Detecting sidewalks in OpenStreetMap using machine learning

LaTeX Version

Master's Thesis in Computer Science

Eirik Gjeruldsen

June 5, 2020
Halden, Norway

Abstract

In the last couple of years there has been multiple machine learning projects focusing on detecting a variety of objects in OpenStreetMap. The explanation for this is the improvements done to both computer hardware and the increase of data in the OpenStreetMap database. As previous research done has primarily been done using image analysis and more traditional algorithms analysing spatial data in combination with tag data has been overlooked. This thesis aim to find the possibilities of using such a model to predict sidewalks, and study the differences between developed and developing countries. This is done by using two machine learning algorithms XGBoost and Neural Network in cooperation with features extracted from spatial data and tags contained within the OpenStreetMap database. The best performing model is then used to classify data from a developing country and exported to a custom made windows application to manually analyse images from the predictions. The experiments conducted indicates a connection between different inputs fed to the models and the ground truth, and produce a result with high accuracy. However, when classifying sidewalks on developing country data the image analysis show a big difference in results when testing on a country from Europe and a country from Africa. Despite the variation in results this thesis provides valuable input in regards to the limitations and constraints of the data in OpenStreetMap.

Keywords: XGBoost, Neural Network, Spatial data, Correlative features, Machine learning, OpenStreetMap, OSM

Acknowledgments

I want to thank my supervisor Kazi Shah Nawaz Ripon for supporting me throughout the work of this thesis and I want to thank Egde Consulting for giving me a great work environment while working on my thesis. I also want to give a special thanks to Andreas Taraldsen which has been my external supervisor throughout the thesis helping me out with technical discussions, every-other-week-meetings, and guidance.

Prerequisites

This thesis covers many different aspect surrounding computer science and machine learning, it is therefore not possible to go into every little detail about the subjects mentioned. It is assumed that the reader is already familiar with computer science terms as a third years student of computer science with a specialized field in machine learning and big data. The reader should have basic knowledge about tree-models and neural networks, and have a basic understand of advance mathematics.

Contents

Chapter 1

Introduction

OpenStreetMap (OSM) is an open source map service that is similar to the more familiar Google Maps[1] created by Google. The key element of OSM is that it is created by anyone who wants to contribute to the project, and its data is available to anyone who needs it. OSM have until this day utilized manual labour in the form of volunteers to generate the information they display on their services. The western world have had a lot of volunteers to help them generate a rich and valuable Geo-dataset while developing countries struggle with less volunteers resulting in a less refined map. This is done through a process of adding nodes containing Geo-location data through longitude and latitude, followed by adding connections between them through two different element types known as ways and relations.

In the past couple years there has been a rise in the amount of projects done with OSM data for the purpose of analysing it with machine learning. Machine learning is becoming increasingly more important for businesses world wide, and the vast amount of data that OSM holds become even more relevant than it has before. Especially due to the increase of the computational capacities of computer hardware, and the rise of a more data driven business style leading to the creation of larger and more accurate datasets. This makes it beneficial for the OpenStreetMap service to utilize machine learning to automate mapping procedures, and/or improve the efficiency for the volunteers. When looking at OSM's own wiki page they have a good overview of what projects are ongoing and have previously been done using machine learning[2]. Here there is a wide variety of projects that we will have a closer look at in Chapter 2.

Today most of the projects that are working on either improving or analysing OSM data has been done using advanced machine learning algorithms. These algorithms often has a high computational requirement, and can require large budgets to produce. To further expand the available methods at detecting objects in OSM this thesis will look to use more traditional machine algorithms with the purpose of classifying sidewalks by looking at nodes with the tag "highway" (Will be explained in Chapter 3). These nodes will contain information about their location in the

[1] https://www.google.com/maps

[2] https://wiki.openstreetmap.org/wiki/Machine_learning

world, what type of road it is connected to, and what objects are in close proximate to their location. The dataset created will consist of data from the European country Norway, and to analyse it with two machine learning algorithms: XGBoost, and Artificial Neural Network.

Furthermore, the thesis aims to answer three questions: 1) Is it possible to classify sidewalks using state-of-the-art algorithms on OpenStreetMap data? 2) Will there be a difference in using feature engineered groupings of data or using the actual value? 3) Testing the method on a pre trained model used on data from a developing country, how will it work? To do this multiple models will be created using both algorithms and the results will be compared. A process of feature engineering will be conducted to test how a grouping of numeric features will do compared to using the actual numeric value. Followed by a case study that will be conducted to see how the model respond to data from a developing country. A country from Africa will be chosen at random, and fed to the best performing model. The predictions will then be exported into a custom made windows application with the purpose of displaying the predictions on top of satellite imagery to allow manual analysis to be conducted.

1.1 Background and motivation

In the past couple of years the open-source map OpenStreetMap has seen an increase in popularity in regards to machine learning. Companies like Microsoft, Facebook, and renounced Universities have started to look at the possibilities using machine learning to improve OpenStreetMap, and researchers alike have started to look on the possibilities to help developing countries generate a free and well functioning mapping service.

The current use of manual mapping can be both time consuming and require individuals with local knowledge about the areas. The western world have multiple people contributing, which in return allows for a rich dataset with frequent updates. Furthermore, developing countries struggles to build their maps and have limited visual objects with almost non existing labels. A good example of this is the two images shown from OSM shown in figure 1.1. These images is a comparison between the city Arendal in Norway, Europe to the city Kidal in Mali, Africa. The European city shows a lot more information than its African counterpart where Arendal has information about bus stations, stores, restaurants, and parking lots something Kidal is missing.

Previous research shows us that creating machine learning models based on data from developed countries can be used on developing countries with some limitation, and indicates that the training data available in developing countries create problems that render the final results incomplete (Chapter 2). They do however show that when using a complete dataset on the same continent it can prove valuable and show good results. Looking further into cases with the use of machine learning one project is particularly exiting where Facebook has been working on mapping out

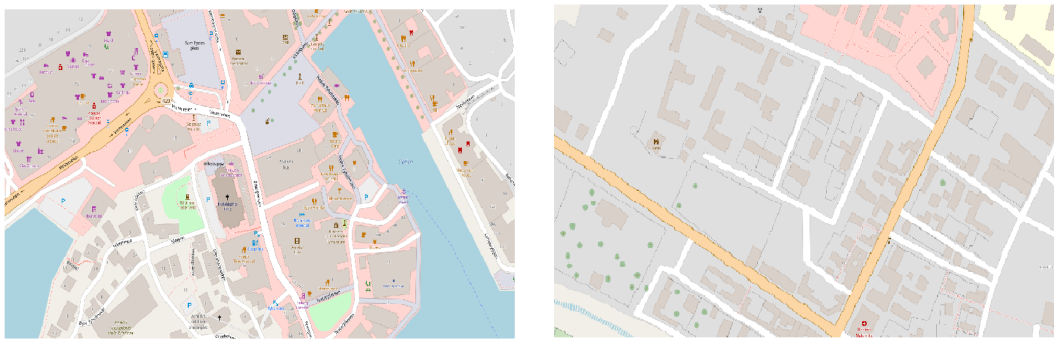

Figure (1.1) Comparison between Arendal, Europe (left) and Kidal, Africa (right) (OSM data).

roads based on aerial photos [3] (Chapter 2). This project have been going on since 2018 and have already completed a total mapping of Indonesia and Thailand. This shows that there is projects that have a great impact on OpenStreetMap. To help assist with this endeavor this thesis look at how different objects are linked and how they can help categorize if a road has a sidewalk next to it or not. Will the road itself as multiple features be enough to categorize this, or would you need a overview of surrounding objects as well? Considering this is a feature based analysis the computational requirements are drastically reduced compared to image analysis and if this method of detecting sidewalks prove reliable it could help OpenStreetMap fill in gaps on lesser objects and again increase the value of the dataset.

1.2 Research question and method

1.2.1 Research questions

As mentioned, exploring the correlation between objects could potentially help improve OpenStreetMap and allow automated processes to take away some of the requirements of manual work. By creating models based on two state-of-the-art algorithms XGBoost and Neural Network we can assess which algorithm should be used to prove that classification of sidewalk is possible. These two algorithms are two of the most renowned algorithms in a wide variety of competitions on a website called Kaggle[4]. This website is a known machine learning and data science community and has attracted many companies to post challenges to difficult problems in return for a price to the best performing contender. By seeing the frequent use of these two algorithms a hypothesis can be made that these algorithms have the possibility to provide good and accurate results. A more in-depth explanation of

[3] https://wiki.openstreetmap.org/wiki/Facebook_AI-Assisted_Road_Tracing
[4] https://www.kaggle.com/

the two algorithms will be given in Chapter 3. From these statements we look to answer the question:

Research Question 1: *Is it possible to classify roads that contain sidewalks using state-of-the-art algorithms in cooperation with OpenStreetMap data?*

Secondly, to determine how important certain features is to predict an accurate assessment of sidewalks an experiment will be conducted focusing on the differences produced by utilizing two methods we will call 'grouped features' and 'original features' (explained further in Chapter 3 and 4). From this statement we want to answer the question:

Research Question 2: *Which features are most important when creating a model based on 'grouped features' and 'original features'?*

Lastly, one of OpenStreetMap's biggest problems is the lack of data in developing countries. When creating model based on European country's data some of the data necessary to feed the trained model with a developing country's data might consist of missing values. To cope with that we want to answer the question:

Research Question 3: *Based on a case study conducted using a trained model on developing country data, is it possible to use a pre-trained model from a developed country to cope with the lack of data in a developing country?*

1.2.2 Research hypothesis

Considering the rules and regulations when building roads in Norway, it should be safe to assume that there are some correlation between objects and tags in the OpenStreetMap dataset making it possible for a machine learning algorithm to produce a model that can classify correctly. It is however uncertain to what extent this model will perform, and what algorithm will be the dominant one. Both algorithms are considered state-of-the-art and have the potential to outshine the other. Considering the generalization possibilities by having 'grouped features' (groupings of values) it is more than possible that it will outperform the 'actual value'. Lastly, looking at what previous researchers has done in the field of using machine learning and OpenStreetMap data it might prove difficult to get an accurate assessment of the developing country. It will how ever be possible to find the limitations with such and approach and categorize the possibilities for improvement.

1.2.3 Research Method

The literature review is done in a semi-systematic fashion where the main focus is to map out the entirety of previous and ongoing projects done with machine learning and OpenStreetMap data. Usually a semi-systematic is done purely through research papers, but OpenStreetMap has many companies contributing with machine learning projects without publishing papers. Something that makes them as

relevant as the research papers, and will therefore have its own section in chapter 2. Search-strings like "machine learning" AND "OpenStreetMap" was used to get a broad overview of the topic, it also shows there are few papers that use OpenStreetMap data in combination with machine learning having most papers being no older than five years.

In this thesis a correlational non-experimental research method will apply to find correlative patterns between objects to train a machine learning model to classify sidewalks. Two different machine learning algorithms are tested up against one another. To accurately assess this, the performance of the model will be represented through a quantitative analysis by comparing the results given by the confusion matrix calculations known as accuracy, precision, recall and f1-score. Lastly, a descriptive research method will be done through a case study where roads from the country Mali, Africa will be analysed based on the classification done by a pre-trained model.

Correctional research

Correlational research look to find the correlation between two or more things to find out what they have in common. In other words "correlational research examines the relationship between variables"[18]. It is worth noting that this research method does not imply that one variable causes change in another variable.

Descriptive research

The descriptive research method looks to describe the characteristics of and existing phenomenon. In other words 'It focuses on events that occur in the present[18]. By utilizing this method we aim to study the limitations of OpenStreetMap data in developing countries, and add suggestions to solutions based on the findings.

1.3 Report Outline

The content of the remaining thesis will look like this:

- Chapter 2 takes a look at what has been done previously with OSM and machine learning to see approaches done before, and find out what type of research is missing.
- Chapter 3 will be a discussion around the design and planning of the project.
- Chapter 4 looks at the implementation of the methods, and how workflow was conducted.
- Chapter 5 present the outcome of the research.
- Chapter 6 is a discussion surrounding the results/findings.
- Chapter 7 includes the final conclusion for the thesis and future work.

Chapter 2

Previous research

2.1 Research topic

This thesis looks to utilize traditional machine learning methods to detect a specified minor object in OpenStreetMap with the purpose of finding an efficient solution. The experiments performed will have a focus on detecting sidewalks, and to better understand the tag data in OpenStreetMap. Unlike most approaches today working with OpenStreetMap data having its core focus on image analysis which utilize satellite images and more computational heavy algorithms, this thesis aim to find a less computational restricted approach at detecting minor objects based on the spatial and tag data contained with in OpenStreetMap. Secondly, this thesis aim to explore the possibilities of using the models produced to help developing countries improve their data.

2.2 Related work

Sturrock [22] aim to predict residential structures using machine learning and remotely enumerated data from OSM. To better generalize their approach they create multiple models in parallel and combine the outputs instead of creating a single model. By doing this they aim to help developing countries who has little to no Geo-data to work with resulting in the possibility for different sectors to take advantage of this data to better improve the countries economy and welfare. The paper proves that machine learning can be used to tag houses, despite of having some limitations. There is a lot of computational time involved in such a project, and the results are only useful when every structure has been mapped, and as they discuss in their research: 'It is only possible to do classification when certain data requirements are in order'. It is also important to note that they tested their research on data from two different locations Swaziland and Botswana having different results in each location.

