Master Thesis, August 2018

Image Segmentation using Convolutional Neural Networks for Change Detection of Landcover

Intelligence Augmentation using Open Data and Deep Learning

Author
Martin Boos

Supervisor
Prof. Stefan Keller

HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL

FHO Fachhochschule Ostschweiz

MSE | MASTER OF SCIENCE IN ENGINEERING

This page is intentionally left blank.

Abstract

Cadastral surveying keeps track of real property boundaries, such as buildings, parking lots, and roads. Periodically, every one to five years, the cadastral survey data must be refreshed. Generally, this means finding changes in landcover, based on aerial imagery such as orthophotos, lidar, and terrestrial measurements. The changes consist of new objects that have been added, existing objects that have changed, or objects that have been removed since the last survey. The renewal of these data requires substantial manual work because it is often not easy to identify the changes.

This study uses an approach in which machine learning is applied and a modern geographic information system (GIS) is used as an editing tool. The goal was to reduce the amount of manual labor required. Unfortunately, state-of-the-art technologies are not advanced enough to rely on completely when trying to find these changes. Instead, they can provide support for human work and use artificial intelligence called "intelligence augmentation". In this study, we analyzed selected objects and classes from landcover in an orthophoto and compared them with the current landcover vector data layer.

The result of this work was a QGIS plugin. The plugin sent an image of the current extent together with loaded cadastral survey data to the backend, which then predicted all objects within the classes using a neural network. The classes were building, highways, vineyard, tennis court and swimming pool. In the next step, these predictions were georeferenced and the actual changes were determined by comparing the old vector data from the cadastral survey to the predictions. Finally, the changes were sent back to QGIS and visualized there according to their type (changed, deleted, added). Because the network was trained using satellite imagery from Microsoft Bing and vector data from OpenStreetMap, only about 80% to 90% of the predictions were correct. The errors were due to the satellite imagery passing through a photogrammetric post-processing step, which can lead to displacements regarding OpenStreetMap vector data.

Management Summary

Introduction Cadastral surveying means keeping track of real property boundaries. Periodically, the cadastral survey data must be refreshed. Generally, this means finding changes in landcover, among other things, based on aerial imagery. The renewal of the data requires substantial manual work as it is often not trivial to identify the changes.

Goals Using machine learning and QGIS, a modern GIS editing tool, the goal was to reduce the amount of manual work required for renewing the cadastral survey data.

Results The result of this work was a QGIS plugin which supports the user in data renewal. The plugin uses a backend on a remote server, which predicts all objects of an image received from the plugin to find the changes in landcover (deleted, added, or changed) based on the current data. The accuracy of the predictions was about 95%.

Future Work Future steps would include extending the plugin to allow a large area to be defined, which is then automatically processed in the background. Additionally, by training the neural network with a larger data set, the prediction accuracy could be improved.

Acknowledgments

Prof. Stefan Keller for his creativity, his vision, and his support – not only throughout this thesis but also in projects that preceded it.

My beloved wife Nadine for always supporting me through listening, sharing ideas and perspectives, and sometimes by doing a bit of additional housework.

My family for always trying to understand what my thesis is actually about.

Declaration of Originality

I hereby declare that,

1. this thesis and the work reported herein was composed by and originated entirely from me unless stated otherwise in the assignment of tasks or agreed otherwise in writing with the supervisor.
2. all information derived from the published and unpublished work of others has been acknowledged in the text, and references are correctly given in the bibliography.
3. no copyrighted material (for example images) has been used illicitly in this work

Seengen, 07.06.2018

Martin Boos

Contents

Overview 1

This chapter gives a brief overview of the chapters in this thesis and their contents. Chapter 2 is about current technologies and developments in the area of computer vision in general and segmentation in particular. Chapter 3 introduces artificial intelligence and explains how it was used in this study. Chapter 4 gives an introduction into convolutional neural networks and their architecture in the context of image segmentation. Chapter 5 is about practical and technical challenges that were encountered during the work. Chapter 6 is a collection of theoretical and experimental results, such as participation in the crowdAI mapping challenge and the data generation tool Airtiler. Chapter 7 is mainly about the QGIS plugin that was developed to enable end users to use the backend for predictions of their data. Finally, Chapter 8 provides some ideas about future research that could flow from this study.

State of the art 2

2.1 Neural Networks

Various techniques and technologies allow for the building of neural networks for machine learning purposes. The increasing computational power of hardware, especially graphic cards, means that more complex and computationally heavy models are being built. Hence with every passing year, research leads to improvements in existing networks or to completely new networks.

