the segments located within a range of 7,5 meters from the image capture locations were considered. The total number of cable segments located within this range was 1.099. Remarkably, most segments (402) were located within a range of 3,5 to 4,5 meters. Furthermore, the angle of every cable network segment to the nearest image capture location was determined. Only the segments located within a range of 7,5 meters of the locations were considered. The distribution of the cardinal directions appeared to be balanced for the test area. Although, the north direction is overrepresented and the southwest direction is underrepresented. Figure 23 shows the distribution of the near distances and cardinal directions of the segments.

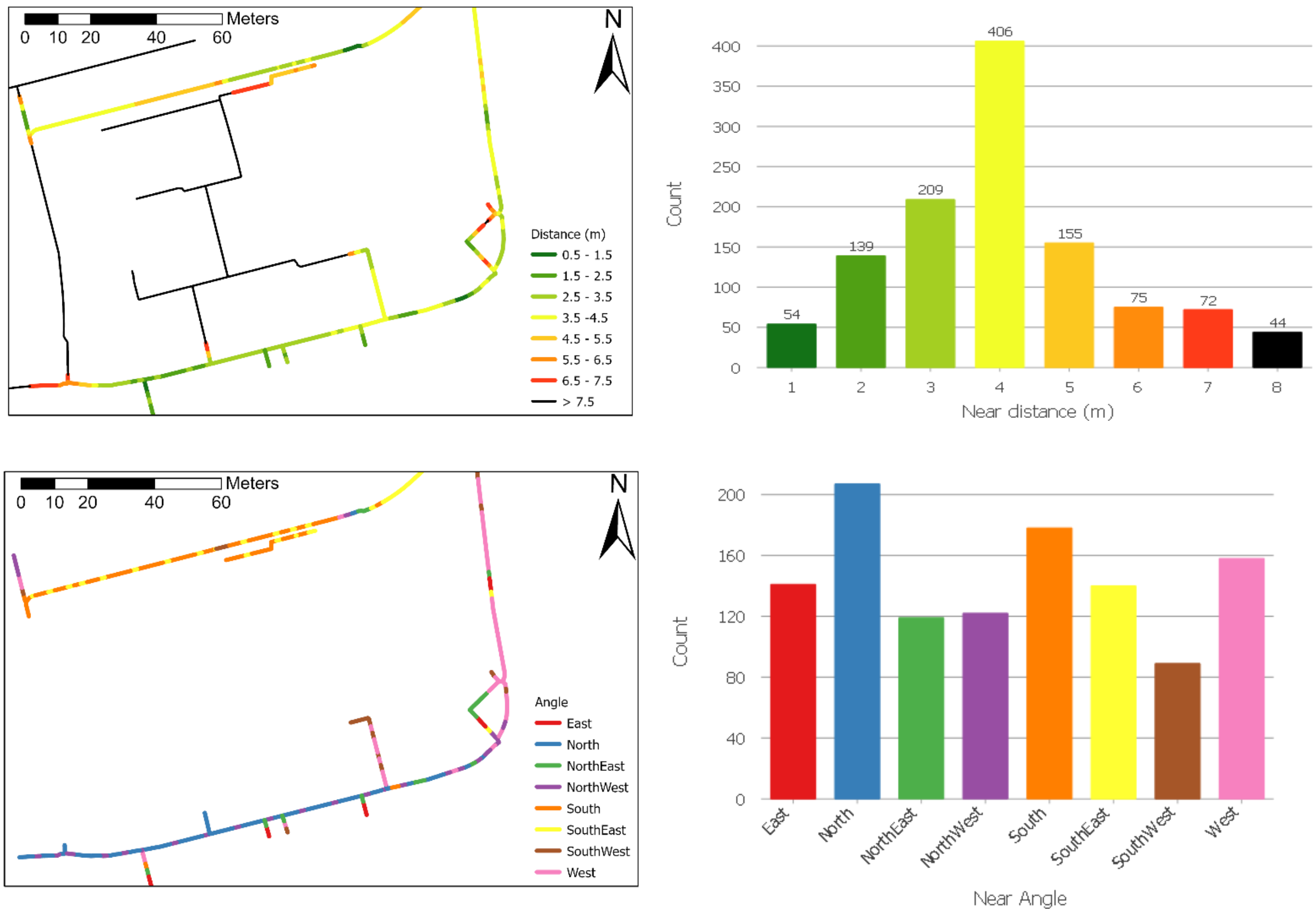

Figure 23. Maps of the spatial distribution of the Near distance (top) and angle (bottom). Besides the maps, graphs are presenting the count of the cable segments per distance and angle.

4.3.2 Urban land cover prediction

In order to be able to predict urban land cover for the image capture location, street view images of the test area were collected and cropped according to the method described in section 3.4. The previously calculated distance and angle were used for this process as well. Furthermore, the weights of the scratch model were used for urban land cover prediction. This process resulted in a land cover classification for every extracted and cropped street view image. The land cover predictions resulting from the model were joined on the cable network. Lastly, in order to see the effect of data augmentation on the performance, the weights of both CNN models with and without data augmentation were used.

The predicted urban land cover is visualized in Figure 24. In total, urban land cover was predicted for 1.099 segments. From the graph can be learned that the predicted urban land cover of this area mainly consists of pavers, tiles, and vegetation. In more detail, the model predicted 5 asphalt segments, 289 pavers segments, 517 tiles segments and 288 vegetation segments. 1.100 segments were not classified since no corresponding image capture locations were found close enough to a cable segment.

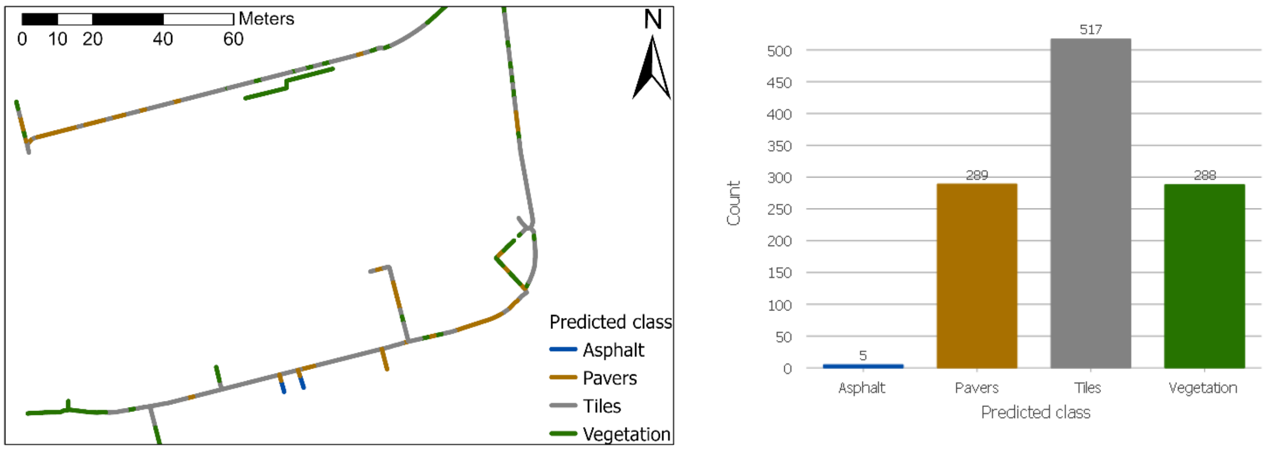

Figure 24. Visualization of the predicted urban land cover of each cable segment by the non-augmented model.

The classified urban land cover resulting from the augmented model is visualized in Figure 25. The classified land cover of this model consists of 34 asphalt segments, 297 paver segments, 611 tiles segments and 157 vegetation segments. Table 12 presents the differences between the amount of non-augmented and the augmented model classifications per land cover class. First of all, the number of predicted paver classes was similar for both models. However, more differences were present for the other classes. The most striking differences occurred for the asphalt and vegetation class. In comparison to the non-augmented model, the augmented model classified 34 asphalt segments, while the non-augmented model classified 5 asphalt segments. Furthermore, the augmented model classified 157 asphalt segments, while the non-augmented model classified 288 asphalt segments.