Grippa [9] focus on how essential it is to have up-to-date and reliable land-use information for a variety of reasons such as monitoring urban environment. Their aim is to develop a methodology for the creation of street block geometries using

very-high resolution satellite imagery and derived land-cover maps as input. By mapping the street block geometries by looking at OSM data and Silver Polygons[1] they managed to create a processing chain that would automate the creation of street block polygons.

Kaiser [12] worked with semantic segmentation of high-resolution aerial photos and have labels assign to each pixel. Their general motivation is more considering the big data aspect. The data available is to large for a human to process and therefore it would be beneficial to create machine learning methods to handle these kind of problems. To do this they utilize the supervised learning algorithm Convolutional Neural Network (CNN). Their results shown low accuracy, but they state that this is acceptable due to the volume of the training data, and that having multiple cities in their dataset gave better results.

Schultz [19] used VGI (Volunteered Geographical Information) tags to answer the problems they found with the LULC(Land Use Land Cover) data that was available. Here they looked at tags and how relations in OSM could be used to create LULCs. If an open source LC (Land Cover) product could cover the lack of data despite of the VGI's spatial incompleteness, and how accurate would this LC product be? To accomplish this they created a fractional Open Location Code using the VGI tags followed by filling in data gaps in the area they conducted their studies by using supervised non-parametric classification on remote sensing data. The authors describe their research as a proof of concept and showing to their results were several classes had achieved high accuracy. However, some classes had high errors due to the confusion between each other, and the site of the operation was in Heidelberg, southern Germany which makes the model applicable to Europe and it is uncertain how it would perform on another continent.

Chen [7] is looking into the use of machine learning to outsource volunteers using a software named MapSwipe. The fact is that most Geo-data labels today are mapped by volunters. With the use of machine learning this would no longer be the case, and could create a better way of getting the correct data not having to rely on a random samples created by people. To do this they used a CNN algorithm to analyse aerial photos. The author proves that their is a place for machine learning when trying to automate mapping functionalities. They concluded that using a combination of multiple VGI (Volunteered geographic information) would outperform one single VGI data, and combining them would help with technical solutions in regards to quality issues. The authors of this paper seems overly confident that they have solved the issue, but as a reader it seems like there is still more to explore.

Castagno [6] also look into the problem with having manually mapped data. This is why they are looking to propose a method for automatically label building roof shapes from publicly available GIS (Geographic Information System) data. The data in OSM is currently manually plotted which could lead to mistakes. This is why machine learning can help automate the process to better label roof tops. To achieve this they use satellite imagery and airborne LiDAR data by processing and manually labeling to create a diverse annotated roof image dataset for a small to large urban

[1] https://en.wikipedia.org/wiki/Sliver_polygon

city, followed by running the data through multiple CNN architectures to be trained and tested. Their results show that combining LiDAR data and satellite images works best together and provided a greater classification accuracy than using either data types alone. The models they produced show promise of being generalized and they believe that they could be applied when labeling roof shapes around the world. Considering this is a proof of concept paper it is uncertain that the last statement would hold up when working with a multi-continental setting, and will need further testing to confirm it.

Ibrahim [10] looks to identify slum in cities as current and future informal regions using machine learning and data from OSM. Informal regions within cities is a crucial issue for policymakers and governments in developing countries. Therefore finding good methods to find them is important to make the development of a country easier as the urban areas grow in population. They divided their modelling framework into two phases. The first one generating computed variables using Nearest Neighbours and Gi*. While the second phase consists of the model trying the formal and informal regions using an ANN (Artificial Neural Network) algorithm. They manage to create a modelling framework that can predict informal settlements, and by training the model on data from one city it allowed for positive results when predicting on another. The problem with the model is that it deals with a static state of a given time, while in reality the time is non-static. There is also some limitations with the model. The model can only look at a map up to 100 meter grid, and the validation relies on data on a well-identified informal settlement.

Jochem [11] also looks into identifying residential areas due to the expanding interest in remote sensing techniques in map and monitoring of urban land, and they noticed that previous researchers had not payed much attention to geospatial vector data. They try to identify neighbourhood types from settlement points using a machine learning approach by creating a model using the machine learning algorithm Random Forest and a 20m spatial resolution grid they prove that it is feasible to develop such a model, but they state that it is ahead of its time and the required data is not yet available for the public.

Ochoa [17] want to create an approach that fulfills the required tasks for agricultural resources. With the use of machine learning they can create an opportunity for researchers to perform faster and cheaper inspections on larger areas. To accomplish this they developed two CNN models, one for object localization and one to perform semantic segmentation that could mask the streets. They concluded that the approach required less feature engineering then other approaches while improving the overall performance. It also made it possible for faculty to gain a broader understanding of the patterns inside the data. Their approach required spending a long time on training, and saw the need for a lot of data to be able to create these models. They also mention the lack of publicly available data, which lead the researchers having to develop their own sets of images.

2.3 Machine learning projects in OpenStreetmap

There is a range of ongoing machine learning projects that are not published in any papers. Despite of this one can find an overview at OpenStreetMap's own wiki-page[2] containing multiple projects from a range of companies, the renounces being Facebook and Microsoft. Some of these projects are proprietary software and others are under an open-source license.

Firstly, we will have a look at the Open-source projects created with machine learning. The organisation Development Seed has created two projects completed during 2018 in OpenStreetMap. Looking glass a tool that can identify buildings in aerial imagery by using segmentation algorithms to label individual pixels. Mapping the Electric Grid, a tool to help users trace high-voltage infrastructures using machine learning. In 2018 Humanitarian OpenStreetMap Team worked on OSM Analytics Gap Analysis which look for potential gaps in the OpenStreetMap data by using machine learning predictions. Together these two organisations worked on a project in 2019 named ml-enabler with a goal to create a registry and facilitator for machine learning in the OpenStreetMap ecosystem. They also had collaberation with both Facebook and Microsoft and aimed: "to build AI-based mapping tools to empower mappers around the world."[3]. Here Facebook released RapiD in 2019 which is an extension of the OpenStreetMap browser editor iD and look to use AI-generated features to help the mapping process. At the same time as Development Seed created Looking glass the organisation Mapbox created a similar machine learning model specialising in feature extraction by using image segmentation on aerial and satellite imagery. While the other organisations looked at top down view, OpenStreetCam(telenav) looked at sign detection from street level imagery through their project Telenav-AI back in 2018. At the same time MIT CSAIL worked on a project with the name of RoadTracer which looked to create an automatic extraction of road networks from aerial images.

At OpenStreetmap's current state there are four proprietary software that are utilizing machine learning. The first to mention is AI-Assisted Road Tracing from back in 2016, which is Facebook's first publicly known project using machine learning in the OpenStreetMap environment. This project is still ongoing to this day and has been able to map multiple countries with more on the agenda. Another project that was done back in 2016 is Recognizing and Label Objects in the Wild by Mapillary. This project looked to do object and feature detection from street level imagery. Facebook is not the only big company taking part in the development in the OpenStreetMap environment. Microsoft detected building footprints in the United States using machine learning back in 2018. Lastly, Development Seed cooperated with Radiant Solutions the same year to create Urchn. A tool for urban change detection, creating opportunities to see how cities are changing and to help keep the map up-to-date.

[2]https://wiki.openstreetmap.org/wiki/Machine_learning
[3]https://www.hotosm.org/updates/the-machine-learning-enabler/

2.4 Summary

OpenStreetMap has been around for a long time, but the use of machine learning within the OpenStreetMap environment has not been seen until recently when considering the earliest recorded project on its own wiki-page is back in 2016. The amount of papers found during the literature search justifies the notion that this can be considered a new topic that has only been explored in recent years. There is a common denominator when developing the machine learning models. This is the use of image analysis, especially segmentation. This process is known to require a lot of computational power. Facebook shows that there is great possibilities with machine learning and image analysis, but research shows that there is computational constraints limiting researchers that has limited resources. This is why there should be a method utilizing more traditional machine learning approaches to generate data and allow researchers to have the possibility to use a less computational intensive method to further improve OpenStreetMap. There is also a wide discussion across the researchers that developing countries struggle. By accessing such a method more people can assist developing countries to enhance their maps and make way for a range of possibilities.

Chapter 3

Design plan

In this chapter we will go through a design plan that will be the base of our implementation of the different experiments. First a plan is made in regards to the data found in OpenStreetMap focusing on the extraction and understanding of the data. This is then followed by looking at approaches to generalize data and creating set rules when conducting feature engineering to generate features. Next is the machine learning algorithms XGBoost and Neural Network where reasons for choosing the algorithms and an in dept explanation will be given. Lastly, a plan will be made on how to perform the case study in regards to the developing country.

3.1 Data extraction

The first thing to do when starting the experiment of this project is to get a hold of the data available. OpenStreetMap provides an export of the users current view in their map services. This exports a filetype named .osm and provides the same syntax as a XML2 file. The structure of the file can be seen in Figure 3.1. These files are built up by three different type of elements[1]: **nodes**, **ways**, and **relations**.

Firstly, nodes are a defining point in space and includes the information regarding longitude, latitude, node id, and tag.

Node id *Every node in OpenStreetMap is unique, and to keep track of the individual node IDs are generated to keep track of them. It is important to note that there are overlaps with the ID when comparing nodes to ways and relations.*

Longitude *Is using the standard WGS84 projection and looking at the coordinates in degrees. It looks at an arbitrary value within the range greater than minus 180 and less than 180 with seven decimal places.*

Latitude *Is using the same WGS84 projection as longitude and looking at the coordinates in degrees. It looks at decimal numbers between greater than minus 90 and less than 90 with seven decimal places. It is important to note that some latitudes above/below 85 degrees might not be accepted for some projections.*

[1] https://wiki.openstreetmap.org/wiki/Elements

Tag *This is where the information regarding the node lies. It is a set of key/value pairs with an unique key, and is a representation of the physical features on the ground known as map features*[2].

Now that we know what a node is we can look at **way**. A **way** is a list of nodes that normally has at least one tag, and is included within a **relation**. They can contain anything from two to two-thousand nodes, but there could be outliers which contains zero nodes or a single node. There are two type of ways, an open way and a closed way. The closed way is where the start node and the end node is the same node. This can be interpreted as a closed polyline, an area, or both. The closed way is the opposite where the first and last node in a way is different.

Lastly, **relations** describes a group of elements. "To be more exact it is one of the core data elements that consists of one or more tags and also an ordered list of one or more nodes, ways and/or relations as members which is used to define logical or geographic relationships between other elements"[3]. This element is used to model logical or geographical relationships between objects. The guidelines when creating relations is that a relation should not include more than 300 members due to the technical limitations of the OpenStreetMap system. Different types of relations are but not limited to: multipolygons (one of two methods to register an area in OpenStreetMap), and bus routes.

The standard way of extracting data from the OSM database is by using the built in tool on the website. This is done through inserting world coordinates, that will create an overview of an area. To determine how much data will be extract you manipulate the zoom-value (the value in which decide how far away from the ground you are), but this has a certain constraints as you are not allowed to go beyond a certain threshold. This makes data extraction both time consuming and unreliable, and if we were to speed things up using automated extractions it might result in negative effect on the OSM-servers. Luckily there is a German based company named Geofabrik[4] which specialize in OpenStreetMap implementation for businesses that has already extracted the data, and made it public for anyone to use.

This file from the Geofabrik's website is encrypted to reduce the size of the file. Instead of using XML they use a more effective encryption tool named 'osmpbf'[5]. The decryption to access the data can be found on Github written in Go-lang, but also have newer additions in both C++ and Python. In this thesis Python libraries will be used to create machine learning models, that is why it would be prefered using the decryption tool in Python even though it has some limitations compared to the Go-Lang version when it comes to visualization of the data. Instead everything will be converted into a DataFrame using the Pandas library[6] which is using C++ to do the different operation returning a faster result than if the back-end was created

[2]https://wiki.openstreetmap.org/wiki/Map_Features
[3]https://wiki.openstreetmap.org/wiki/Relation
[4]http://www.geofabrik.de/
[5]https://wiki.openstreetmap.org/wiki/PBF_Format
[6]https://pandas.pydata.org/

```
<way id="392994566" visible="true" version="3" changeset="78742988" timestamp="2019-12-22T19:50:48Z" user="gormur" uid="103253">
 <nd ref="3966742000"/>
 <nd ref="3961804044"/>
 <nd ref="3961804041"/>
 <nd ref="7078241117"/>
 <tag k="highway" v="footway"/>
</way>
<way id="392994567" visible="true" version="2" changeset="78742988" timestamp="2019-12-22T19:50:48Z" user="gormur" uid="103253">
 <nd ref="7078241117"/>
 <nd ref="7078241116"/>
 <nd ref="3961804045"/>
 <nd ref="3961804046"/>
 <nd ref="3961804047"/>
 <nd ref="3961804035"/>
 <tag k="highway" v="service"/>
 <tag k="service" v="parking_aisle"/>
</way>
<way id="392994568" visible="true" version="1" changeset="36760762" timestamp="2016-01-23T16:12:20Z" user="NKA" uid="3497659">
 <nd ref="3961804048"/>
 <nd ref="3961804049"/>
 <tag k="highway" v="residential"/>
</way>
<way id="392994569" visible="true" version="1" changeset="36760762" timestamp="2016-01-23T16:12:20Z" user="NKA" uid="3497659">
 <nd ref="2612362067"/>
 <nd ref="3961804050"/>
 <nd ref="3961804051"/>
 <nd ref="3961804052"/>
 <tag k="highway" v="residential"/>
</way>
<way id="392994570" visible="true" version="1" changeset="36760762" timestamp="2016-01-23T16:12:20Z" user="NKA" uid="3497659">
 <nd ref="3961804053"/>
 <nd ref="3961804054"/>
 <nd ref="3961804055"/>
 <nd ref="3961804056"/>
 <tag k="highway" v="residential"/>
</way>
<way id="392994571" visible="true" version="2" changeset="41717971" timestamp="2016-08-26T15:55:37Z" user="NKA_import"
```

Figure (3.1) Example of how the data is shown in a .OSM file.

in Python. To cope with the visualization problem Matplotlib[7] will be used to plot the different data points when they need to be looked over. All files created will be saved as .csv file.