A challenge in computer vision is instance segmentation, which not only tries to detect occurrences of a specific class but also tries to identify single instances within a class. That is, segmentation tries to find the contour. To compare various architectures built for such a purpose, various challenges have been published on the internet. One such leaderboard is from the Microsoft COCO challenge [1], which annually tries to find the most competitive candidates for various tasks. These tasks include object detection, instance segmentation, and keypoint detection.

At the time of writing, one of the best-performing neural networks for conducting instance segmentation was Mask R-CNN [2], based on Faster R-CNN [3]. Another highly performing architecture was U-Net [4], which was created for medical purposes but can also be used for semantic segmentation of any image data.

2.2 Picterra

Picterra[1], a platform to extract information from satellite or drone images, allows amongst others to detect building footprints.

This platform comes with several different object detection models, which are **building footprint**, **airplane**, **train**, **ship** and **building**.

2.3 Mapbox Robosat

Robosat[2] is an open source pipeline created by Mapbox[3], which allows to do feature extraction from orthophotos.

[1] https://www.picterra.ch/ (29.06.2018)
[2] https://github.com/mapbox/robosat (29.06.2018)
[3] https://www.mapbox.com/ (29.06.2018)

Introduction 3

3.1 Intelligence Augmentation

One of the most time-consuming aspects of the cadastral surveying is to find the actual changes – for example, a newly built house or a garage added to an existing building. In general, this is a tedious task that cannot easily be automated. Our intention was not to develop a fully automated method but instead to support the person who is doing the work, with such support provided by artificial intelligence (AI). This form of AI is called intelligence augmentation (IA) [5] and is rather an old concept. As described in the previous chapter, our intention was to provide IA using state-of-the-art technologies. To illustrate the difference between IA and AI, Figure 3.1 shows some examples.

	Intelligence Augmentation ("IA")	Full Automation ("AI")
Enterprise	• Special-purpose learning technologies that automate many mundane tasks at work • Virtual automated assistants • AR glasses used by oil workers, repair professionals, surgeons	• Artificial General Intelligence – machines can perform any task a human can – and better + faster
Autonomous Vehicles	• Automated driving on highways, campuses and in other access controlled situations. Humans needed to handle exceptions and tougher driving conditions, coordinate/operate the overall system	• Networks of fully autonomous vehicles. Society moves away from vehicle ownership and human driving is made illegal • Fully Autonomous vehicles, drones redefine logistics
Robots / Industrial IoT	• Collaborative Robots that work alongside humans and can handle tasks that are hard, unsafe or repetitive • Connected Factories – Predictive Maintenance, Asset Monitoring, Better safety	• Fully automated factories. Robots and automated systems replace all humans on the factory floor
Drones	• Intelligent (but human piloted or limited scope) drones for inspections of cellular towers, building roofs, power cables, remote areas – enabling many new tasks that are unsafe or impossible for humans	• Fully autonomous drones that don't require human supervision, intervention or processing

FIGURE 3.1 Comparison between IA and AI
Source: https://www.cbinsights.com/research/ai-vs-intelligence-augmentation-applications, 02.06.18

3.2 Architecture

Figure 3.2 shows an overview of the information flow between all participating components of the pipeline developed throughout this thesis, with a combination of both the learning and the prediction architecture. Step 2 (loading the satellite imagery from Microsoft Bing) was required only in the learning phase; that is, for the generation of the training data which were required to train the neural network.