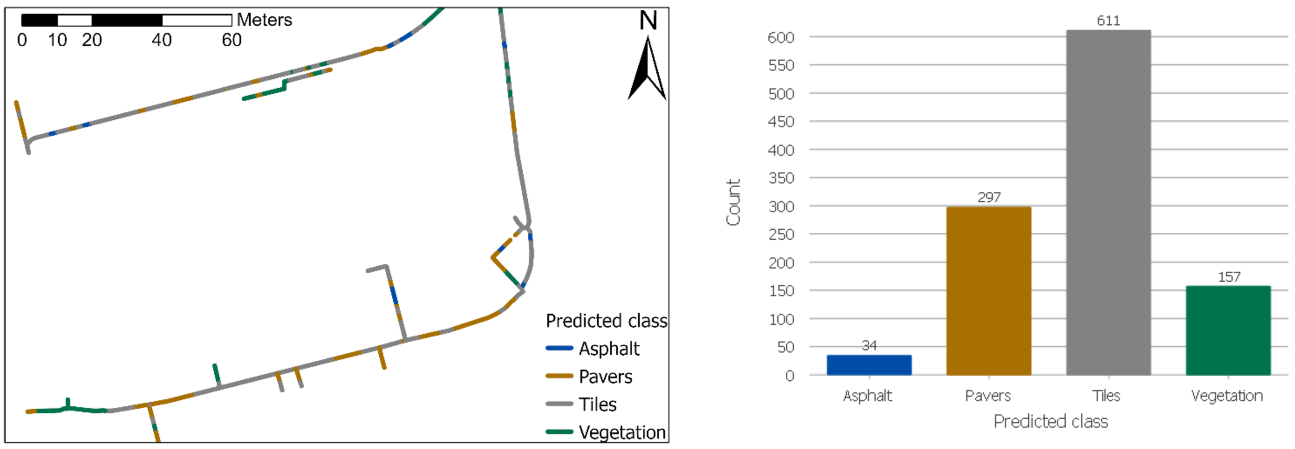

Figure 25. Visualization of the predicted urban land cover of each cable segment by the augmented model.

Table 12. Summary of the number of predicted images per class by the non-augmented model and the augmented model.

Class	Non-Augmented model	Augmented model
Tiles	517	611
Pavers	289	297
Asphalt	5	34
Vegetation	288	157

4.4 Urban land cover validation

In this section, the result of the validation of the previously classified urban land cover of the test area is presented. Furthermore, the performances of the models per distance were validated. Finally, the validation of both non-augmented and augmented are compared.

4.4.1 Non-augmented model validation

Figure 26 shows both true and false predictions of previously predicted urban land cover classes. The overall accuracy of the model is 52%: 520 predicted land cover classes were correct and 480 predicted land cover classes were incorrect. The performance of the model, however, differed per class and is summarized in Table 13. The confusion matrix, presented in Figure 27, shows that many tiles were predicted as pavers and many pavers as tiles. The lower performance of the prediction of tiles and pavers could be attributed to their relatively similar spectral properties. It should be noted that, according to the BGT, the asphalt classes were barely present in the test area. Therefore, the performance of asphalt is not a fair representation. However, it is noteworthy that the model did not predict asphalt for images of the other classes.

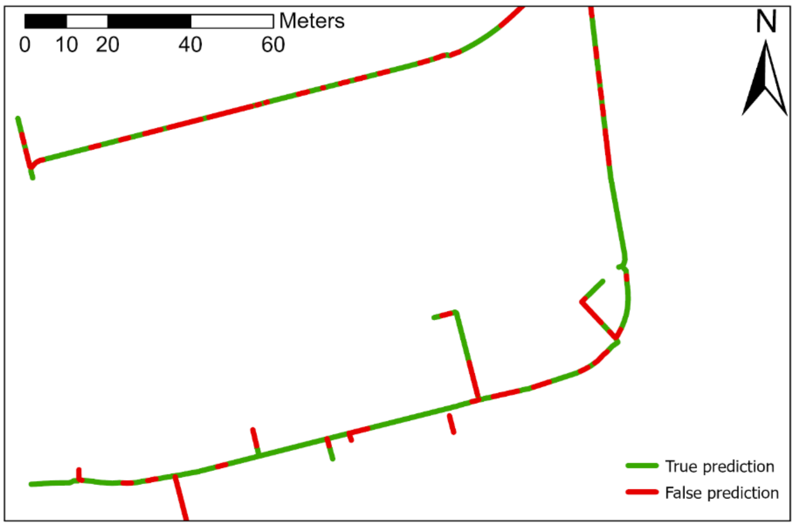

Figure 26. Visualization of the true and false predictions of the urban land cover classification.

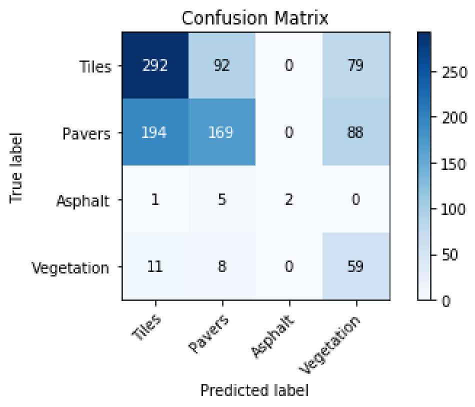

Figure 27. Confusion matrix of the urban land cover predictions classified by the non-augmented model.

Table 13. Performance scores of the urban land cover classification model.

Class	Recall	Precision	F1-score
Tiles	0,63	0,59	0,61
Pavers	0,37	0,62	0,47
Asphalt	0,25	1,00	0,40
Vegetation	0,76	0,26	0,39
Accuracy:			**0,52**

4.4.2 Performance per distance

Although the models were trained on the labelled three-meter images, the urban land cover of other distances was predicted as well. To examine the usefulness of the model more thoroughly, its performance on different distances was examined. If the model can classify urban land cover for different distances and is not limited to land cover on exactly three-meter, the model would be more robust and could be easier be implemented in a real-world application. The confusion matrices of the different distances and the performances are visualized in Appendix B.1. It should be mentioned that for some distances the land cover class asphalt was not included in the confusion matrix. This is because no asphalt existed and/or was predicted for this distance. First of all, Table 14 shows that the model performed the best on images located at a three-meter distance. The accuracy of this distance was 70%. Secondly, the one-meter and two-meter distances performed the worst, with an accuracy of 15% and 4%. This can be explained by the fact that the images of those distances are too close to the car on which the camera is installed. Furthermore, the accuracy of the distances greater than thee-meter had an accuracy ranging between 47% and 55%.

Table 14. Overview of the performance measures per distance.

Measure	Overall	1 meter	2 meters	3 meters	4 meters	5 meters	6 meters	7 meters
Accuracy	52%	15%	42%	70%	52%	49%	55%	51%
Precision	58%	94%	63%	72%	57%	47%	61%	67%
Recall	52%	15%	42%	70%	52%	49%	55%	51%
F1-score	53%	21%	50%	70%	51%	47%	55%	53%

The difference in performances per distances is substantiated in the graphs in Figure 28. Those graphs show the F1-score for the distances per class separately. Similar to the model its overall performance, the highest F1-scores of the classes tiles, pavers and vegetation were reached for the three-meter distances. This was not the case, however, for the asphalt class. This is because no BGT asphalt class existed for the three-meter segments.

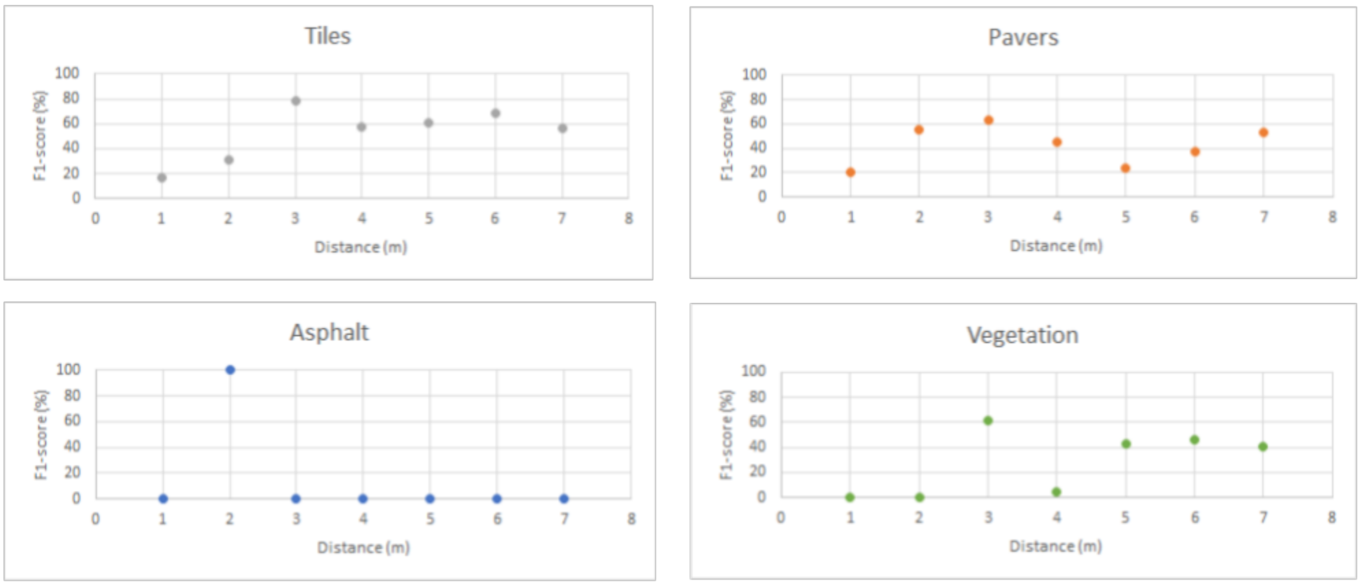

Figure 28. F1-scores per distance visualized for the four urban land over classes tiles, pavers, asphalt and vegetation.