Now that we have established the data we can start making a plan to answer the research questions mentioned in Chapter 1. The correlation between features can somewhat be described in different relations, but they are currently used for single purposes or areas, and does not apply to the creation of new features. This creates an opportunity to find this out with machine learning. Accessing the longitude and latitude of road nodes, and then looking at surrounding features to categorize if there is a sidewalk next to the road or not. It will be plausible using this approach considering there are regulations in the western world in regards to where you have to place sidewalks. To do this we first have to create a dataset that can contain the true-positive and true-negative cases for the different road nodes then create features for each node taking the surrounding objects as input.

3.2 Generalizing data

The end goal of this thesis is to both to find the best solution to categorize sidewalk nodes and to see how the model will interact with developing country's data. Taking the last part into consideration the data has to be generalized and some limitations has to be set when extracting features from tags and surrounding objects. Firstly, the OpenStreetMap tagging system can be determine as a dictionary structure. A tag has a key and the value contained within that key is determined by whomever created the node in question. There are no rule-set for how to set the value, but the keys stay consistent throughout. Each node's tags contains multiple unique keys which are unique inside the their own tag dictionary. Keys can be used to access

[7]https://matplotlib.org/

certain features from a node to get a better understanding about it, and will be the main target for feature extraction. However, some keys might be specific for a certain country or area creating a localized bias if used. To make sure this do not happen, minor tags will be excluded and already generalized tags will be the main focus. As an example: if there are buildings around a road node, then this node will only keep track of amount of buildings around it instead of specifying what type of building. To find all available "generalized tags" a manual analyzes through the OpenStreetMap-iD editor will be done and findings will be cross-referenced with the tag-information in OpenStreetMap's wikipage[8].

3.2.1 Original features & Grouped features

After all features has been created and the original dataset is ready to be fed to a machine learning algorithm a second dataset will be created grouping all the numeric features produced by the feature engineering. From this two datasets will be ready to be analysed by the two different algorithms, these are as mentioned 'original features' containing exact count on numeric values and 'grouped features' a grouping of the numeric values. Example of this can be amount of nearby buildings. If we have three datapoints $a = 0, b = 5, c = 22, d = 23$ we could see how they would look in each dataset:

Original features *All datapoints would be the same as mentioned above.*

- $a = 0$
- $b = 5$
- $c = 22$
- $d = 23$

Grouped features *All datapoints will be grouped based on a predetermine scale (example a new group for every 10th building nearby.*

- $a = 1(grp1)$
- $b = 1(grp1)$
- $c = 3(grp3)$
- $d = 3(grp3)$

Both datasets will be used during the experiment to see which one yields the best result.

[8] $https://wiki.openstreetmap.org/wiki/Main_Page$

3.3 Machine learning

To better verify that this approach works, multiple machine learning algorithms will be used. This is to test state of the art algorithms and to verify the results of the different models.

To better verify our findings two state-of-the-art algorithms will be used to conduct this experiment, XGBoost and Neural Network. The last two algorithm we will be testing is what is considered state of the art algorithms, these are ANN (Artificial Neural Network) and XGBoost. XGBoost is more known in the Machine Learning community and is common to see topping lists on different Kaggle[9] competitions. ANN is the one algorithm that is most known for trying to mimic how human perceive data hence the name Neural Network. When building a large Neural Network it can be considered a deep neural network which is often talked about when doing state-of-the-art predictions. The premise of ANN is using layers of nodes connected by weights, the algorithm calculate these weights through training having a single output layer at the end to determine if the answer it returns is 0 or 1. These two algorithms are usually the two to perform the best at text based analysis, and should both be a part of this experiment to validate the method being used. Finally, by using the book 'Applied Predictive Modeling' by Max Kuhn and Kjell Johnson[13] and a articles on known data-science website as a base of reference we can take a better look at how the algorithms work.

3.3.1 XGBoost

The XGBoost is considered a gradient boosting tree. In terms of gradient boosting, the algorithm aims to reduce errors by using a gradient decent algorithm. What is unique for the XGBoost algorithm is that it has evolved from mere 'gradient boosting trees' to something called 'Extreme Gradient Boosting'[15]. It is known to be a combination of great optimization techniques utilizing a combination of both software and hardware. This can lead to better results with less computational time compared to other tree-based algorithms. Chen and Guestrin[8] describes the XGBoost model as a model that can return state of the art results and that it has a good reputation in regards to data-mining and machine learning challenges. XGBoost was used in every method in the top-10 during KDDCup 2015. "KDD Cup is the annual Data Mining and Knowledge Discovery competition organized by ACM Special Interest Group on Knowledge Discovery and Data Mining, the leading professional organization of data scientists." - KDD's offical website[1]. Not only does it perform well but is also known to be a lot faster than its competitors, Neural Networks and Deep Neural Networks. This is due to the optimization used: "a novel tree learning algorithm for handling sparse data; a theoretically justified weighted quantile sketch procedure enables handling instance weights in approximate tree learning."[8] This in combination with parallel computing makes it easier for the model to explore different options when training. Towardsdatasceince.com has an article that focus

[9]https://www.kaggle.com/

on explaining why XGBoost has become a state-of-the-art algorithm[15]. Here they bring up both system optimizations and algorithmic enhancements. The system optimization has three points: parallelization, tree pruning and hardware optimization. Parallelization is the parallelized implementation when building the sequential tree, and Morde specifies this is because of how the loops are used to build the tree structure. Secondly, tree pruning. XGBoost does a backwards pruning of its trees which improves computational performance. Lastly, the hardware optimization. The core of the algorithm is designed so that it utilize hardware resources efficiently by having cache awareness moving internal buffers in each thread to store gradient statistic. For the algorithmic enhancements there are four reasons why XGBoost perform as it does on an algorithmic level. Firstly there are regularization algorithms used to penalize more complex models using L1 and L2 algorithms. This will in particular help reduce the possibility of overfitting the model. Secondly, sparsity awareness. The algorithm allows a wide variety of inputs/sparse features by learning the best missing value depending on training loss. XGBoost also has a weighted quantile sketch as mentioned by both Morde[15] and Chen[8]. This means that the algorithm uses a algorithm that can effectively find the best possible split points among weighted datasets. Lastly, XGBoost has cross-validation as a part of its core infrastructure removing the need to use third party softwares to do these operations.

To be more precise on how the algorithm works, we can look at this mathematical function mentioned in Chen and Gestrin's paper [8].

$$\hat{y}_i = \varnothing(x_i) = \sum_{k=1}^{K} f_k(x_i), \quad f_k \in \mathcal{F}$$

F indicates the decision tree algorithm: CART, and the K represents an additive function to predict the output. f_k is shown for an independent tree structure represented as q, and w illustrating the leaf weights. This can be summed up through the mathematical function[8]:

$$f(x) = \{w_{q(x)}\}(q : \mathbb{R}^m \to T, w \in \mathbb{R})$$

To calculate and learn the functions used in the model we can reduce the regularized objective by doing this:

$$\mathcal{L}(\phi) = \sum_i l(\hat{y}_i, y_i) + \sum_k \Omega(f_k), \quad f_k \in \mathcal{F}$$
$$\textbf{where } \Omega(f) = \lambda ||w||^2$$

The 'l' after the first $\sum$ is a differential convex loss function which tests the variation between '$\hat{y}_i$' and y_i. This helps to smooth out the final weights to avoid over-fitting.

3.3.2 Artificial Neural Network

Neural network is an algorithm designed to mimic how the brain works by creating nodes that are connected to one another. The neural network model consists of three different layers. These are *input layer*, *hidden layer*, and *output layer* (See Figure 3.2). Input layer indicates the input that you feed the model, in other words the data that the model will be given. The hidden layer is where the neural network does its calculations to help decide what the final result is. Each node in the hidden layer has a transformation function that will manipulate the data input. In figure 3.2 there are arrows in between each node which represents the weights between the nodes and determines how important that node is to the final result. Lastly, we have the output layer which makes the final decision.

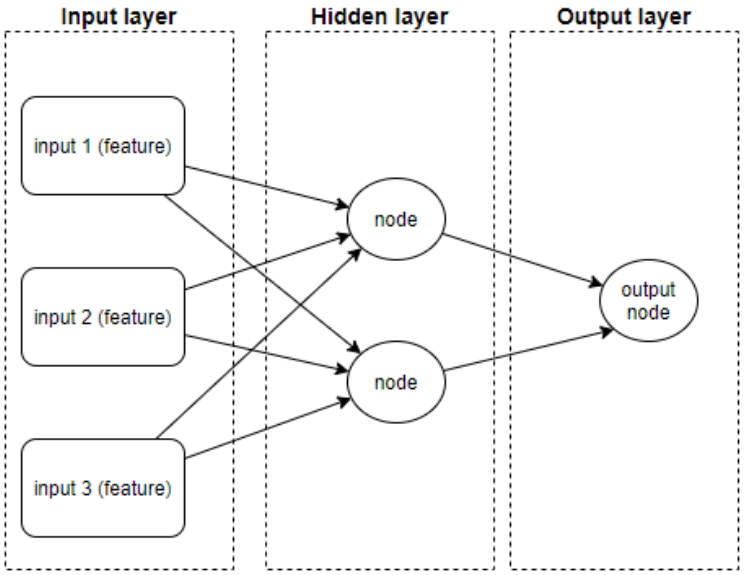

Figure (3.2) Example of a Neural network with 1 hidden layer.

When training a neural network the model aims to modify all the weights to reach the highest possible output depending on the loss function specified when making it. What is unique for the neural network compared to the other models is that the hidden layers are sort of a black box and it would be really hard for a person to analyse the inner working of the trained model. You can however return the weights and look at them, but depending on how large your neural network is this can prove problematic. In the structure of the neural network you can include multiple hidden layers, and each layer can contain a large amount of nodes inside it. When having multiple layers with large amount of nodes then the model is not just a neural network but can be specified as a deep neural network most commonly known as deep learning. As Monge[16] mentions in his article even though deep learning models are more complex, they do not necessary require a lot more computational power and large datasets. This make both normal neural networks and deep neural networks contenders for this experiment.

When building a neural network it is important to understand two concepts: global minima and local minima. To understand this we first have to talk about something called a loss function. A loss function is simply a function that calculate the error of the model. With neural networks we aim to train the model to minimize the total amount of loss[2]. An example of a loss function can be cross-entropy:

$$\sum_{l=1}^{C}\sum_{i=1}^{n} y_{il} \ln f_{il}^{*}(x)$$

Global minima represents the optimal solution based on the the results of the loss function in the entire domain of the function, while the local minima is the optimal solution based on the results in a given range (See figure **??**. In an ideal scenario when creating a neural network you would look to reach the global minima which would be the best possible solution, but more often than not the model produced will stop in the local minima due to a variable in the function called learning rate. This variable decides the magnitude that the model can tune its parameters during training to reduce the result from the loss function. A common problem is having a small learning rate where you reach something called a vanishing gradient, which indicates a point where the model iterates without changing its weights. Another problem would be to keep the learning rate to high as the model could potentially fluctuate and never tuning the parameters so that it would reach the local or global minima.

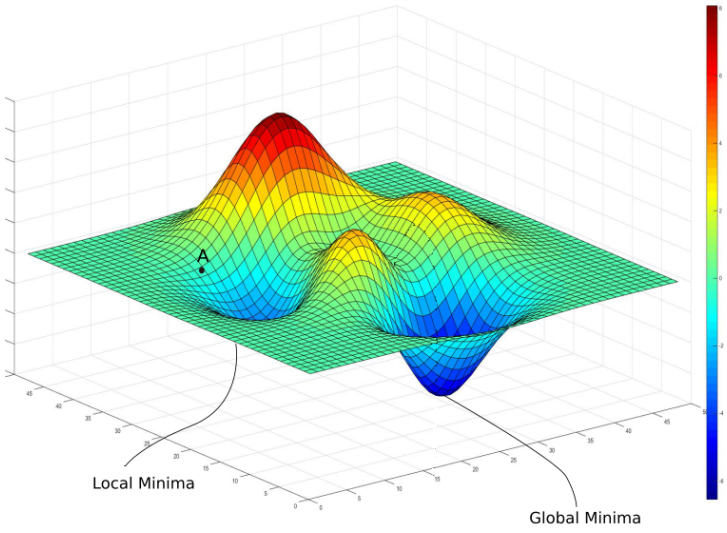

Figure (3.3) Example of global and local minima. (Image taken from blog.paperspace.com)

There are different ways to tackling this problem with learning rate. You could use regularization algorithms or something called dropout which simply reduce the learning rate for each iteration. It is important to note that compared to other

algorithms neural networks have a lot of different tuning parameters, and to get a complete overview of how the algorithm perform it would be necessary to do something called hyper-parameter-tuning.