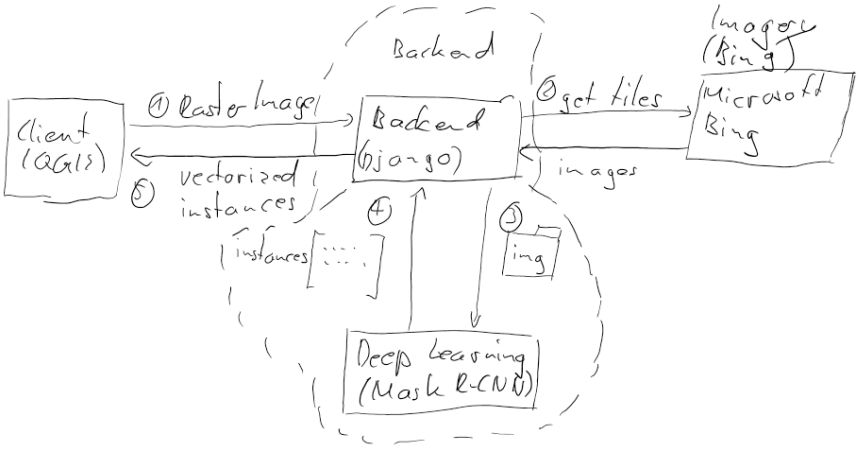


FIGURE 3.2 Architecture overview

Figure 3.3 shows the data flow during the training of the neural network. For this step, satellite imagery from Microsoft Bing was used, and OpenStreetMap (OSM) data were also mapped. The satellite imagery was downloaded tile-per-tile for a predefined bounding box and zoom-level; at the same time, binary images were created from the OSM data, which represented the ground truth. To simplify this step and make it available to the public, a tool called Airtiler [6] was created and published.

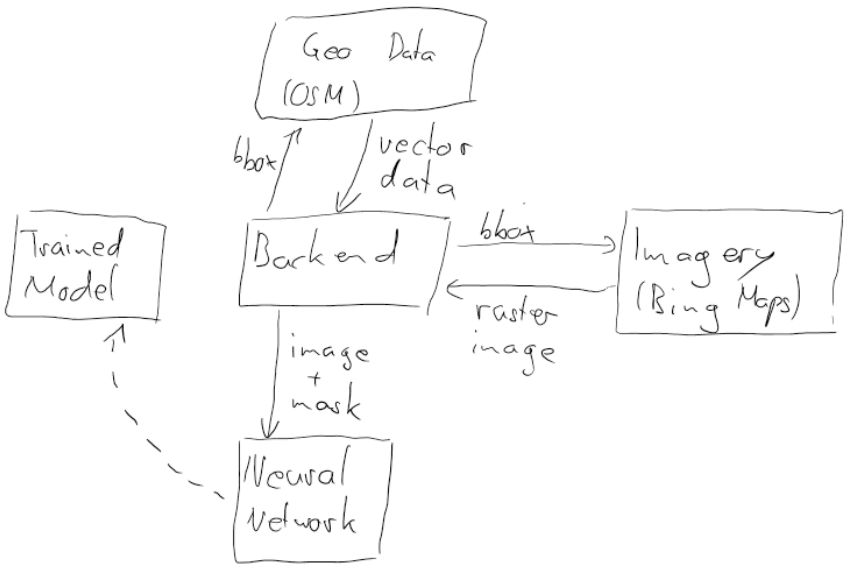


FIGURE 3.3 Learning phase

As soon as the training was completed, the prediction could begin. For this thesis, the prediction was split into two phases. Figure 3.4 shows the data flow of phase 1, which passed the current extent as base64 encoded image data as well as the bounding box of the current QGIS extent to the configured backend webserver. The pretrained neural network was then used to predict all instances on the current image. In the next step, the predicted instances were georeferenced using the bounding-box information that was sent to the backend at the beginning of the prediction phase. Finally, the georeferenced instances were sent back to the frontend client, the QGIS plugin. The plugin then visualized the data on a new layer in QGIS.