4.5.2 Augmented model validation

The results of the validation of the classifications made by the augmented model are presented in Figure 29 and Figure 30. The accuracy of the augmented scratch model was 54% (Table 15). The predictions of the augmented trained scratch model seem to reflect the same pattern as the classification of the non-augmented scratch model. Remarkably, the performance scores of the asphalt appeared to be zero. This is because no asphalt segment was correctly classified by this model. Moreover, the augmented model predicts 29 False Positive asphalt classes.

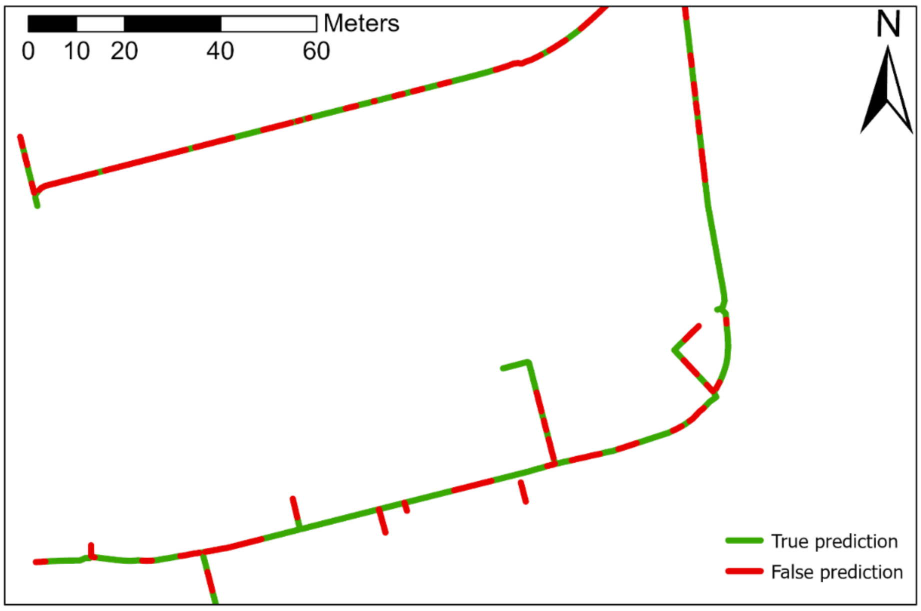

Figure 29. Visualization of the true and false predictions of the urban land cover classification of the augmented model.

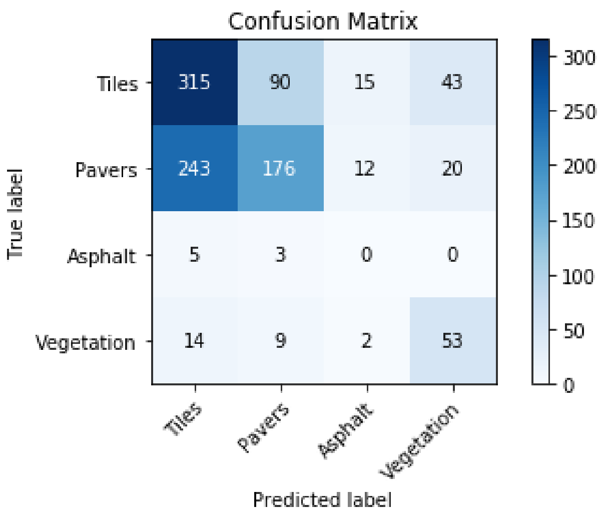

Figure 30. Confusion matrices of the classifications of the images classified by the model trained on augmented images.

Table 15. Performance scores of the urban land cover classification model trained on augmented images.

Class	Recall	Precision	F1-Score
Tiles	0,68	0,55	0,61
Pavers	0,39	0,63	0,48
Asphalt	0,00	0,00	0,00
Vegetation	0,68	0,46	0,55
Accuracy:			**0,54**

Table 16 shows the weighted average of the performance measures of both non-augmented and augmented models. In comparison, the augmented model had a 2% higher accuracy, 2% higher recall and 1% higher F1-score than the non-augmented model. In contrast, the precision of the non-augmented model appeared to be 1% higher than the augmented model.

Table 16. Summary of performances of the urban land cover classification by the non-augmented model and the augmented model.

Measure	Non-augmented model	Augmented model
Accuracy	52%	54%
Precision	58%	57%
Recall	52%	54%
F1-score	53%	54%

5. DISCUSSION

This research aimed to develop a geo-computational workflow that is able to classify urban land cover. The results of this research shown that a convolutional neural network trained from scratch is capable of classifying urban land cover on a highly challenging dataset using purely supervised learning. Although this research has accomplished a workflow that is able to classify land cover based on street view imagery, the issue deserves further additional research. The following sections address multiple suggestions for improvement for future work.

First of all, this research was done for specific study areas in the Dutch city Bergen op Zoom. The proposed workflow performed successfully in this area. However, especially in the Netherlands, the characteristics of street cover, such as design, structure, and texture, can differ substantially between cities, districts, and even neighbourhoods. At this stage of research, it is uncertain how the workflow performs in other areas. In future research, it would be interesting to see how the model responds to the different characteristics of urban land cover in different areas. Elaborating, the images that were used during this research are captured within a short time period under the same weather and light conditions. This is a disadvantage because this decreases the robustness of the model. With the use of data augmentation, different light conditions were simulated. However, this does not cover all different conditions such as rain or shadow effects. In future research, it would be interesting to see how the workflow performs in a different city and under different weather conditions.

The second improvement refers to the size of the dataset that was used for this research. The image extraction and cropping method provided a large image dataset that can be used for training the models. However, a significantly high amount of images were not useable for training due to the random extraction of the street view imagery. This random extraction resulted in images that include, for example, multiple classes or contained classes that were not used for this research. The result of the random extraction was an image dataset with unbalanced classes. At this point, the training image dataset consisted of 7.345 images, of which only 25% was usable for the training of the model. In future research, the street view extraction method could be improved by only extracting images that include the land cover classes of interest. For example, this could be done with the use of the reference map (e.g. BGT) and calculating the angle and distance to the land cover of interest. After combining the results of that calculation, street view images that are facing the land cover of interest could be extracted. Extending the training image dataset with more labelled images could help to increase the performance of the CNN.

In this research, the models were trained to recognize images the urban land cover classes tiles, pavers, asphalt, and vegetation. However, there are more types of urban land cover that can be classified such as bare ground, water, different types of vegetation, decorative pavement, stones and marble stones. Further research could integrate the other classes in the model. In addition, the street view data used for this research is annually collected for every accessible road in the Netherlands. In further research, land cover change over time could be measured using this workflow when historical images are available. With the use of those historical images, urban land cover change over time could be analyzed.

Furthermore, the model architecture of the used deep learning models could be improved. This study used two different training models: one model built from scratch and one pre-trained model (VGG-16). There exist more pre-trained models that can be used. It would be interesting to see how the performance differs when other pre-trained models are used. Moreover, the model from scratch consisted of three convolutional layers, two pooling layers, and two dense layers. Although the performance of the model is satisfying, further research could try to find another model architecture that results in a higher performance of the model. This can be done by adding more convolutional or pooling layers for example. There is not a rule that states how many layers should be added to the model. Every model acts differently due to different input data. More experiments should lead to a more optimal model architecture.

A limitation of this research is the size of the area that can be classified. This study was bound to the accessibility of roads in an area. Moreover, solely land cover with a maximum range of 7 meters from the capture location was classified. In order to overcome this issue, the workflow could be extended with the use of high resolution (about 10 cm) aerial imagery. Previous researches have argued that high-resolution aerial imagery is able to classify land cover. However, in urban areas top views may be blocked which make street view image classification necessary. A combination of both street view and aerial imagery could make it possible to classify urban land cover that exists outside of the seven-meter range.