3.3.3 Grid-search and Hyper-parameter-tuning

When creating models there are parameters that has to be changed depending on the data, to do this model tuning is done to find the best parameters for the given model and data. [14]. To do find the best performing model parameters we will use a Scikit-learn tool named GridSearchCV[3] which will be referenced as 'grid search' going forward. This search will allow you to do an automated procedure that can fine tune the model parameters in the give parameter space. You will still have to define the parameters you want to try out, but it will return the best combination of the given space. This will be an iterative process where a trial-and-error method will be utilized trying to narrow down the search space. When doing a grid search on the XGBoost algorithm we aim to narrow down the search space until the model return the same combination of parameters after two consecutive iterations. The starting search space will be picked at random and worked from using a trial-and-error approach.

Secondly, the Neural Network will be using a RandomizedSearchCV[4] instead of a grid search for automating the fine tuning of the parameters. Random search will select a random amount of combination to try to find the best combination from the given parameters search space. Firstly, a search will be done using 2, 3 and 4 layers in combination with a variance in batch size, epochs and dropout to see which parameters yields the best results. After that is complete a second random search will be done doing a more narrow search increasing the amount of iterations, and tune the model on more similar parameter variables. As the XGBoost tuning a trial-and-error method will be used, having some manual tuning in combination with the automated version. Throughout all iterations the activation function ReLu (rectified linear unit):

$$f(x) = x^+ = \max(0, x),$$

will be used for the hidden layers, and the sigmoid function:

$$S(x) = \frac{1}{1 + e^{-x}} = \frac{e^x}{e^x + 1}$$

will be used for the output layer. The model will compile by using an adam optimizer which is an updated version to the RMSprop optimizer:

$$\begin{gathered} m_w^{(t+1)} \leftarrow \beta_1 m_w^{(t)} + (1 - \beta_1)\nabla_w L^{(t)} \\ v_w^{(t+1)} \leftarrow \beta_2 v_w^{(t)} + (1 - \beta_2)(\nabla_w L^{(t)})^2 \\ v_w^{(t+1)} \leftarrow \beta_2 v_w^{(t)} + (1 - \beta_2)(\nabla_w L^{(t)})^2 \\ \hat{m}_w = \frac{m_w^{(t+1)}}{1 - \beta_1^{t+1}} \end{gathered}$$

$$\hat{m}_w = \frac{m_w^{(t+1)}}{1-\beta_1^{t+1}}$$
$$\hat{v}_w = \frac{v_w^{(t+1)}}{1-\beta_2^{t+1}}$$
$$\hat{v}_w = \frac{v_w^{(t+1)}}{1-\beta_2^{t+1}}$$
$$w^{(t+1)} \leftarrow w^{(t)} - \eta \frac{\hat{m}_w}{\sqrt{\hat{v}_w} + \epsilon}$$
$$w^{(t+1)} \leftarrow w^{(t)} - \eta \frac{\hat{m}_w}{\sqrt{\hat{v}_w} + \epsilon}$$

Optimizers used in making neural networks derives from the Stochastic gradient descent method. The main goal of optimizers is to reduce the computational requirements when training the model. In combination with the adam optimizer binary cross entropy will be used to calculate loss. Due to the network being a feed forward neural network a backpropagation function is required hence the use of binary cross entropy to compute the gradient of the loss function.

3.3.4 Validating the results

For the result returned by the models we use something called 'cross validation'[21]. In simple terms this means that we do a x amount of tests on the same trained model and return the mean of the result to reduce variance in the different results. In this experiment 5-fold cross validation will be utilized, in other words it will test the score of the trained model five times before accessing the result of the trained model.

To follow normal research practice the dataset will be divided into a training dataset and a test dataset. Here a 80/20 split will be done using sklearn's train_test_split to divide the data at random. Another 80/20 split will be done on the training data dividing the data into training and validation data. The training data is what the model will be given during training, validation will be used to validate the model during training, and test will be a one time use to check the final result of the model once training is complete.

Lastly, the model will be evaluated based on an accuracy matrix. Looking at figure 3.4 we can use this formula to calculate the accuracy:

$$accuracy = (TP + TN)/(TP + TN + FP + FN)$$

The benefits of using accuracy is that it looks at the performance of all predictions made while other functions will look at specific performances. In this experiment we aim to find how well the model performs on both positives and negatives, hence the use of accuracy. For the final analysis all metric calculations will be added: accuracy, recall, precision, f1-score, and the confusion matrix providing TruePositive (TP), FalsePositive (FP), FalseNegative (FN), and TrueNegative (TN).

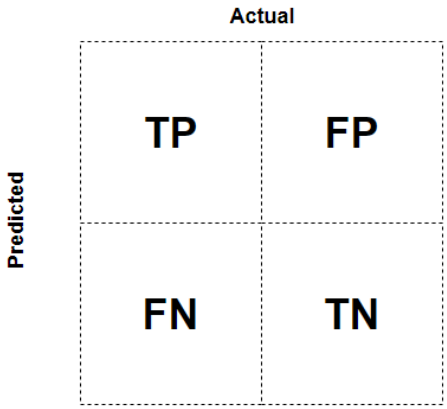

Figure (3.4) Visual representation of an evaluation matrix.

3.3.5 Case study

The case study will be conducted by using the best performing model from XGBoost and Neural Network followed by manually analysing the models prediction.

While feeding the model the data from a developing country it will be important to keep track of which features are missing when extracting them from the OpenStreetMap database. Some tags or features might not exists in these countries and might render the experiment incomplete because of it. This is why training new models might be necessary if there some features missing during extraction. After a final prediction is complete images are extracted using a custom made windows application having each node represented as a point on a satellite imagery allowing for the extraction of an image containing a representation of the prediction and its surrounding environment.

Chapter 4

Implementation

In this chapter the implementation of the design plan will be created and problems surrounding the implementations will be discussed, as well as altering solutions when the original design plan proves incomplete. Firstly we will go through the setup, data and feature engineering done for the data used for the experiments. This section also includes general information in regards to hardware and software used when conducting the experiments. Followed by the implementation and creation of different machine learning models in the section Machine learning. Lastly we will look at how the case study was implemented and challenges that arose during the implementation.

4.1 Setup, data and feature engineering

For this project a conda environment is created to have a work place. The environment is installed and kept up-to-date on a local computer. All time based decision are based on the computational power in the list below:

- Graphic card: MSI GeForce RTX 2080 Duke OC V1, 8GB GDDR6, 1515/1800MHz
- Processor: AMD Ryzen 7 2700X, 8-Core, 16-threads, 4.35 GHz, 20 MB cache
- Memory(RAM): 2x DDR4 2666MHz 8GB

The project is done on a Windows 10 (Home) computer using a variety of softwares and packages that is listed in Figure 4.1. A mix of programming languages, and tools have been selected to reduce potential development time. As mentioned in chapter 3 the data from Geofabrik was utilized as the base of the dataset. To access the data you had to decode it from a .pbf format. This was done using a combination of a foreach loop in Python and the esy.osm.pbf library. This library allows you to read a .pbf file and show it as layers of dictionaries. This will allow for iterative processes where you can use a foreach loop to get each datapoint. This datapoint contains different sub-dictionaries depending on what type of class it is bound too. The sub-dictionaries can be seen in the list below, and has been mentioned previously in chapter 3.

- 'esy.osm.pbf.file.Node'
- 'esy.osm.pbf.file.Way'
- 'esy.osm.pbf.file.Relation'

To get a better overview of the difference between way and relation each of the three classes was separated into different CSV files to create an easier way to understand and process the different datapoints. To bring out the features of each datapoint a manual inspection of each property was done by extracting a datapoint with the different class properties. The result of this returns these sub-dictionaries for each class:

1. Node
   (a) id (node-id, unique for each node)
   (b) tags (information regarding the node, can be null)
   (c) lonlat (world coordinate of the node)
2. Way
   (a) id (way-id, unique for each way)
   (b) tags (information regarding the way)
   (c) refs (reference to all nodes contained within the way)
3. Relation
   (a) id (way-id, unique for each relation)
   (b) tags (information regarding the relation)
   (c) member (objects contained within the relation)

When dividing the data into their designated classes it is important to note that the amount of data in the file consists of over 100 million datapoints, and when adding everything into your memory it will create problems. To fix this a iterative process exporting a csv file containing 3% of the total amount of nodes to both relieve the memory of overloading and create smaller csv files which was both easier to read and load. These files were kept in their own folder to import them by using the pandas library and inserting them into a DataFrame using an iterative process combining two libraries: pandas and glob. They iterate through each file adding them into a combined DataFrame. (See Listing 4.2). The benefits of keeping everything in pandas.DataFrame is the possibility to use pandas powerful search functionality which is many times faster than doing loops within Python itself, while still allowing everything to be accessed in the same software/programming language. Additionally, pandas DataFrames is one of the most common way to feed information into the machine learning libraries available and by keeping the standard throughout the process it could potentially save time. It is worth nothing

Software/package
Python 3.6
Keras
Visual Studio
CSVHelper
Sklearn
Pandas
Jupyter notebook
OpenStreetMap.com
OpenStreetMap iD-editor
esy.osm.pbf
glob
Numpy

Table (4.1) The different packages and software used in this project.

that the pbf-reader is simply reading directly from a file and do not insert it into memory unless you have to manipulate the data. The code snip below show how a simple extraction of the data was done using the pbf library having the 'osm' variable holding a direct link to the pbf file (Listing 4.1).

Listing (4.1) Simple pbf extraction of the node class.

```
for entry in osm:
    if type(entry) == types[0]:
        typeNode.append("node")
        ids.append(entry.id)
        tags.append(entry.tags)

        lon.append(entry.lonlat[0])
        lat.append(entry.lonlat[1])
```

Listing (4.2) Importing multiple CSV-files in Python using Pandas-library.

```
import pandas as pd
import glob

path = 'Nodes/'
all_files = glob.glob(path + "/*.csv")

li = []

for filename in all_files:
```

```
    df = pd.read_csv(filename, index_col=None, header=0)
    li.append(df)
    print(filename, " was read")

frame = pd.concat(li,axis=0, ignore_index=True
```

On further inspection the relevant object are nodes and ways, as we want to look at how single objectives will affect the prediction of sidewalks and ways to create the ground truth for our dataset. The first thing done was to import all ways into a DataFrame and look through both the OpenStreetMap wiki page and navigating around the different tags found in the iD editior. The tag that seem to be the determining factor for a sidewalk or not was the tag-key 'foot' with the value of 'yes'. Using pandas syntax the search itself went quick using this code snip:

Listing (4.3) Check for Ground Truth.

```
footYes = df.loc[:][df.tags.str.contains("'foot': 'yes'")]
```

The dataset returned as 'footYes' was now considered the positives of the roads. The remaining ways containing the tag 'highway' will therefore be categorized as negatives. After further inspection there was also a value name 'highway' : 'footway' which was also added to the positive side of ground truth

Feature extraction from tags

Having so many tags to chose from it would be important to convert them to features to allow the algorithms to get more insight to how what defines that said way. Again another iteration of searching through OpenStreetMap wiki page, and looking through the iD-editor was done to see what features could be brought out without de-globalizing the dataset. To keep it globalized only two tags were extracted: "maxspeed" which indicated speed limit shown in the code-snip below:

Listing (4.4) Extracting 'maxspeed' from tags.

```
maxSpeed = list()

for i in range(len(completeDf)):
    tagsInDf = eval(completeDf.iloc[i].tags)
    if 'maxspeed' in tagsInDf.keys():
        maxSpeed.append(tagsInDf['maxspeed'])
    else:
        maxSpeed.append(-1)
```

And what type of road it was based on the predefined rules by the creators of OpenStreetMap that had the original key being 'highway' which was used to extract the correct ways from the original dataset. Highway tags [1] was more difficult than maxspeed as there were many different types of highways, and the spread was between many different tags.

[1] https://wiki.openstreetmap.org/wiki/Key:highway

Tag-name	Count
Trunk	43165
Residential	6452
Secondary	5640
Unclassified	3714
Tertiary	2047
Service	1739
Cycleway	1178
Trunk_link	6452
Motorway	450
Track	397
Living_street	369
Primary_link	305
Footway	272
Path	81
Unknown	77
Motorway_link	68
Construction	17
Road	14
Pedestrian	6

To cope with the wide variety of highway categories, five groups were made using a simple integer to track each of the groups ranging from 0-3, and having the last being -1 which indicated that a datapoint should be deleted.