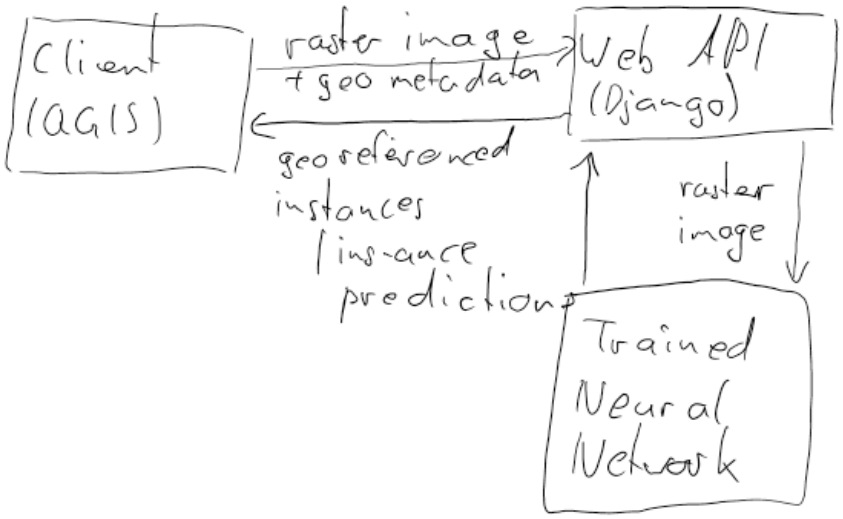


FIGURE 3.4 Prediction phase 1

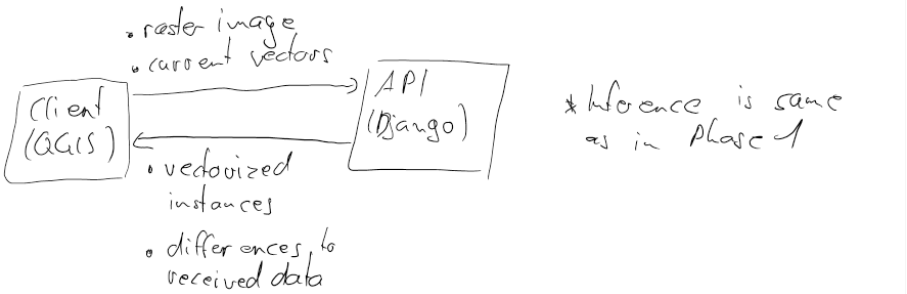


FIGURE 3.5 Prediction phase 2

3.3 Hardware Requirements

To train the neural network, a graphics card is required. We used a Nvidia P100 with 12 GB of memory. A Nvidia GTX 750 was insufficient as it had only 2 GB of memory. Additionally, CUDA[1] had to be installed as well as Docker[2] and Nvidia-docker[3]. How the docker image can be built and run is described in detail in the readme of the repository[4].

3.4 User Scenario

This section describes a user scenario that can be used for demonstration or to instruct the actual end user in how to use the plugin with its corresponding backend. This user scenario requires the plugin to be correctly installed in QGIS. The backend must also be configured correctly in the settings of the plugin, which can be accessed through the dropdown menu of the plugin (Figure 3.8). Additionally, the backend was required to be running.

1. Start QGIS
2. 2. Load the cadastral survey data. This may, for example, be a zip archive containing shape files according to the MOpublic [7] specification. In this case, the archive can simply be dragged and dropped into QGIS. A dialog will then open, which allows the use to select several layers.
3. Select the layers containing buildings or other geometries. Typically, according to MOpublic, this layer is named *BB_BoFlaeche*.
4. Load an imagery layer like Bing maps or Google maps, which can easily be done using the *OpenLayers* QGIS plugin.
5. Open the prediction dialog (Figure 3.6) by clicking the plugins icon which appears on the **Manage Layers** toolbar. Figure 3.7 shows the plugins icon on the toolbar. **Alternative:** click the arrow next to the icon and select an option from the dropdown.

[1] https://developer.nvidia.com/cuda-toolkit (22.06.18)
[2] https://www.docker.com/ (22.06.18)
[3] https://github.com/NVIDIA/nvidia-docker (22.06.18)
[4] https://github.com/mnboos/osm-instance-segmentation/blob/master/README.md (22.06.18)

6. Select the imagery layer from the dropdown. As shown in Figure 3.6, a preview of the image that will be sent to the backend is shown at the bottom of the dialog. The appearance of large white areas means that one or more tiles were loaded incorrectly from the corresponding service. In this case, the Refresh button must be clicked at least once, until all white blobs have disappeared.

7. Finally, by clicking the Predict button, the dialog closes and the vectors found are sent to the backend. **Note:** The backend might take a while to process all the data and hence QGIS might close the network connection; this results in an error if the backend tries to send the response. To solve this problem, the network timeout in the QGIS settings (Figure 3.9) should be set to a higher value. The connection will then be kept open for a longer period so that the backend can send the response without problems.