Furthermore, due to time limitation, the models in this research were trained on the urban land cover that was located in a range of 2,5 and 3,5 meter from an image capture location. The results of the validation showed that the workflow reached an accuracy of 70% for the classification of land cover that was located within this range. Future research could investigate what the effect of training the models on images with multiple distances is on the performance of the workflow. Moreover, the used cropping frames had a size of one by two meters. In further research, the effect of different cropping frame sizes could be investigated. It would be interesting to see the effect of decreasing the size of the cropping frames on the accuracy of the workflow. Smaller cropping frames could make it possible to classify urban land cover even more precise. However, smaller images provide less training material for the model. This could have an effect on the accuracy of the model as well.

Finally, this research developed a workflow that was able to automatically classify the urban land cover above a cable network. In this case, the cable network was a feature line. However, this workflow is not bounded to this geometry type. The workflow can be adjusted so it could be used to classify features with other geometries such as polygons and points. In order to achieve this, the cropping frames have to be adjusted to the size of the area of interest. This workflow can be applied for other purposes than urban land cover classification. For example, to classify the quality of roads, to recognize litter in urban areas and to recognize objects along the road.

6. Conclusion

6.1 Research question 1

The first research question is: "What are the best parameters and pre-processing strategies for training a deep learning model for classifying urban land cover?"

This question was answered by comparing three different pre-processing strategies. First of all, the training dataset was divided into five separate image datasets, containing 400, 800, 1.200, 1.600 and 1.992 images respectively. The learning curves of the models showed that the larger the dataset, the lower the validation accuracy. However, this was in contrast to the results of the performances test of the different models. The result of this analysis showed that the overall accuracy of the models trained on a larger image dataset performed better, by considering accuracy, recall, precision and F1-score.

Secondly, the performance of the non-augmented and the augmented images were analysed. The results of this analysis showed that the overall accuracy of the models trained on non-augmented images was 1% higher than the model trained on the augmented images. However, the model trained on augmented images scored 1% higher on precision, which means that more of the positive classifications were correct. Based on these results, it can be concluded that training a model to classify urban land cover with augmented images did not significantly improve the performance. However, it makes the trained model more robust and able to classify land cover in images under different conditions as well.

6.2 Research question 2

The second research question is: "What are the performance differences between a pre-trained and newly trained deep learning model?"

In order to answer this sub-question, a comparison between the performances of a pre-trained and a model trained from scratch was made. With the use of transfer learning, the pre-trained VGG-16 model was trained to classify urban land cover images. In conclusion, the model trained from scratch resulted in an accuracy of 76%. The fine-tuned VGG-16 resulted in significantly lower accuracy, namely 27%. Moreover, the overall accuracy resulted in even lower accuracy of 24% after training the model on augmented images. In addition, the learning curve of this model showed that the train and validation loss are both relatively high compared to the loss of the scratch model. This could indicate that the model is not to be able to find patterns in the training data. It can be concluded that the pre-trained VGG-16 model, as used in this research, was not able to successfully predict urban land cover.

6.3 RESEARCH QUESTION 3

The third research question is: "What level of accuracy, precision and recall can be reached in classifying urban land cover using a geo-computational workflow?"

This question was answered by applying the trained scratch models (non-augmented and augmented) for classifying urban land cover above a hypothetical cable network located in the test area. The accuracy of the non-augmented model resulted in 52%, which means that 52% positive classifications were correct. Moreover, the recall score is 52%, the precision score was 58%, and the F1-score was 53%. In addition, the model trained on augmented images was also used for classifying urban land cover in the test area. This model resulted in an overall accuracy of 54%. Furthermore, the recall score was 56%, the precision score was 57% and the F1-score 53%. It can be concluded that this workflow was able to successfully classify 52-53% of the land cover in urban areas.

6.4 RESEARCH QUESTION 4

The fourth research question is: "What is the influence of different image distances on the performance of urban land cover classification?"

To answer this sub-question, the performance of the model trained for urban land cover classification was analysed for distances ranging from one to seven meters separately. Of those distances, the image with a three-meter distance appeared to be the best performing distance with an accuracy of 70%. The distances of one to two meters returned significantly worse results. In addition, the performances of the model on distances of four to seven-meter resulted in accuracy scores ranging between 47% and 55%. It can be concluded that the model has more difficulties in classifying urban land cover correctly when the distance changes.

6.5 MAIN CONCLUSION

Due to urbanization, there is a growing demand for knowledge on urban patterns and their dynamics at multiple spatial scales. Reliable information on urban land cover is important for dealing with this urbanization. The existing methods, such as analysis of remotely sensed imagery, do not help to accurately classify urban land cover. The main objective of this research was to develop a novel geo-computational workflow to automatically classify urban land cover at large scale using high-resolution street view imagery and deep learning.

This study developed a successful method for extracting and pre-processing street view imagery. In addition, the workflow accomplished an overall accuracy of 54% for urban land cover classification. In more detail, the results show that the workflow was able to correctly classify 70% of the urban land cover located on a three-meter distance. However, the accuracy reached by the developed workflow is not sufficient enough for real-world application. By improving the model, that is developed in this research, a higher performance of urban land cover classification can be achieved and the workflow can be applied for the whole of the Netherlands.

7. BIBLIOGRAPHY

Agarap, A. F. (2018). *Deep Learning using Rectified Linear Units (ReLU)*. (1), 2–8. Retrieved from http://arxiv.org/abs/1803.08375

Aggarwal, C. C. (2018). Neural Networks and Deep Learning. In *Neural Networks and Deep Learning*. https://doi.org/10.1007/978-3-319-94463-0

Amara, J., Bouaziz, B., & Algergawy, A. (2017). A deep learning-based approach for banana leaf diseases classification. *Lecture Notes in Informatics (LNI), Proceedings - Series of the Gesellschaft Fur Informatik (GI)*, *266*, 79–88.

Anzanello, M. J., & Fogliatto, F. S. (2011). Learning curve models and applications: Literature review and research directions. *International Journal of Industrial Ergonomics*, *41*(5), 573–583. https://doi.org/10.1016/j.ergon.2011.05.001

Atkinson, M., Gesing, S., Montagnat, J., Taylor, I., Atkinson, M., Gesing, S., … Taylor, I. (2017). Scientific workflows : Past , present and future. *Future Generation Computer Systems*, *75*, 216–227.

Basha, S. H. S., Dubey, S. R., Pulabaigari, V., & Mukherjee, S. (2019). Impact of fully connected layers on performance of convolutional neural networks for image classification. *Neurocomputing*. https://doi.org/10.1016/j.neucom.2019.10.008

Brownlee, J. (2019a). Difference Between a Batch and an Epoch in a Neural Network. Retrieved December 1, 2019, from https://machinelearningmastery.com/difference-between-a-batch-and-an-epoch/

Brownlee, J. (2019b). How to use Learning Curves to Diagnose Machine Learning Model Performance. Retrieved February 23, 2020, from https://machinelearningmastery.com/learning-curves-for-diagnosing-machine-learning-model-performance/

Brownlee, J. (2019c). Supervised and Unsupervised Machine Learning Algorithms. Retrieved February 16, 2020, from https://machinelearningmastery.com/supervised-and-unsupervised-machine-learning-algorithms/

Cao, R., Zhu, J., Tu, W., Li, Q., Cao, J., Liu, B., … Qiu, G. (2018). Integrating aerial and street view images for urban land use classification. *Remote Sensing*, *10*(10), 1–23. https://doi.org/10.3390/rs10101553

Chellapilla, K., Puri, S., & Simard, P. (2006). *High performance convolutional neural networks for document processing*. La Baule (France).

Cireşan, D., Meier, U., Masci, J., & Schmidhuber, J. (2012). Multi-column deep neural network for traffic sign classification. *Neural Networks*, *32*, 333–338. https://doi.org/10.1016/j.neunet.2012.02.023

CycloMedia. (n.d.). Retrieved October 16, 2019, from https://www.cyclomedia.com/en

Different Kinds of Convolutional Filters. (2017). Retrieved February 25, 2020, from https://www.saama.com/different-kinds-convolutional-filters/

Digitaleoverheid. (n.d.). Retrieved November 8, 2019, from https://www.digitaleoverheid.nl/overzicht-van-alle-onderwerpen/basisregistraties-en-afsprakenstelsels/inhoud-basisregistraties/bgt/

Garcia, C., & Delakis, M. (2002). A neural architecture for fast and robust face detection. *Proceedings - International Conference on Pattern Recognition*, *16*(2), 44–47. https://doi.org/10.1109/icpr.2002.1048232

Geostandaarden - Asphalt. (n.d.). Retrieved February 5, 2020, from https://definities.geostandaarden.nl/concepten/imgeo/doc/begrip/asfalt

Geostandaarden - Tegels. (n.d.). Retrieved February 6, 2020, from https://definities.geostandaarden.nl/imgeo/doc/begrip/Tegels_fysiekVoorkomen WegPlus

Gupta, D. (2017). Architecture of Convolutional Neural Networks (CNNs) demystified. Retrieved December 5, 2019, from https://www.analyticsvidhya.com/blog/2017/06/architecture-of-convolutional-neural-networks-simplified-demystified/

Hensman, P., & Masko, D. (2015). *The Impact of Imbalanced Training Data for Convolutional Neural Networks*.