Listing (4.5) Grouping the different highway tags.

```
def ReturnFeature(string):
    if string == "trunk" or string == "motorway_link":
        return 0
    elif string == "primary" or string == "secondary"
    or string == "tertiary" or string == "track" or string == "path":
        return 1
    elif string == "residential" or string == "service"
    or string == "pedestrian" or string == "living_street":
        return 2
    elif string == "cycleway" or string == "road":
        return 3
    else:
        return -1
```

Group name	Description
0	A highway that has divided lanes from ongoing traffic.
1	A general highway that is normally found between towns, but could also be main roads cross country. Can be considered a highway without physical division of ongoing traffic
2	Roads that are in between houses or are used for specific purposes, these are roads that are considered small roads in an urban area
3	These are general roads with little to non extra information other than that this is a road with limited rules bound to them.
-1	Tags that leads to the node being deleted. Can be any type of road that is hard to categorize, and therefore can lead to results that are misleading when analysing with machine learning. Example: construction of a road.

The original plan was to use the distance formula[2] to find the nearby objects.
$d = \sqrt{(x2 - x1)^2 + (y2 - y1)^2}$

This formula returns the distance between two points/Vector2(x,y). After doing a quick implementation in Python it became apparent that using multiple math operations on a not specified variable type increased the computational time by a lot, especially when looking at nearby buildings where the math operations necessary to complete was more than a hundred million iterations. In Python an iteration could take a couple of seconds, and the time estimate for finishing would be above four hundred days of computation. To deal with this, multiple CSV files were exported from the Python project. One containing the main-dataset, and four more csv files consisting nodes exported with a specific globalized label. These four labels where:

Buildings *A node that has a designated tag named 'addr' (address)*[3].

Leisure *A node related to certain venues. This could be arcades, beach resorts, cinemas, etc.*[4]

Land Use *land Use reference a certain type of land. This can be anything from commercial, construction and industrial ground to forest, parks, and colleges.*[5].

Amenities *An object that is an amenity: toilets, EL-charging stations for cars, gas-pumps, etc.*[6]

[2]https://mathblog.com/reference/arithmetic/distance-two-points/
[3]https://wiki.openstreetmap.org/wiki/Key:addr
[4]https://wiki.openstreetmap.org/wiki/Key:leisure
[5]https://wiki.openstreetmap.org/wiki/Key:landuse
[6]https://wiki.openstreetmap.org/wiki/Key:amenity

roads.png roads.png

Figure (4.1) Examples on how the different roads are divided.

The first though was to import it all into a C++ project by building the project with simple classes and running it through a console based software in main.cpp. C++ is known to be quit sufficient when it comes to mathematical operations, but the limitations found was in regards to reading files. When looking at library implementation for reading CSV-files it could take a lot of extra time just to read the CSV and write the results after completion because there were not a unified way of reading and writing all types of CSV. Instead the focus was switched towards C# using a community version of Visual Studio. Here you had available 'NuGet' (libraries in Visual Studio) that could help with the importing and writing of CSV files, hence the NuGet CSVHelper was installed into a simple console project. With the help of CSVHelper reading and writing became an easy task by only having to use a couple of lines of code.

Listing (4.6) Syntax for reading/writing CSV using the C# NuGet named CSVhelper

```
using (var reader = new StreamReader(path))
```

```
{
    using (var csv =
    new CsvReader(reader, CultureInfo.InvariantCulture))
    {
        csv.Read();
        csv.ReadHeader();
        while (csv.Read())
        {
            mainDf.Add(new DataPoint(csv.GetField<string>("nodeId"),
            csv.GetField<double>("lon"),
            csv.GetField<double>("lat")));
        }
    }
}

using (var writer = new StreamWriter("..\\AmenityCount.csv"))
using (var csv = new CsvWriter(writer, CultureInfo.InvariantCulture))
{
    csv.WriteRecords(amenity);
}
```

The idea of the export was to get the amount of an object in the surrounding area, as specified tags could potentially be geographical based while the four labels mentioned above would be generalized and can be found all across the world. Having filtered tags in Python previously created an easy workflow when doing the math operations in C# creating a minimal amount of labeling.

To create the new features a counter was made for each label, and each node would iterate through the list of objects using the distance formula mentioned above and adding itself as a neighbouring object if the distance were less or equal to the value chosen. This created an iterative process to see how the distance was going to be considered.After doing a couple of iterations one thing became apparent; to avoid having a lot of similar data-entries a certain distance was required. Looking at decimal degrees[7] 0.01 appeared to be the best option due to it still allowing zero cases, while also filling in different amount of data into the dataset. The 0.01 was said to represent the size of a town or village, and was therefore chosen. Code implementation can be seen below:

Listing (4.7) C# implementation of the distance forumla

```
public bool CalculateDistance(double x1, double y1,
                              double x2, double y2)
{
    return Math.Sqrt(Math.Pow(x2 - x1, 2)
    + Math.Pow(y2 - y1, 2)) <= 0.01f;
}
```

Listing (4.8) How the features are created in C#.

[7] https://en.wikipedia.org/wiki/Decimal_degrees

```
public struct DataPoint
{
    public string nodeId;
    public double lon;
    public double lat;

    public DataPoint(string idIn, double lonIn, double latIn)
    {
        nodeId = idIn;
        lon = lonIn;
        lat = latIn;
    }
}


public List<int> CreateFeatureCount(string featureFileName)
{
    ReadCSV("..\\Dataset_ReadyForNaboringFeatures.csv", true);
    ReadCSV("..\\" + featureFileName + ".csv", false);

    List<int> countOfListFeatures = new List<int>();

    for (int i = 0; i < mainDf.Count; i++)
    {
        double x1 = mainDf[i].lon;
        double y1 = mainDf[i].lat;
        int count = 0;
        for (int j = 0; j < featureDf.Count; j++)
        {
            double x2 = featureDf[j].lon;
            double y2 = featureDf[j].lat;

            if(CalculateDistance(x1,y1,x2,y2))
            {
                count++;
            }
        }
        countOfListFeatures.Add(count);

    }
    return countOfListFeatures;
}
```

The check consists of a double for-loop which is also a reason it took a long time to process using Python. By using C# computational time requirements for these operations went down to seven/eight hours using the specs mentioned at the beginning of this chapter.

The final implementation in main looked like this:

Listing (4.9) Main() function in the C# implementation.

```
static void Main(string[] args)
```

```
{
    Functionality functions = new Functionality();
    //LAND USE
    List<int> LandUse = functions.CreateFeatureCount("LandUse");
    using (var writer = new StreamWriter("..\\LandUseCount.csv"))
    using (var csv = new CsvWriter(writer, CultureInfo.InvariantCulture))
    {
        csv.WriteRecords(LandUse);
    }
    // LEISURES
    List<int> Leisures = functions.CreateFeatureCount("Leisures");
    using (var writer = new StreamWriter("..\\LeisuresCount.csv"))
    using (var csv = new CsvWriter(writer, CultureInfo.InvariantCulture))
    {
        csv.WriteRecords(Leisures);
    }
    // AMENITIES
    List<int> amenity = functions.CreateFeatureCount("amenity");
    using (var writer = new StreamWriter("..\\AmenityCount.csv"))
    {
     csv.WriteRecords(amenity);
    }
    // Adress / buildings
    List<int> houses = functions.CreateFeatureCount("Houses");

    using (var writer = new StreamWriter("..\\housesCount.csv"))
    using (var csv = new CsvWriter(writer, CultureInfo.InvariantCulture))
    {
     csv.WriteRecords(houses);
    }
}
```

Lastly, some of the features had to be removed due to each individual point containing data that is specific to only that given point and could quickly train the model to look at something else than it is suppose to. These features are: nodeId, wayId, longitude, and latitude. The first two are removed because they are only used to identify the nodes in OpenStreetMap, and for the first part of this experiment practical implementation does not have a focus. Instead each datapoint is given a random index from zero to max length of the dataset to make it possible to get the nodeId and wayId at a later point in time. The lon/lat coordinates are specific to each datapoint, and having a model trained on coordinates makes it less globalized and more specified to a single purpose something that is important to avoid when working with machine learning. To clarify this a test was done using XGBoost and looking at the feature_importance map exported from the trained model, and will be discussed further in 6.

4.2 Machine learning

In this section a detailed description of how the implementation was done after the initial dataset was ready to be used by machine learning algorithms.

4.2.1 Division and preparation of data

On first attempt at implementing the machine learning algorithms it was apparent that some preprocessing had been forgotten, considering the rule of only using generalized tags and only using the key value of the tag to produce features was overruled with one feature named 'maxspeed'. This feature was thought to be widely important as when creating road systems this will be a factoring value when deciding to put a sidewalk or not. The problem that occurred was that there had been some typed text-based information in the tag, when original thought to only be numeric. The text that was inserted was: 'RU:rural' which indicates it is an undefined speed limit. To deal with this the datapoints with "$maxspeed == RU : rural$" would be replaced with a 0 speed limit as this was a speed limit not contained within the dataset already and would indicate 'no speed limit' for future reference. It is also important to note when creating such a numeric value it requires the value to be grouped separately in its own unique group when converting to grouped features.

Listing (4.10) Changing speedLimit outliers.

```
maxSpeed = df['maxSpeed']

for i in range(len(maxSpeed)):
    if maxSpeed[i] == 'RU:rural':
        maxSpeed[i] = 0

df['maxSpeed'] = maxSpeed
df['maxSpeed'] = pd.to_numeric(df['maxSpeed'])
```

4.2.2 Grouping of features

To better create good grouping for each feature, all features with numeric value which is not already part of a group was manually analysed using a simple for-loop in Python.

Listing (4.11) Example to check different groups.

```
maxspeed = df['maxSpeed']
output = list()
for x in maxSpeed:
    if x not in output:
        print(x)
        output.append(x)

#This loop returned:

90 80 60 70 50 30 40 100 10 0 20 5 15 6 8 110
```

Using the code snip above as an example the groupings would have to follow two criterias: 1. There should not be an abundance of groups, meaning each group

should reflect a propose and/or a valid reason based on the unique variables listed. 2. The groupings should be generalized and should be able to be created on a different countries dataset.

Most of the groupings where based on finding a good split based on the highest value available in the dataset, while others like the example above for speed-limit was based on what restrictions these speed-limits brings. The grouping was done using a simple function returning a group based on the input value. See function below:

Listing (4.12) Function to divide into different groups.

```
def GetSpeedGroup(currentSpeed):
    if currentSpeed == 0:
        return 1
    elif currentSpeed > 0 and currentSpeed <= 20:
        return 2
    elif currentSpeed > 20 and currentSpeed <= 40:
        return 3
    elif currentSpeed > 40 and currentSpeed <= 60:
        return 4
    elif currentSpeed > 60 and currentSpeed <= 80:
        return 5
    elif currentSpeed > 80:
        return 6
    else:
        return 0

maxspeed = df['maxSpeed']

new_maxspeed = list()
for x in maxspeed:
    new_maxspeed.append(GetSpeedGroup(x))
```

After all the groups had been added into a list, the list was then inserted back into the dataframe.

Listing (4.13) How to replace an already existing column with a new list.

```
df['maxSpeed'] = new_maxspeed
```

The process was repeated for all features that was meant to be grouped. These features were: maxSpeed, nearbyLandUse, nearbyLeisures, nearbyAmenity, and nearbyBuildings.

4.2.3 Algorithm implementation

To make sure that there was no bias in the data a model was trained using the XGBoost algorithm to determine what features was deemed important to the tree-model. Based on the feature_importance returned by the model it was safe to assume that the dataset had no bias-feature, there was a spread of importance in all features, and the feature most important was speed-limit which has no direct correlation with

our ground truth. Once this was complete a process of hyper tuning both models started to not allow personal-bias disturb the potential findings.

XGBoost

Listing (4.14) Grid search wtih XGBoost

```
from sklearn.model_selection import GridSearchCV
from xgboost import XGBClassifier

xgb = XGBClassifier(learning_rate=0.02, n_estimators=600,
                    objective='binary:logistic',
                    silent=True, nthread=1)

#params used for second iteration grid-search (narrow search space):
params = {
        'min_child_weight': [0.25,0.5,0.75,1],
        'gamma': [1.25, 1.5, 1.75],
        'subsample': [0.4, 0.5, 0.6],
        'colsample_bytree': [0.7, 0.8, 0.9],
        'max_depth': [5, 25, 100]
        }

grid_search = GridSearchCV(xgb, param_grid=params,
        scoring='roc_auc', cv=5, n_jobs=-1,
        cv=skf.split(X_train, y_train), verbose=5)

grid_search.fit(X_train, y_train)
```

The XGBoost hyper parameter tuning was done in two iterations of grid search. First had a wide search space meaning that the variables used had a high variance. From the result from the first iteration a second grid search was done using a narrow search space with a low variance compared to the best performing parameters from the first iteration. After the second grid search returned the same result as the first iteration a decision was made that this would be the optimal model parameters for future usage.