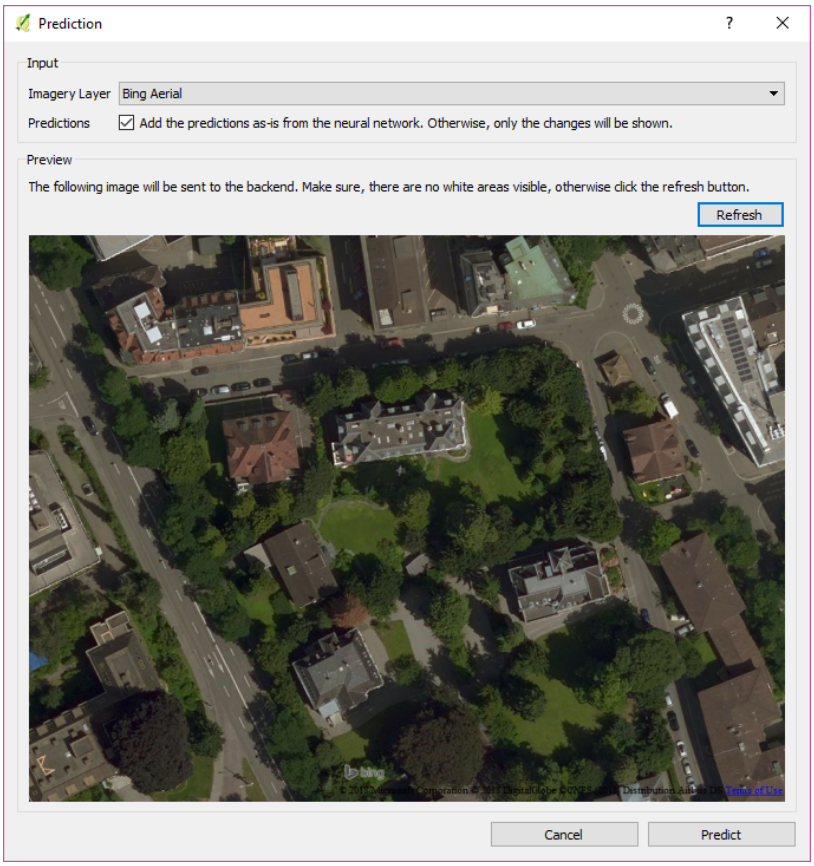


FIGURE 3.6 The prediction dialog

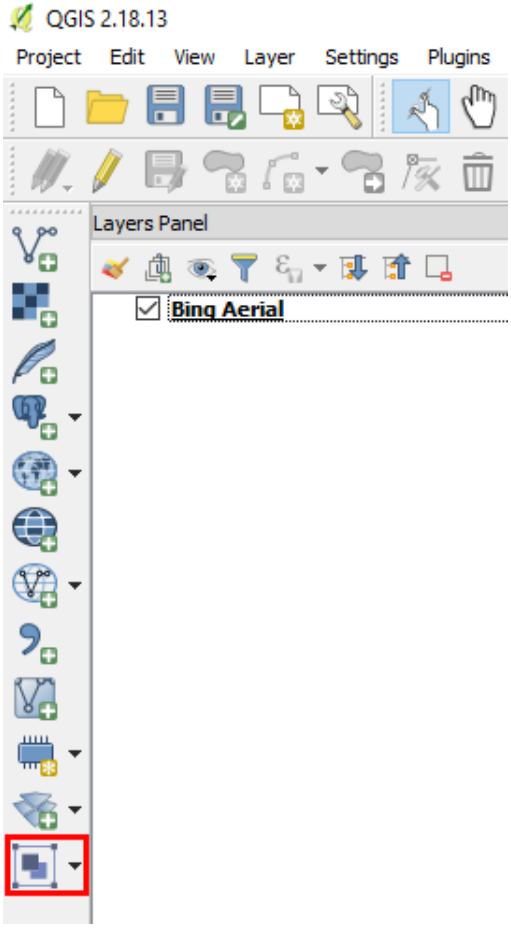


FIGURE 3.7 The plugins icon on the Manage Layers toolbar

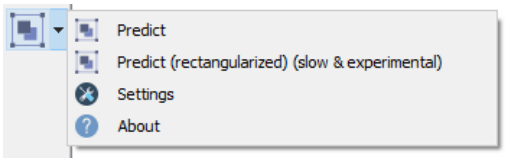


FIGURE 3.8 The plugins dropdown menu

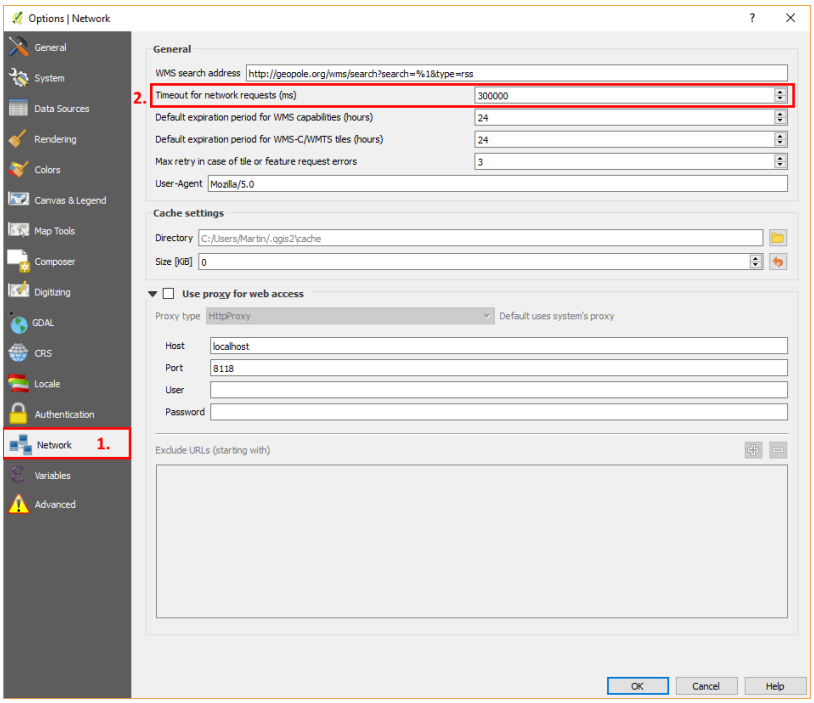


FIGURE 3.9 Network timeout configuration in QGIS

As mentioned earlier, the plugin searches for existing vectors in the loaded layers and sends those that fulfil the QGIS expression **"Typ" in ('Gebaeude', 'Strasse_Weg', 'Reben', 'Wasserbecken', 'uebrige_befestigte')** to the backend. For further details about the communication between the QGIS plugin and the backend, the API specification appear in the appendix of this paper.

Image Segmentation with Convolutional Neural Networks (CNN) 4

4.1 Introduction

With the increasing computational power of recent graphic cards, increasingly complex neural networks can be used for increasingly challenging tasks. In the area of image processing, deep learning is gaining in popularity, partly because of the wide availability of data sets. In addition, companies recognize the vast amount of knowledge and information that can be retrieved with such technologies. The following sections provide a brief introduction to image segmentation using convolutional neural networks.