Huang, H., Wang, J., & Abudureyimu, H. (2012). Maximum F1-score discriminative training for automatic mispronunciation detection in computer-assisted language learning. *13th Annual Conference of the International Speech Communication Association 2012, INTERSPEECH 2012*, *1*, 814–817.

Kamilaris, A., & Prenafeta-Boldú, F. X. (2018). Deep learning in agriculture: A survey. *Computers and Electronics in Agriculture*, *147*, 70–90. https://doi.org/10.1016/j.compag.2018.02.016

Kang, J., Körner, M., Wang, Y., Taubenböck, H., & Zhu, X. X. (2018). Building instance classification using street view images. *ISPRS Journal of Photogrammetry and Remote Sensing*, *145*, 44–59. https://doi.org/10.1016/j.isprsjprs.2018.02.006

Ker, J., Wang, L., Rao, J., & Lim, T. (2017). Deep Learning Applications in Medical Image Analysis. *IEEE Access*, *6*, 9375–9379. https://doi.org/10.1109/ACCESS.2017.2788044

Keras Documentation. (n.d.). Retrieved December 4, 2019, from https://keras.io/

Keskar, N. S., Nocedal, J., Tang, P. T. P., Mudigere, D., & Smelyanskiy, M. (2019). On large-batch training for deep learning: Generalization gap and sharp minima. *5th International Conference on Learning Representations, ICLR 2017 - Conference Track Proceedings*, 1–16.

Krizhevsky, A., Sutskever, I., & Hilton, G. E. (2012). Imagenet classification with deep convolutional neural networks. *Advances in Neural Information Processing Systems*, 1097–1105. https://doi.org/10.1201/9781420010749

Kussul, N., Lavreniuk, M., Skakun, S., & Shelestov, A. (2017). Deep Learning Classification of Land Cover and Crop Types Using Remote Sensing Data. *IEEE Geoscience and Remote Sensing Letters*, *14*(5), 778–782. https://doi.org/10.1109/LGRS.2017.2681128

Lecun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature*, *521*(7553), 436–444. https://doi.org/10.1038/nature14539

Ling, C. X. (1995). Overfitting and generalization in learning discrete patterns. *Neurocomputing*, *8*(3), 341–347.

Ma, L., Liu, Y., Zhang, X., Ye, Y., Yin, G., & Johnson, B. A. (2019). Deep learning in remote sensing applications: A meta-analysis and review. *ISPRS Journal of Photogrammetry and Remote Sensing*, *152*, 166–177. https://doi.org/10.1016/j.isprsjprs.2019.04.015

MathWorks. (2019). Pretrained VGG-16 convolutional neural network. Retrieved December 1, 2019, from https://www.mathworks.com/help/deeplearning/ref/vgg16.html#bvo3twr-1.mw_6dc28e13-2f10-44a4-9632-9b8d43b376fe

Najafabadi, M. M., Villanustre, F., Khoshgoftaar, T. M., Seliya, N., Wald, R., & Muharemagic, E. (2015). Deep learning applications and challenges in big data analytics. *Journal of Big Data*, *2*(1), 1–21. https://doi.org/10.1186/s40537-014-0007-7

Ning, F., Delhomme, D., Lecun, Y., Piano, F., Bottou, L., Barbano, P. E., … Lecun, Y. (2005). Toward automatic phenotyping of developing embryos from videos. *EEE Transactions on Image Processing, Institute of Electrical and Electronics Engineers*, *14*(9), 1360–1371.

Parmar, R. (2018). Demystifying Optimizations for machine learning. Retrieved December 3, 2019, from https://towardsdatascience.com/demystifying-optimizations-for-machine-learning-c6c6405d3eea

Perez, L., & Wang, J. (2017). *The Effectiveness of Data Augmentation in Image Classification using Deep Learning*. Retrieved from http://arxiv.org/abs/1712.04621

Polamuri, S. (2017). Difference between softmax funtion and sigmoid function. Retrieved February 26, 2020, from https://dataaspirant.com/2017/03/07/difference-between-softmax-function-and-sigmoid-function/

Pro.ArcGIS.com. (n.d.). Retrieved February 6, 2020, from https://pro.arcgis.com/en/pro-app/tool-reference/analysis/near.htm

Renuka, J. (2016, September 9). Accuracy, Precision, Recall & F1 Score: Interpretation of Performance Measures - Exsilio Blog. Retrieved February 4, 2020, from https://blog.exsilio.com/all/accuracy-precision-recall-f1-score-interpretation-of-performance-measures/

Saha, S. (2018). A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way. Retrieved December 1, 2019, from https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53

Schmidhuber, J. (2015). Deep Learning in neural networks: An overview. *Neural Networks*, *61*, 85–117. https://doi.org/10.1016/j.neunet.2014.09.003

scikit-learn - precision and recall. (n.d.). Retrieved February 7, 2020, from https://scikit-learn.org/stable/auto_examples/model_selection/plot_precision_recall.html?highli

Siegmund, D., Prajapati, A., Kirchbuchner, F., & Kuijper, A. (2018). An integrated deep neural network for defect detection in dynamic textile textures. *International Workshop on Artificial Intelligence and Pattern Recognition*, (September), 77–84. https://doi.org/10.1007/978-3-030-01132-1

Smith, S. L., Kindermans, P. J., Ying, C., & Le, Q. V. (2018). Don't decay the learning rate, increase the batch size. *6th International Conference on Learning Representations, ICLR 2018 - Conference Track Proceedings*, (2017), 1–11.

Sorokina, K. (2018). Image Classification with Convolutional Neural Networks. Retrieved December 5, 2019, from https://medium.com/@ksusorokina/image-classification-with-convolutional-neural-networks-496815db12a8

Svozil, D., Kvasnicka, V., & Pospichal, J. (1997). Introduction to multi-layer feed-forward neural networks. *Chemometrics and Intelligent Laboratory Systems1*, *39*, 43–62.

Tang, T. A., Mhamdi, L., McLernon, D., Zaidi, S. A. R., & Ghogho, M. (2016). Deep learning approach for Network Intrusion Detection in Software Defined Networking. *Proceedings - 2016 International Conference on Wireless Networks and Mobile Communications, WINCOM 2016: Green Communications and Networking*, 258–263. https://doi.org/10.1109/WINCOM.2016.7777224

Thakur, V. (2019). scienceabc.com. Retrieved November 6, 2019, from https://www.scienceabc.com/innovation/what-is-the-difference-between-deep-learning-and-artificial-intelligence.html

Tracewski, L., Bastin, L., & Fonte, C. C. (2017). Repurposing a deep learning network to filter and classify volunteered photographs for land cover and land use characterization. *Geo-Spatial Information Science*, *20*(3), 252–268. https://doi.org/10.1080/10095020.2017.1373955

Xing, F., & Yang, L. (2016). Machine learning and its application in microscopic image analysis. In *Machine Learning and Medical Imaging*. https://doi.org/10.1016/B978-0-12-804076-8.00004-9

Xu, G., Zhu, X., Fu, D., Dong, J., & Xiao, X. (2017). Automatic land cover classification of geo-tagged field photos by deep learning. *Environmental Modelling and Software*, *91*, 127–134. https://doi.org/10.1016/j.envsoft.2017.02.004

Yin Low, J. (2020). Supervised Learning vs. Unsupervised Learning. Retrieved February 12, 2020, from https://blog.supahands.com/2020/01/21/supervised-learning-vs-unsupervised-learning/

Yu, W., Yang, K., Bai, Y., Xiao, T., Yao, H., & Rui, Y. (2016). *Visualizing and Comparing AlexNet and VGG using Deconvolutional Layers*. Retrieved from http://www.robots.ox.ac.uk/