4.2.4 Neural network

Listing (4.15) Random search for Neural Network (keras)

```
from sklearn.model_selection import GridSearchCV, RandomizedSearchCV
from keras.wrappers.scikit_learn import KerasClassifier
from keras.layers import Dense, Input, Dropout
from keras import Sequential

def create_model(dropout_rate,optimizer,init,layers,activation):
    model = Sequential()
    model.add(Dense(10, input_dim=7,
        kernel_initializer=init, activation='relu'))
    for layer in layers:
```

```
        model.add(Dense(layer activation='relu'))
        model.add(Dropout(dropout_rate))
    model.add(Dense(1, kernel_initializer=init,
        activation='sigmoid'))

    model.compile(loss='binary_crossentropy',
        optimizer=optimizers,
        metrics=['accuracy'])
    return model

kears_estimator = KerasClassifier(build_fn=create_model, verbose=1)

#parameters used in one of the
#last search to find the best amount of
#node for the layers.
layers = [(70,45,15,5), (20,15,10,5), (100,75,50,25)]
optimizers = 'adam'
dropout_rate = [0.1]
init = ['uniform']
epochs = [200]
batches = [5]
activation = ['relu']

param_dist = dict(optimizer=optimizers,
                  layers=layers,
                  dropout_rate=dropout_rate,
                  epochs=epochs,
                  batch_size=batches,
                  activation=activation,
                  init=init
)

grid = RandomizedSearchCV(estimator=kears_estimator,
    param_distributions=param_dist,
    n_jobs=14,
    cv=5,
    verbose=2)

grid_result = grid.fit(train_x, train_y)
```

The hyper tuning done on the Neural Network was done with a combination of random search and manual tuning. For each iteration done with random search one of the parameters were in focus. As mentioned in chapter 3 some parameters were predetermined to reduce computational time based on previous experience done by the writer of this thesis. The main focus of the tuning was to find the proper batch size, how many hidden layers, how many node in each layer, and lastly find the best dropout rate. This was followed by tuning the learning rate of the model by utilizing manual tuning based on the parameters given from the random search done previously. To do this you add a new value inside the optimizer used in this case: adam('learning rate). The default used during the random search was a value of 0.001. After further testing 0.0001 yielded best result and will be used for future

training.

Final training

To make sure everything is done correctly after the parameter tuning is done, four models are created using a standard model creation approach by defining all parameters and having the validation set as a baseline for the tuning of the models. Two XGBoost and two Neural Network models was made differentiated by using original features and grouped features. Once they all had been trained they are fed the training set to do an accurate prediction. For both the XGBoost and Neural network model the confusion matrix is returned and recorded and four metric calculations are done to verify the result:

Accuracy The overall performance of the model.

Precision Percentage of your results which are relevant

Recall Percentage of total relevant results correctly classified by your algorithm.

F1 score A combined score of both precision and recall.

Lastly, for both the XGBoost models the feature_importance is extracted to see what features were most relevant.

4.3 Case study

As the example mentioned in the introduction chapter Kidal in Mali, Africa was used as an example and therefore the entire country Mali in Africa will be the base of our case study. It is worth mentioning after having a look at geofabrik's data on African countries[8] it was apperant that a lot of data was missing as the file size of a country would differ from a few kb to just about 100 mb of data. Mali seemed to be a nice middle point as it had a total of 92MB of data. The total dataset consists of close to 19 million datapoints, and after extracting the ways containing the 'highway' tag we were left with 250601 ways.

From here on out feature extraction was done by following the approach done on the Norwegian dataset, but first it was necessary to check if Mali had the same level of tagging as Norway having a ground truth in regards to sidewalk. After a loop going through the DataFrame of ways not one node contained the "'foot': 'yes'" tag used for the Norwegian dataset, and only 62 of the 250601 ways contained the key-tag "foot". This also excluded 'footway' as this value includes 'foot'. From that a conclusion was drawn that the original hypothesis that the developing countries were lacking data was correct. Now knowing there was no way to gather the actual ground truth it became apparent that only option was to create a pre-trained model from a developed country.

[8] https://download.geofabrik.de/africa.html

Before increasing the amount of datapoints by converting it to way-nodes some tags could be valuable to check out. Especially if any 'maxspeed' key could be found in the dataset and if the value of 'highway' was able to be translated into a features like it was done on the previous experiment. 'Maxspeed' had 191 cases, but 'highway' was containing values for type of roads. Knowing this a new model would have to be created without the maxspeed feature, but road type could be kept.

The first thing to do when extracting the features was to: first get the count of nodes in the way, and second get the road type based on the same function used for the Norwegian dataset.

Listing (4.16) Creating the Mali road groups.

```
highways = list()
for i in range(len(newDf)):

    tempString = str(newDf.iloc[i].tags)
    tagsInDf = eval(tempString)
    if 'highway' in tagsInDf.keys():
        highways.append(tagsInDf['highway'])
    else:
        highways.append('Uknown')

areaType = list()

#ReturnFeature groups the
#'highway'-tag in same groups as the Norwegian dataset.
for each in highways:
    areaType.append(ReturnFeature(each))

newDf['roadType'] = areaType
grid_result = grid.fit(train_x, train_y)
```

Once this was complete all data from the ways were extracted and could now be converted to nodes. This was done though a similar iteration, but had to some string conversion to get the proper data out. The best approach was to have all nodes imported into a separate DataFrame and then setting nodeId as its index. By using pandas syntax you would allow for faster searches and you could insert the nodeId contained in the way-datasets column 'refs' to get the correct longitude and latitude.

Listing (4.17) Converting Mali ways to nodes.

```
for i in range(len(newDf)):

    currentString = str(newDf.loc[i].refs)

    currentString = currentString.replace("(", "")
    currentString = currentString.replace(")", "")
```

```
    nodeNumbers = list(currentString.split(","))
    for j in range(len(nodeNumbers)):
        currentNodeNumber = int(nodeNumbers[j])

        if(currentNodeNumber in nodeDf.index):
            lon.append(nodeDf.loc[currentNodeNumber].lon)
            lat.append(nodeDf.loc[currentNodeNumber].lat)

            wayId.append(newDf.loc[i].ids)
            connectedNodeAmount.append(newDf.loc[i].AmountOfNodes)
            roadType.append(newDf.loc[i].roadType)
            nodeId.append(nodeNumbers[j])
        else:
            removedEntries += 1
```

Compared to Norway, Mali had a lot more roads contained in its dataset and for this reason the first 1.000 data points were kept for prediction to reduce computational time instead of the original 4.702.391 datapoints.

```
completeDf = finalDf.iloc[:1000][:]
```

Finally by using the same C# script we could check the remaining four features: nearbyLandUse, nearbyLeisures, nearbyAmenity, nearbyBuildings. This is when you could notice there was a big difference in amount of data. The CSV containing all nodes with the key 'landuse' had a total of 60 entries, and this repeated itself for all four features. This lead to having to remove them from the initial model. The end result was a dataset containing two possible features to test on. By creating another XGBoost model based on these two features and Norwegian road data we could get some predictions. Based on the predictions four images were produced to see if there is some correlation with the prediction and the images.

Now that all the predictions were ready, a small windows application was created using Visual Studio and two NuGet packages GMap and CSVHelper. By reading a CSV exported from Python and combining it with satellite images from Bing[9] it created a way to get an overview of the results returned by the predictions.

Listing (4.18) Initializing the GMap functionality.

```
//gMapControl1 == the window containing the map.
gMapControl1.MapProvider = GMapProviders.BingSatelliteMap;
gMapControl1.Position = new PointLatLng(-3.008200, 16.770149);
gMapControl1.MinZoom = 5;
gMapControl1.MaxZoom = 100;
gMapControl1.Zoom = 6;
```

And then a simple function was made to iterate through the different predictions made.

Listing (4.19) Next function to iterate through the predictions.

[9]https://www.bing.com/maps?FORM=Z9LH2

```
private void Next()
{
    Console.WriteLine("NextPoint");
    Random rand = new Random();
    int randomIndex = rand.Next(0, mainDf.Count-1);
    gMapControl1.Position =
    new PointLatLng(mainDf[randomIndex].lat, mainDf[randomIndex].lon);
    gMapControl1.SetPositionByKeywords
    (mainDf[randomIndex].prediction.ToString());

    GMap.NET.WindowsForms.GMapMarker marker
    = new GMap.NET.WindowsForms.Markers.GMarkerGoogle(
        new PointLatLng(mainDf[randomIndex].lat,
            mainDf[randomIndex].lon),
        GMap.NET.WindowsForms.Markers.GMarkerGoogleType.blue_pushpin);
    marker.ToolTipText =
    mainDf[randomIndex].prediction.ToString() + " is the prediction";
    markers.Markers.Add(marker);
    gMapControl1.Overlays.Add(markers);
}
```

The images were then extracted using Windows snipping tool using the similar size and zoom on each image.

After the extraction one thing became apparent in regards to the country of Mali. The country seem to have an underdeveloped road network, and the current state of the roads left little room for detecting sidewalks. Originally the country was based on finding a medium sized dataset in Geofabrik's database that could indicate that it was in between the most developed countries, and the worst developed countries in Africa. After a few searches about Mali, an article from the WorldBank [5] appeared and explained the economical difficulties that the country have had. This created a problem in regards to using Mali alone as an example for this study, and another developing country was chosen to be compared to Mali. Trying to find a similar sized dataset the country of Serbia was chosen. The data was extracted in the same manner as Mali. However, Serbia had multiple cases of datapoints containing the tags necessary to create features. The feature 'maxSpeed' was kept, as well as all the nearby objects used in the best performing model. All the datapoints were then put through the best performing model made previously using all the 'original features'. Further the data was plotted into the windows application created and pictures were extracted using a semi-random approach. A random integer was created within the size of all datapoints contained within the list. This was then followed by a manual inspection of the images, and chosen if the image provided valuable information. However, when analysing the images it proved difficult to make a final conclusion in regards to if the computer is right or wrong because of two factors. 1) We do not have a set ground truth that can be collected from the dataset. 2) It is hard with an untrained eye to spot a sidewalk on a low resolution image. Trying to cope with this two images from each tag was extract having zoomed the image as close as possible without blurring the image. Conclusions was then made based on manual analysis

and by primarily looking at the setting the road was at. Was it in a neighbourhood? Is it a highway? Out in the woods? There was also indication of white lines next to the roads which could appear to be sidewalks on inspection.

Chapter 5

Results

In the results chapter of this report we aim to describe all results that can answer the three research questions mentioned in Chapter 1. The chapter is devided into three parts: 1) Show the models using 'original features'. 2) Show the models using 'grouped features' 3) Show the images produced by the case study done on the two countries Mali, Africa and Serbia, Europe.

5.1 Original features

XGBoost

The XGBoost model performed better than expected, but had some limitations where it utilize 'maxSpeed' (speed limits) as a major factor to make its predictions (See figure 5.1). This feature can have a localized correlations with the ground truth, in other words the feature itself can have a direct link to how the rules and regulations are in Norway. This might create some problems when utilizing the model to make predictions in another country. It does however show that it can provide a good enough performance to predict sidewalks on Norwegian roads. The confusion matrix above reflects a good spread in both Positive and Negative guesses. The Recall and Precision scores also indicates that it is a generally good performing model, and when something is classified as positive or negative it is safe to assume that this is correct with a low chance for error.

Results from best performing XGBoost model using original features.

Accuracy	Precision	Recall	F1-score
97.57%	98.04%	98.28%	98.16%

XGBoost confusion matrix from original features.

		Predicted	
		True	False
Actual	True	4183	166
	False	145	8308

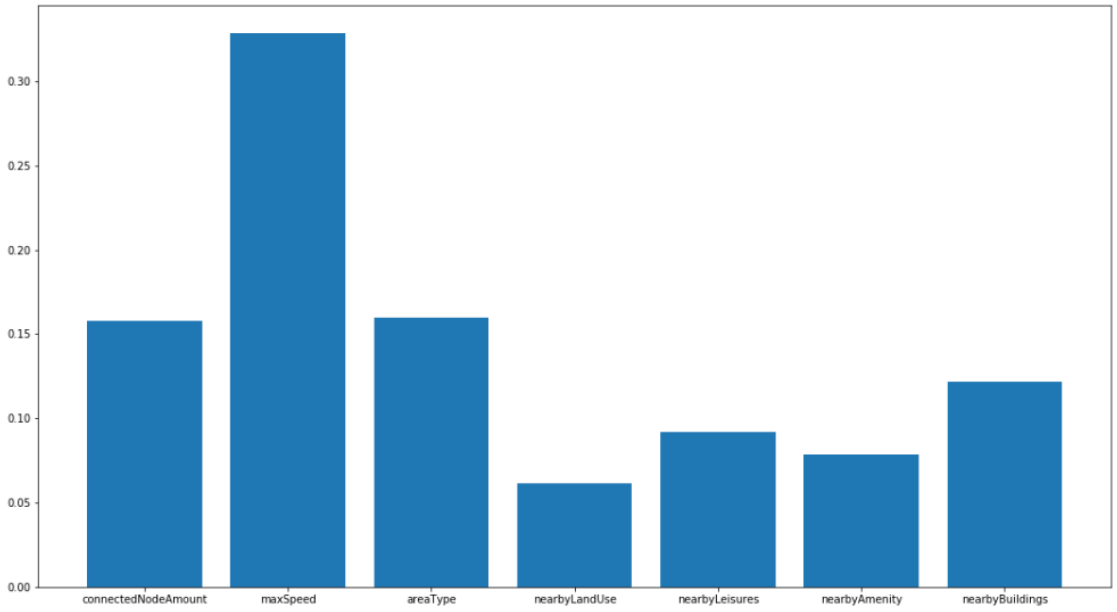

Figure (5.1) Feature importance done returned by the XGBoost model trained on 'original features'.