8. Appendix

A. Learning curves

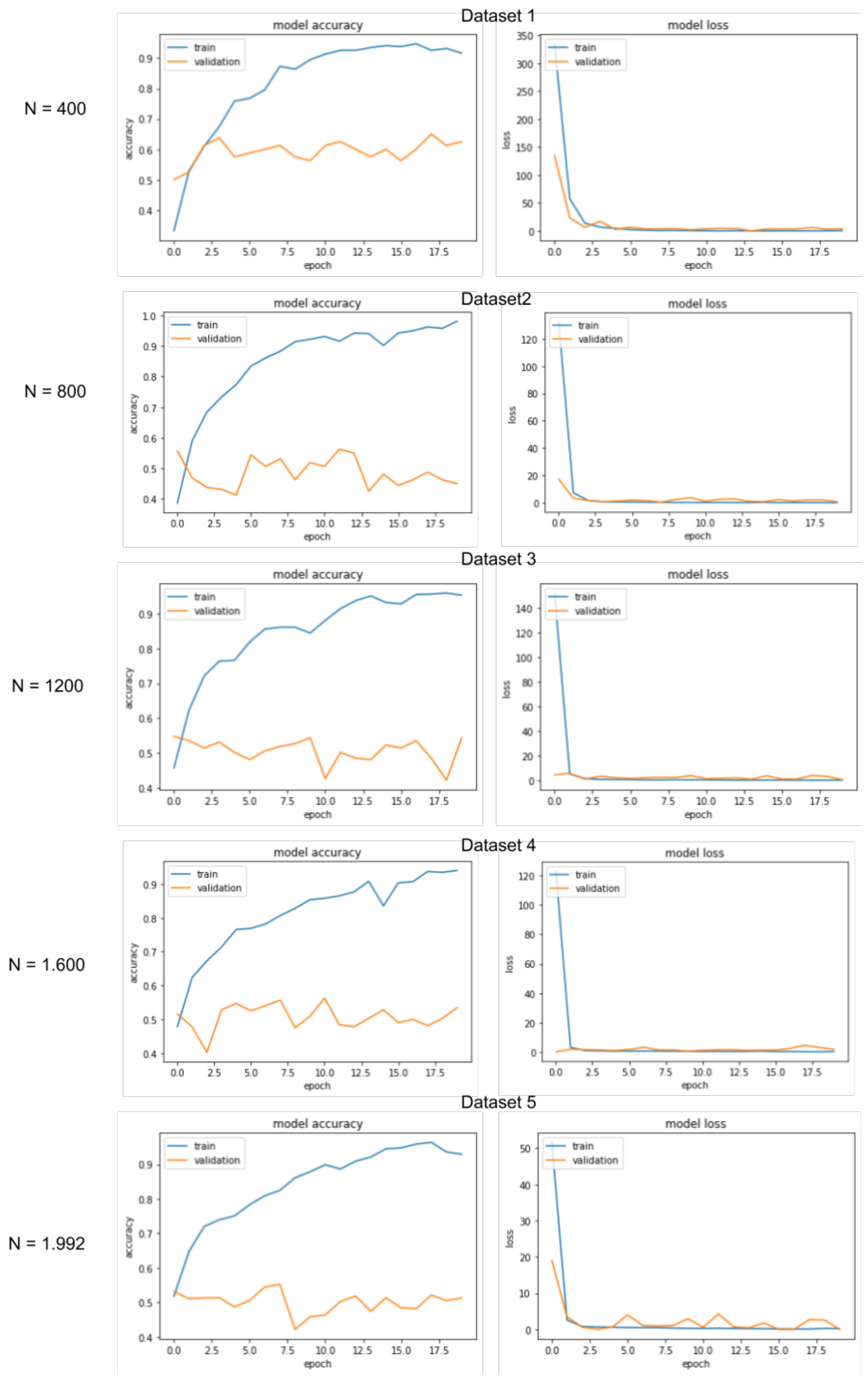

Figure A.1. Learning curves, of the model accuracy and loss, plotted for the models trained on the different image datasets.

B. PERFORMANCES PER DISTANCE

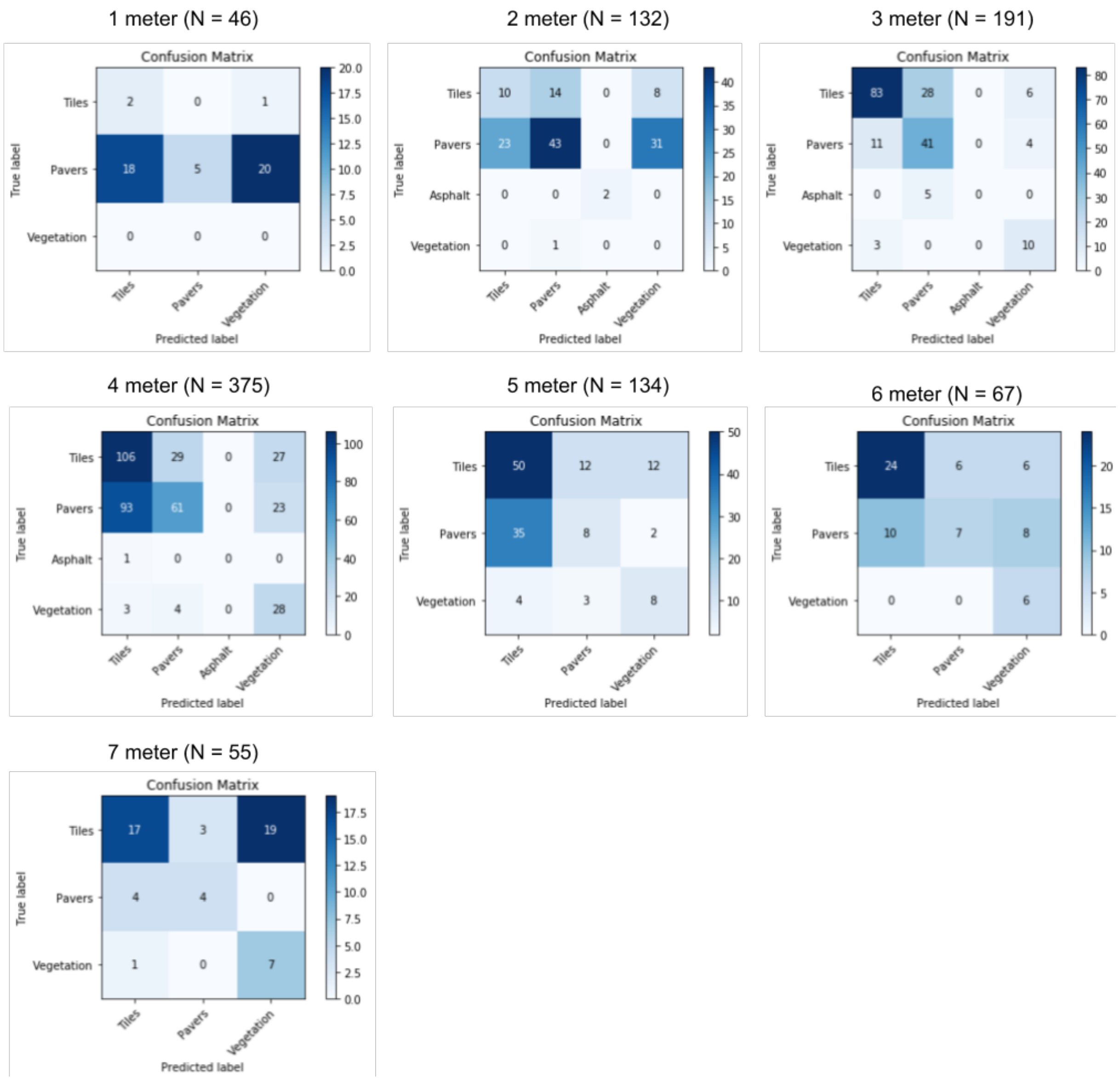

Figure B.1. Confusion matrices of the classifications of the images on distances ranging from one to seven meter.

C. Near maps

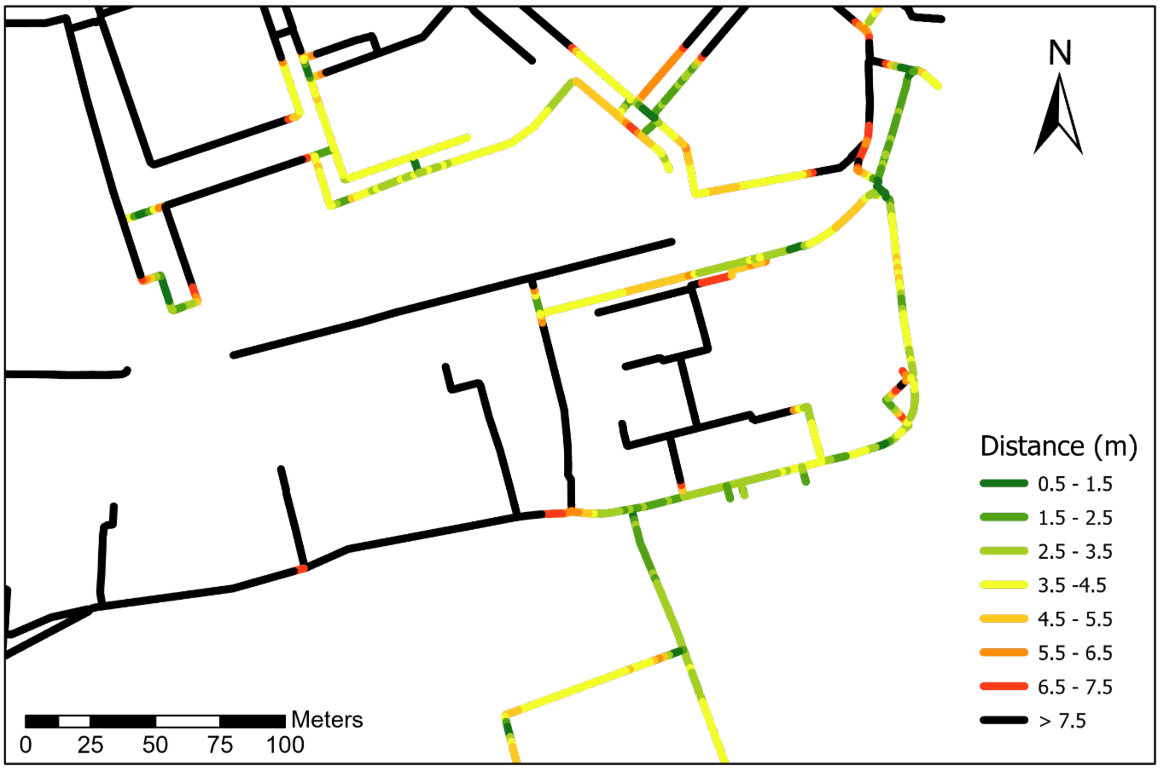

Figure C.1. The nearest distance between cable network segments and the image capture locations in a range of 7.5 meters.

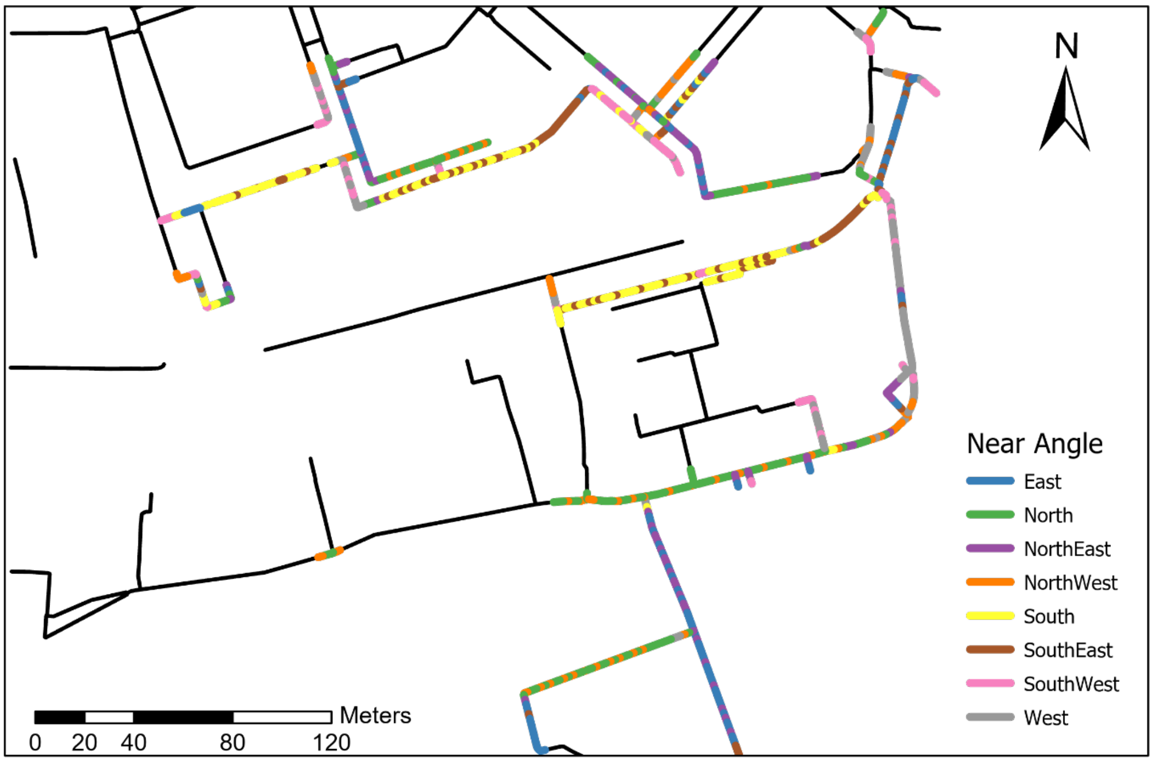

Figure C.2. Cardinal direction of the cable network segments to the closest image capture locations in a range of 7.5 meters.

D. Urban land cover classification maps

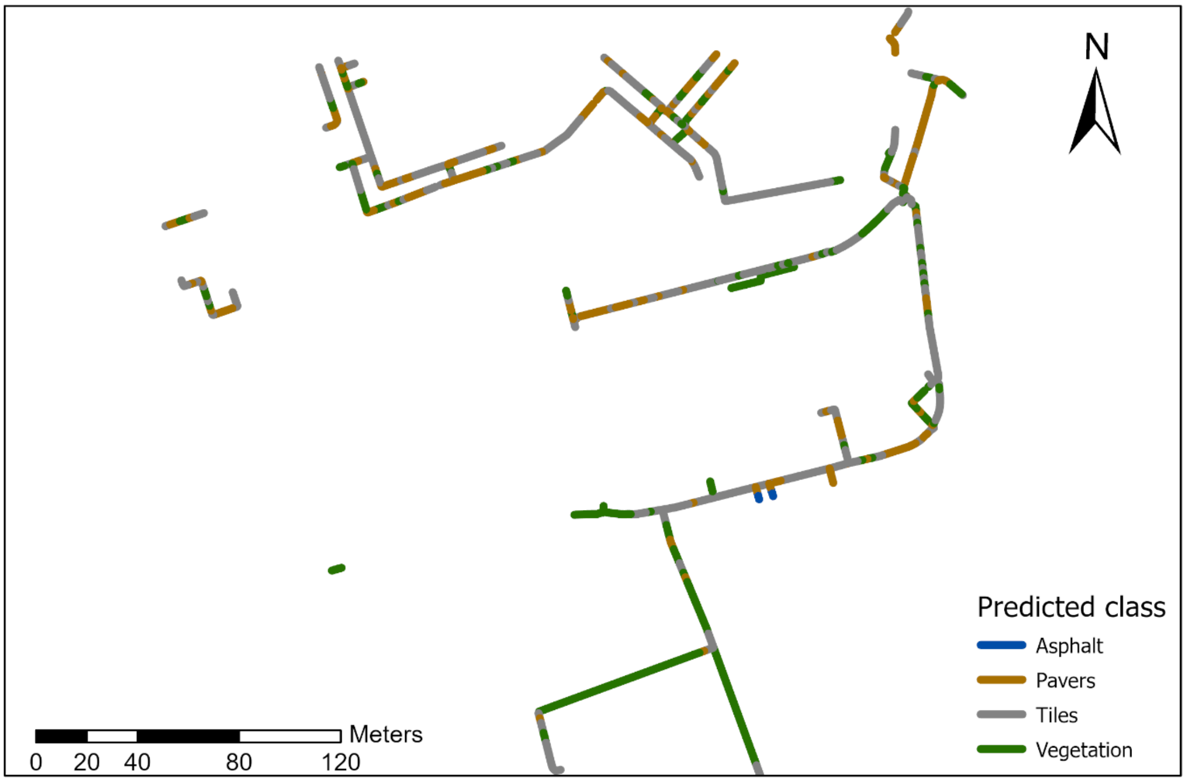

Figure D.1. Visualization of the predicted urban land cover of each cable segment by the non-augmented model.