Neural Network

The Neural Network was outperformed by the XGBoost model having particularly bad performance in regards to false negatives shown in the confusion matrix above. Compared to its competitor XGBoost it seem to classify most cases where it is uncertain into the negative category. Due to the black box that is Neural Network it is hard to extract the feature importance as mentioned in the design chapter, which makes it difficult to see which features it deem the most important.

Results from best performing Neural Network model using original features.

Accuracy	Precision	Recall	F1-score
89.00%	88.43%	95.88%	92.01%

Neural Network confusion matrix from original features.

		Predicted	
		True	False
Actual	True	3289	1060
	False	348	8105

5.2 Grouped features

XGBoost

Even though the performance of the XGBoost model with 'grouped features' have a lower accuracy than the 'original features' it shows some importance discoveries

in regards to the spread in feature importance. Looking at figure 5.2 it is apparent that when features are grouped they become more important for the classification of sidewalks. Especially when looking at the feature nearbyBuildings which increased +/- 5%, and decreasing the importance of the maxSpeed feature by 5-10%. The confusion matrix indicates that the model shifts from having more false-positives to more false-negatives, as well as a shift in precision and recall allowing precision to be higher than the recall score. Despite of this the model still performs worse than the model trained with 'original features' when testing it on a test set consisting of data from the same country as it was originally trained on.

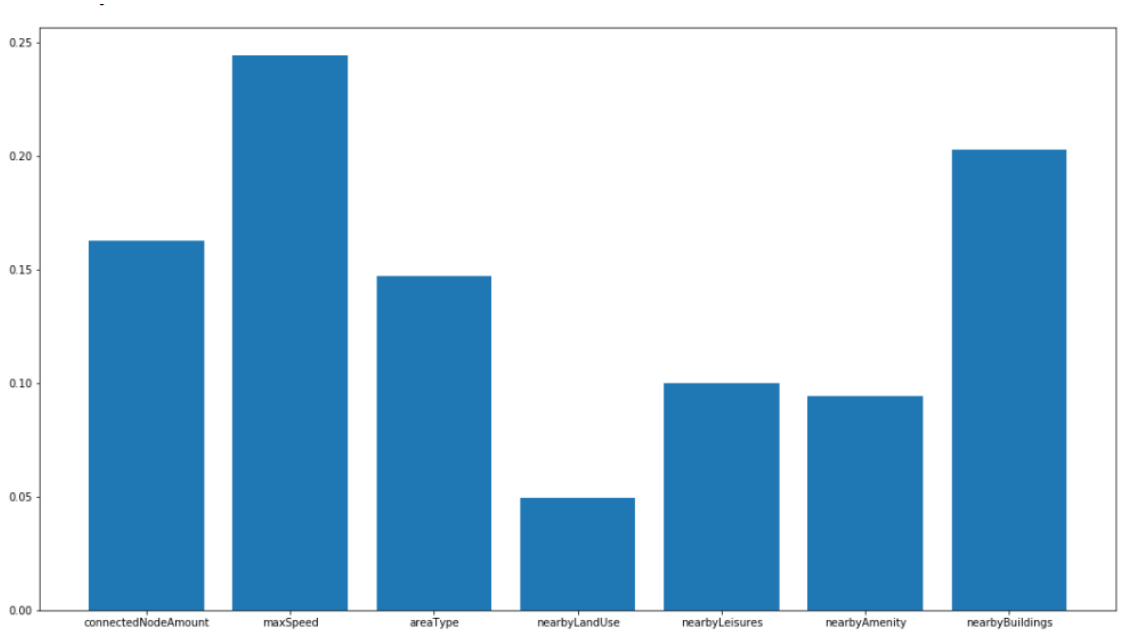

Figure (5.2) Feature importance done returned by the XGBoost model trained on 'grouped features'.

Results from best performing XGBoost model using grouped features.

Accuracy	Precision	Recall	F1-score
91.77%	94.57%	92.75%	93.65%

XGBoost confusion matrix from grouped features.

		Predicted	
		True	False
Actual	True	3988	446
	False	607	7761

Neural Network

Unlike the XGBoost model, the Neural Network did not see the same reduction in accuracy when changing to 'grouped features'. However, the performance of the model did still perform worse the with 'original features'. The biggest different in the two neural networks is the major increase in number of false positives predicted by the model with a total increase of 424 cases. It did however predict more true positives, which can indicate that the 'grouped feature' model allowed for more cases to be classified as positive.

Results from best performing Neural Network model using grouped features.

Accuracy	Precision	Recall	F1-score
85.21%	87.14%	90.77%	88.92%

Neural Network confusion matrix from grouped features.

		Predicted	
		True	False
Actual	True	3313	1121
	False	772	7596

5.3 Case study

5.3.1 Model tailored for Mali, Africa

The model created for the Mali dataset shows signs of overfitting on the feature that represents the type of road it is. The feature areaType has a 80% importance when deciding which label to give the prediction (seen in Figure 5.3). Despite of this the model still manage an accuracy of 79.38% when feeding two features in the model, making it apparent that there is some correlation between the features chosen and the ground truth.

Results from the model containing only road type and count of nodes in road.

Accuracy	Precision	Recall	F1-score
79.38%	86.39%	81.72%	83.99%

XGBoost confusion matrix from the model containing only road type and count of nodes in road..

		Predicted	
		True	False
Actual	True	3237	1091
	False	1549	6925

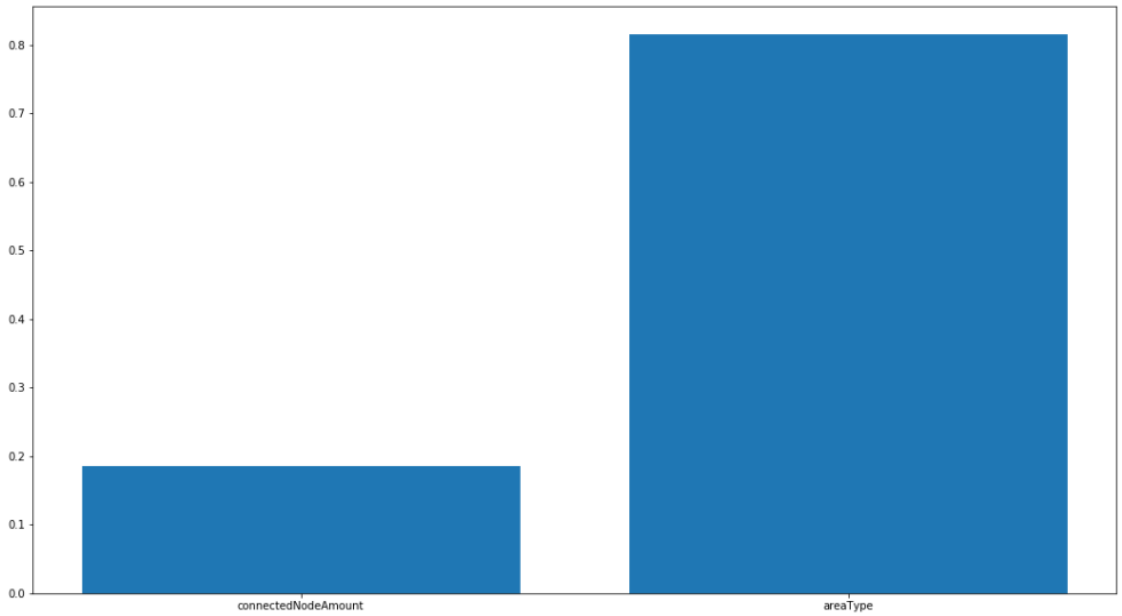

Figure (5.3) Feature importance from the model containing only road type and count of nodes in road.

5.3.2 Predicted results

Mali

Figure 5.4 and 5.5 represents three random selected images from each classification category. What is noticeable with the prediction is two things: 1) The images shows no sign of roads made with concrete. All roads found was either dirt or gravel which often do not contain sidewalks. 2) There is a clear distinction in the the between the categorise. Figure 5.4 has a lot more surrounding forest, and there is less sign of humans living in close proximity to the road. On the other hand figure 5.5 shows some houses, and has multiple roads which are easier to spot. The results shown can be traced to the 'highway' labels that indicates what road it is, and the feature importance of the model shown in figure 5.3. The model seem to predict primary roads / highways as roads with sidewalks, while having 'lesser roads' categorized without sidewalks.

Serbia

Serbia shows road structures similar to what you would expect from a European country and can be compared to the Norwegian road network used to train the model. Based on the knowledge from road structures in Norway in regards to sidewalks it becomes apparent that there are some similar features to where you would expect there to be sidewalks. In figure 5.6 and 5.7 it is a lot easier to spot difference in road types from just looking at the images. There are however some major mistakes with the predictions produced by the model. As an example having a positive label made indicating that there is a sidewalk, and when doing conducting a manual

Figure (5.4) Random selection of negative predictions in Mali.

Figure (5.5) Random selection of positive predictions in Mali.

inspection of the image it is easy to spot this false positive prediction. As the best performing model utilizing all features were used to produce these predictions, and based on figure 5.1 it can have some correlation with the 'maxSpeed' that differentiate between Norway and Serbia. Speed limits tend to vary from country to country and can leave inconstancy in the results because of it.

Deep dive into Serbian images.

To better understand the results shown in figure 5.6 and 5.7 close up images was shown to give a more thorough inspection of the predictions. These images can be seen in figure 5.8, 5.9, 5.10, and 5.11. The first we have a look at is the two negative cases produced:

Firstly, figure 5.8 shows what can be expected as an accurate image of a negative predicted label. This is due to the road representing a side-road of what looks to be a primary road going through a small town. It seems to be a dead end that goes to a couple of buildings, there is an indication that this might be a bus-parking space having multiple white entries that looks like bus-roofs. There are no apparent visual cues that make a sidewalk appear, and it can be concluded that the prediction is correct.

Secondly, figure 5.9. The first that become apparent when looking at this photo is that this is a primary road most likely in a rural or sub-rural area (an area just outside an urban area). There is also white indications around the road that can be thought to be a sidewalk. This however can also be gravel considering the roads leading away from the primary road having the same color. A sidewalk does not necessary have to be made out of concrete, and the prediction was considered inaccurate in regards to being predicted negative when there are indications that there is a sidewalk next to the road.

In figure 5.10 and 5.11 that provides a positive label prediction we can see that there is variety in the nodes predicted as positive. Figure 5.10 show similar traits as the inaccurate image from the negative label (Figure 5.9). The white line next to the road can indicate a sidewalk, and the surrounding landscape can indicate a rural / semi-rural environment. Figure 5.11 however is impossible to be analysed as there are many trees in the way that leave the image analysis incomplete. It is however worth mentioning that having a sidewalk inside a forest seems to be something that is a rare case, but it is hard to conclude if this is false or not as there are no clear visual on the road. There is houses nearby looking like a farm, which can indicate that the coordinate can be correct having a road there.

Figure (5.6) Random selection of negative predictions in Serbia.

Figure (5.7) Random selection of positive predictions in Serbia.

Figure (5.8) Close up of a thought to be accurate negative prediction.

Figure (5.9) Close up of a thought to be inaccurate negative prediction.

Figure (5.10) Close up of a thought to be accurate negative prediction.

Figure (5.11) Close up of a thought to be inaccurate negative prediction.

Chapter 6

Discussion

In this chapter the three research questions mentioned in Chapter 1 will be the baseline for a discussion when looking through the results. Firstly, we will go through data extraction due to how much time finding the correct data and preparing it required to reach the final results. Secondly, we will go through the best performing model using 'original features' and discuss the results from both the XGBoost model and the Neural Network model. Lastly, a discussion explaining the different images produced by the case study and comparing them to one another.

Data Extraction

One thing that has proven to be really important when working with OpenStreetMap data is the ability to understand that extracting and preparing the data is a time consuming endeavor, and getting the correct key values for the tags is something that require a cooperative combination of OpenStreetMap's iD-editor and OpenStreetMap's wiki-page. It is important to know that the value of tags can vary depending on who mapped out that part of the map, and irregular data might persist through multiple tags. When extracting value from these tags it is important to have a set of rules on how to deal with the irregular data. Is it supposed to be dealt as a 'NaN' variable, or should you manually go through the values to see if there is some information you can gather? Another important note is that the different countries might have different norms on how to fill in their value, or the value differentiate between countries based on that country's rules and regulations. The best approach to generalize data in this thesis was trying to keep all possible feature with in the 'key' space of the data, rather the 'value'. Despite it showing great promise on Norwegian and Serbian data, when accessing data from Mali it created more problems lacking many of the original features, having very few houses, amenities, leisure, and land-use objects. In retrospect it could prove valuable to create more features based on distance to nearest house, and have radius driven features that count amount of houses within raidus-one, radius-two, radius-three and so forth allowing more data from the surrounding area to be fed to the model. This would not affect developing countries, and looking at the results in chapter 5 the Norwegian based model performed above the 95% mark.

An important note is that in this thesis a Python library was used to read a .pbf file, but the original pbf-reader was created in Go-lang which again could be considered a better option to have a more systematic way to approach the data. The 'esy.osm.pbf' library used is just a small version of the tool available in Go-lang. It could potentially be better if large amount of data had to be extracted to contact the main developers of OpenStreetMap and to get access to an API that could do send request directly to the database removing the need for a third-party website to access the data.