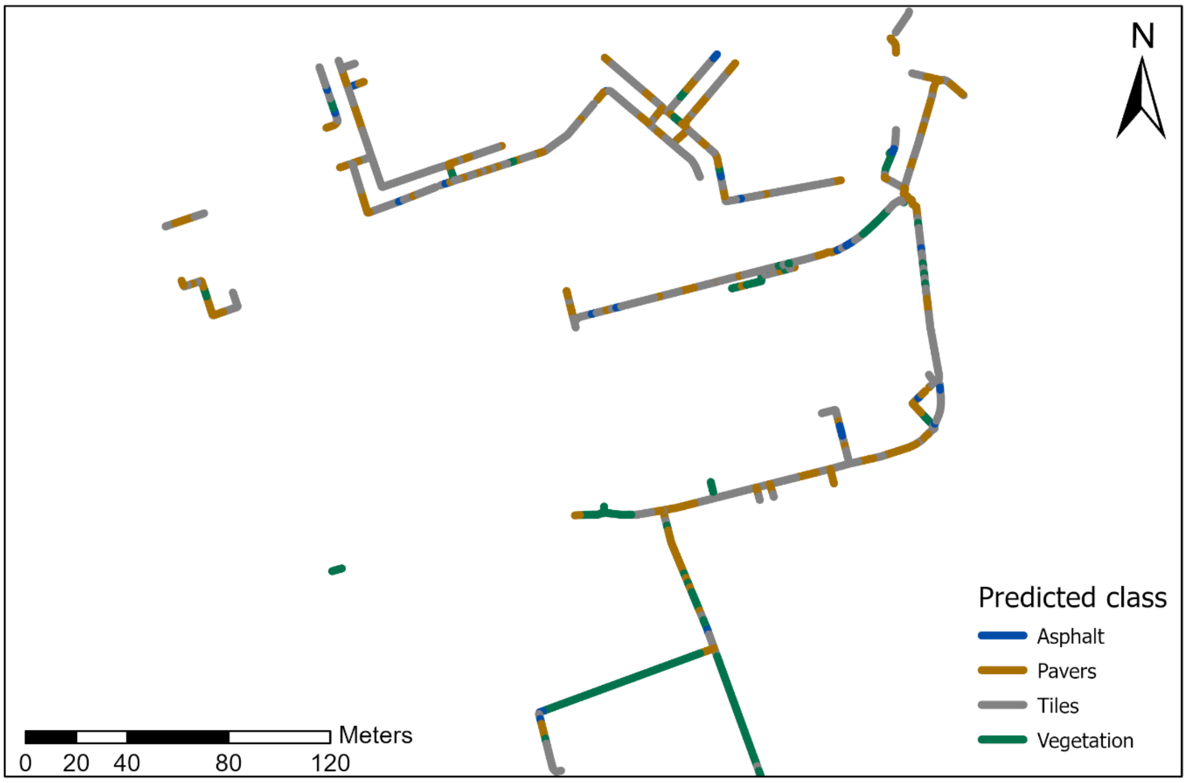

Figure D.2. Visualization of the predicted urban land cover of each cable segment by the non-augmented model.

E. Score maps

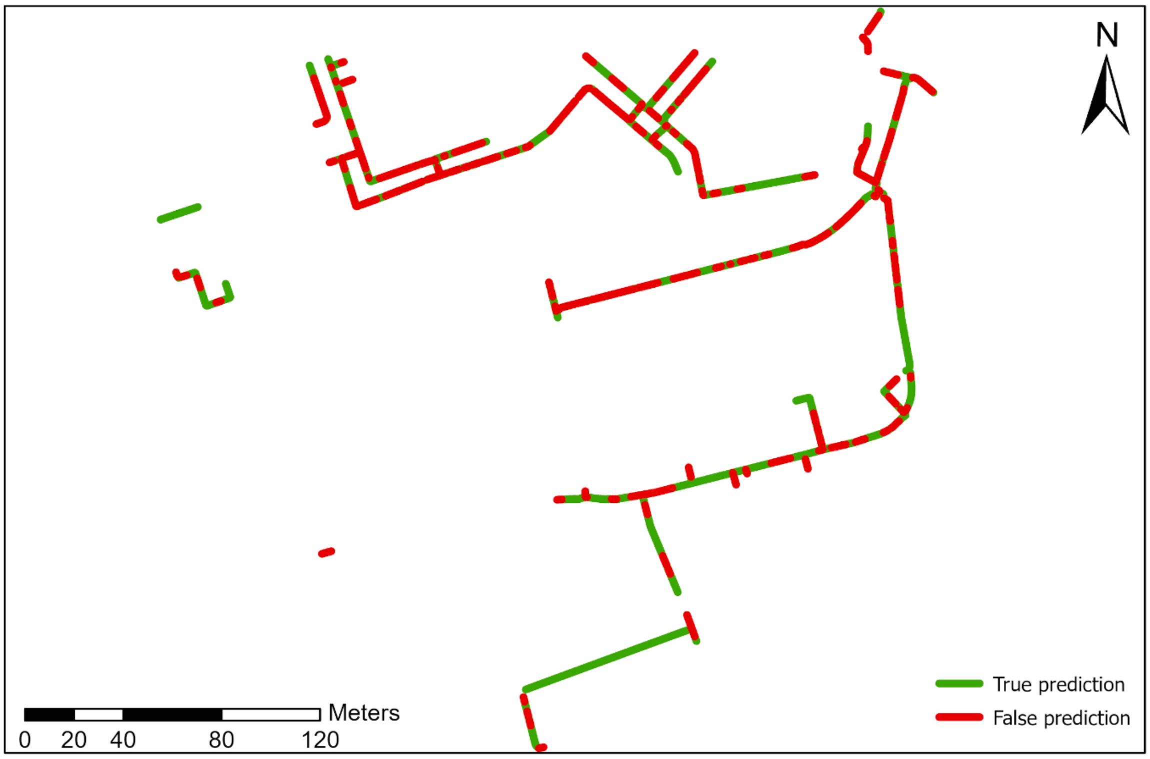

Figure E.1. Visualization of the true and false predictions of the urban land cover classification by the non-augmented model.

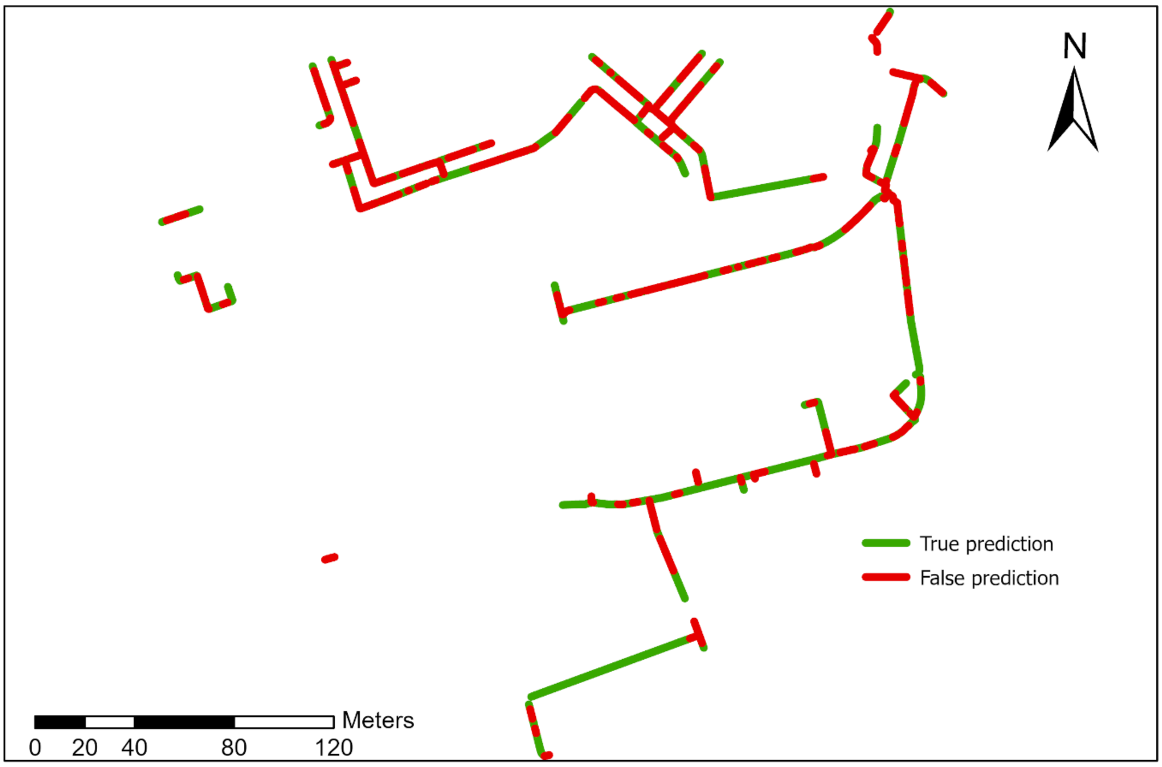

Figure E.2. Visualization of the true and false predictions of the urban land cover classification by the augmented model.