When working with OpenStreetMap data it proved difficult to handle it using only Python do the data manipulation. In this thesis C# was utilized to do the more computational heavy calculations, especially the one taking distance from one point to another. It is recommended to plan out a good way of handling large amount of data when trying to replicate or produce a similar method. Sixty thousand data points checking over a hundred million nodes took five hours using the specs mentioned in Chapter 4. Looking back it could have proven more beneficial to export the entire pbf file into a local database and send request for data manipulation from there.

The best performing algorithm

When looking at the results shown in chapter 5 it is easy to see that the XGBoost model outperformed the Neural Network. It is however worth mentioning that the Neural Network could potentially have shown greater results if a large-long term parameter search had been done. Another way to look at it is that the core of the method was to find an alternative to computational heavy methods, and having a model using multiple days to fine tune its parameter could contradict its original intent. Training data only contained around 38.500 data entries, and can be considered a small dataset. With the data provided the optimal solution seem to go in XGBoost favour basing it on the 8.57% accuracy gap between the two models. XGBoost produce a good overall result not showing any discrepancy in either precision or recall, while the Neural Network model shows a good recall result but a less optimal precision result. From this it is safe to assume that the XGBoost model allow itself to predict a positive label more often than the Neural Network model. There is some variance in the two models when looking at the confusion matrix. XGBoost has more False-Positives than False-Negatives, but the different in the two is not significant. Neural Network on the other hand has a large difference in the two, having almost three times as many False-Positive than False-Negatives.

Difference between 'original features' and 'grouped features'

The Neural Network's core is what is known as a black box which makes it hard to do a feature analysis on the model to see what features are important for its decision making. XGBoost however is a tree-based model and allows for the extracting of feature importance. When looking at figure 5.1 then it is apparent that there is one feature that stands out. As mentioned in the hypothesis, the rules and

regulations in Norway require some road types and/or speed-limits to have a working sidewalk next to it. The results shown in the XGBoost feature importance (figure 5.1) correlates with this statement having a over 30% importance on speed-limit and a little over 15% importance from road type. ConnectedNode amount is the third most importance and is a OpenStreetMap feature that is based on how the road is mapped and represents either 1) the length of the road, or 2) how straight the road is. The positives side with the feature importance shown is that no feature seem unnecessary and all features contributes to the best result of 97.57% accuracy.

When looking into the 'grouped features' and how they perform it is safe to say that the model under performed compared to the previously best performing model using 'original features', but when analysing the feature importance (figure 5.2) created from the best performing model there is a larger spread in distribution. An importance thing to note is that nearby buildings outperformed both the structure of the road and the road type, while the speed-limit became less important just under 25% importance. There is also a positive increase for nearbyLandUse, nearbyLeisures, and nearbyAmenity. When reflecting on the two models one would possibly perform better in a Norwegian environment using the 'original data', while the 'grouped feature' model seem to have a better spread in features and can be considered more globalized because of it. The Neural Network had a decrease in all the matrix calculations, but the gap between precision and recall had decreased going from 7.45% to 3.63%. The False Positives had barley increased when changing from 'actual features' to 'grouped features', but the overall performance of the model fell 4.79%.

Developing countries, can the model be used?

The case study conducted was originally planned to have one developing country from Africa to be the primary source for exploring the possibilities of using a pre-trained model to categorize sidewalks. After images had been extracted from Mali two things became apparent: 1) African road structure seem to be underdeveloped having mostly dirt and gravel roads. 2) Due to the underdeveloped roads a more thorough expectation of the countries economical standing was done, and the results were an article rating Mali as one of the world's 25th poorest countries [20], ranking at number 22. When looking through the images in figure 5.4 and 5.5 it is easy to spot some differences, especially the difference between road types as there are more distinct roads on the primary roads. These roads can be thought to be roads that see traffic on a regular basis, as seen on image one in figure 5.5 it shows some houses that indicates a place where people travels. In figure 5.4 however, there are roads that seem to be less traveled as it blends more with the surrounding nature. Though there was not any concrete roads found during the manual analysis so it is safe to say that there are no need for detecting sidewalks until the road network is improved. This is however hard to conclude as there is to little data, the model lack sufficient training data to get other analytic angles, and a familiarity with the country's road structures is missing.

With the concluding factor that there is hard to find sidewalks in Mali, a second

experiment was conducted extracting data from Serbia. This country had a similar road structure to other European countries like the Norwegian road structure. The dataset included roads that had all the necessary perquisites to create the feature used in the best performing model. This resulted in images that could be analysed as the road network was detailed enough to conduct an actual image analysis. When comparing the images in figure 5.6 and 5.7 it is hard to spot any differences, but when comparing it using the feature importance from the model (figure 5.1 some things become apparent. 1) There is an indication that negative cases are often correlated with roads next to primary roads, it does however not predict this without fail. 2) The positive predictions seem to be heavily correlated with 'maxSpeed' as shown in image one figure 5.7, having predicted a sidewalk on a highway exit. 3) Positive predictions are also in correlation to primary roads, which can have a connection with the amount of traffic that is expected on the road.

As the images shown in figure 5.6 and 5.7 are from a high altitude it does not allow for an accurate assessment on the actual prediction outside of looking at the surrounding. To contemplate for this four new images were extracted from lower altitude, though there were still limitations having the image getting more and more pixelated as the altitude lowered. Due to this thesis having a limited budged the utilization of free satellite images was used through the map service from Microsoft Bing. This allowed for a certain altitude to be extracted as shown in figure 5.8, 5.9, 5.10, and 5.11. The manual analysis conducted indicates the model being inconsistent in its prediction but still show promise of having correlation with its surrounding. The indication of white next to roads can be thought as sidewalk, but it is again hard to do an accurate assessment without more detailed images and/or a ground truth.

Chapter 7

Conclusion

This thesis demonstrate a method to classify sidewalks by creating two different machine learning models based on the algorithms Neural Network and XGBoost in combination with Norwegian road data from the OpenStreetMap database. Out of the two algorithms XGBoost provides the best accuracy result with a 97.56% accuracy. It provides evidence that using OpenStreetMap data for classification purposes proves successful and show a potential method to accomplish this. By comparing the results to the best performing model from the predictSLUMS[10] experiment that looked to create a model that could differentiate between slums and informal settlements. The best performing model shown in the predictSLUMS experiment show an accuracy of 97.6% using Artificial Neural Network creating a total variance of 0.03% in the favor of predictSLUMS compared to the XGBoost model produced in this thesis. It does however not demonstrate a solution for classifying all objects in OpenStreetMap, and the findings can be limited to sidewalks alone based on the features utilized.

Secondly, a second range of models was created basing itself on another set of features. Originally the model was trained using the actual numeric value for all features, while for this experiment all numeric values were given individual groups to see how the model would perform. The difference between 'original features' and 'grouped features' proves that each approach should be considered when using such a model. When 'original features' are used some features are prioritized over others, and the model is most likely fit to use for specific cases as proven how inaccurate some prediction are when extracting them to unfamiliar data seen in figure 5.9 and 5.11. The 'grouped features' might perform better on a globalized level, but in returns reduce the local accuracy as shown in Chapter 5 under the grouped features section. The difference in accuracy was greatly decreased for the XGBoost algorithm, and slightly for the Neural Network.

Lastly, by using the best performing model from the last two experiments, images were created using a combination of spatial data, predictions, and a custom made windows application utilizing the Bing satellite imagery viewer. The experiment was conducted on Serbia and Mali and created a couple of criterion that had to be met in order to make an accurate assessment. 1) The road network has to be at such

a detailed level that you can extract the same amount of features as the original training data. 2) Some cases of ground truth from the developing country has to be incorporated into the training data to add localized correlations to the training. 3) When conducting an image analysis of satellite imagery through manual analysis it is important to have high resolution images to allow for an accurate assessment. This thesis does however show great indication that cross country models can be utilized when categorizing sidewalks if the criteria mentioned above is met.

Future work

There are multiple improvement that can be made to the methods used in this thesis. . 1) Creating an automated extraction of OpenStreetMap data. Looking back at the work done, procuring the data and getting it ready to work required the most time of all the processes in here. By having an automated extraction tool it could have saved countless hours of headache and frustration that could instead be used to optimize models and try other algorithms.

2) Testing the method on other minor objects. By also testing this method on other objects it will provide an answer to how standardized this method of using OpenStreetMap is, and how many objects it can be utilized for.

3) Create a multi continental dataset. By creating a multi continental dataset and making it public it can allow scientist all over the world to help create a unified method at predicting objects and continue the creation automation in OpenStreetMap mapping.

4) Use image analysis to help developing countries create a better starting point. As mentioned through the case study, the lack of data shown in Mali proves that this is where the biggest different can appear when using automated procedures. However, the data required to do this is not yet available, and projects like AI-Assisted Road Tracing from Facebook can provide this through satellite image analysis.

Bibliography

[1] Kdd cup 2020. https://www.kdd.org/kdd2020/kdd-cup. Accessed: 2020-06-05.

[2] Loss and loss functions for training deep learning neural networks. https://machinelearningmastery.com/loss-and-loss-functions-for-training-deep-learning-neural-networks/. Accessed: 2020-06-05.

[3] sklearn.model_selection.gridsearchcv. https://scikit-learn.org/stable/modules/generated/sklearn.model$_s$*election.GridSearchCV.html.Accessed* : $2020 - 05 - 04$.

[4] sklearn.model_selection.randomizedsearchcv. https://scikit-learn.org/stable/modules/generated/sklearn.model$_s$*election.RandomizedSearchCV.html?highlight* = *random*%20*searchsklearn.model*$_s$*election.RandomizedSearchCV. Accessed* : $2020 - 06 - 05$.

[5] The world bank in mali. https://www.worldbank.org/en/country/mali/overview. Accessed: 2020-06-05.

[6] Jeremy Castagno and Ella Atkins. Roof shape classification from lidar and satellite image data fusion using supervised learning. *Sensors*, 18(11):3960, 2018.

[7] Jiaoyan Chen, Yan Zhou, Alexander Zipf, and Hongchao Fan. Deep learning from multiple crowds: A case study of humanitarian mapping. *IEEE Transactions on Geoscience and Remote Sensing*, 57(3):1713–1722, 2018.

[8] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. *CoRR*, abs/1603.02754, 2016.

[9] Taïs Grippa, Stefanos Georganos, Soukaina Zarougui, Pauline Bognounou, Eric Diboulo, Yann Forget, Moritz Lennert, Sabine Vanhuysse, Nicholus Mboga, and Eléonore Wolff. Mapping urban land use at street block level using openstreetmap, remote sensing data, and spatial metrics. *ISPRS International Journal of Geo-Information*, 7(7):246, 2018.

[10] Mohamed R Ibrahim, Helena Titheridge, Tao Cheng, and James Haworth. predictslums: A new model for identifying and predicting informal settlements and slums in cities from street intersections using machine learning. *Computers, Environment and Urban Systems*, 76:31–56, 2019.

[11] Warren C Jochem, Tomas J Bird, and Andrew J Tatem. Identifying residential neighbourhood types from settlement points in a machine learning approach. *Computers, environment and urban systems*, 69:104–113, 2018.

[12] Pascal Kaiser, Jan Dirk Wegner, Aurélien Lucchi, Martin Jaggi, Thomas Hofmann, and Konrad Schindler. Learning aerial image segmentation from online maps. *IEEE Transactions on Geoscience and Remote Sensing*, 55(11):6054–6068, 2017.

[13] Max Kuhn and Kjell Johnson. Applied predictive modeling, 2013.

[14] Max Kuhn and Kjell Johnson. Over-fitting and model tuning. In *Applied predictive modeling*, pages 61–92. Springer, 2013.

[15] Vishal Morde. Xgboost algorithm: Long may she reign. https://towardsdatascience.com/https-medium-com-vishalmorde-xgboost-algorithm-long-she-may-rein-edd9f99be63d, 2019. Accessed: 2020-06-05.

[16] Maryam M Najafabadi, Flavio Villanustre, Taghi M Khoshgoftaar, Naeem Seliya, Randall Wald, and Edin Muharemagic. Deep learning applications and challenges in big data analytics. *Journal of Big Data*, 2(1):1, 2015.

[17] Karla Saldana Ochoa and Zifeng Guo. A framework for the management of agricultural resources with automated aerial imagery detection. *Computers and Electronics in Agriculture*, 162:53–69, 2019.

[18] Neil J. Salkind. *Exploring research*, 2017.

[19] Michael Schultz, Janek Voss, Michael Auer, Sarah Carter, and Alexander Zipf. Open land cover from openstreetmap and remote sensing. *International journal of applied earth observation and geoinformation*, 63:206–213, 2017.

[20] Samuel Stebbins. These are the 25 poorest countries in the world. https://eu.usatoday.com/story/money/2019/07/07/afghanistan-madagascar-malawi-poorest-countries-in-the-world/39636131/. Accessed: 2020-06-05.

[21] Mervyn Stone. Cross-validatory choice and assessment of statistical predictions. *Journal of the Royal Statistical Society: Series B (Methodological)*, 36(2):111–133, 1974.

[22] Hugh JW Sturrock, Katelyn Woolheater, Adam F Bennett, Ricardo Andrade-Pacheco, and Alemayehu Midekisa. Predicting residential structures from open source remotely enumerated data using machine learning. *PloS one*, 13(9):e0204399, 2018.