4.2 Object detection and segmentation

Object detection existed long before deep learning became popular. In object detection, the goal is to determine whether an object of a specified class (for example “car”) occurs within an image. Another type of object detection is additional classification, which means finding all objects in an image, together with their class, and calculating the probability that the object belongs to the determined class. Figure 4.1 shows an example of object detection and classification.

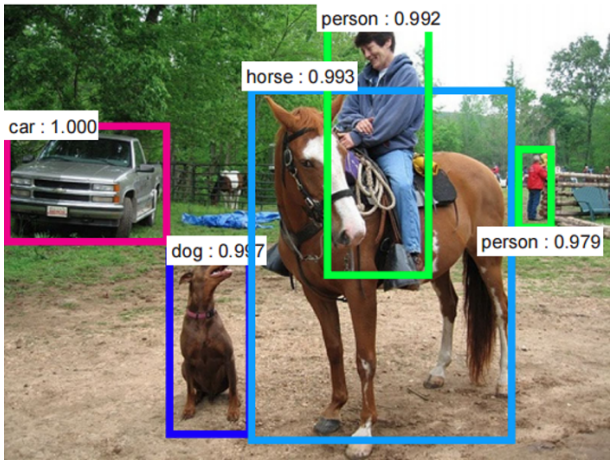


FIGURE 4.1 An example of object detection, showing the detected classes and the confidence level for each prediction.
Source: https://dius.com.au/2016/12/06/the-cleverness-of-deep-learning/ (27.05.2018)

In contrast to object detection, the aim of image segmentation is not only to state whether an object occurs in the image but also to label each pixel of the image with a class, such as "car" or "building". Different types of image segmentation are shown in Figure 4.2.

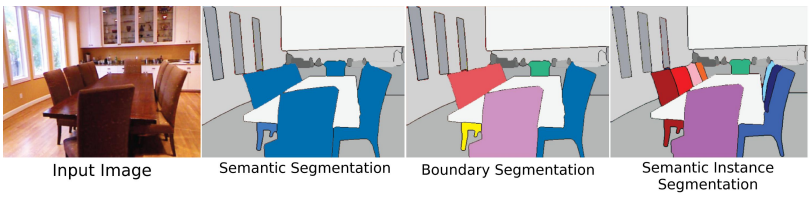


FIGURE 4.2 Types of image segmentation.
Source: https://i.stack.imgur.com/mPFUo.jpg (27.05.2018)

4.3 Convolutional Layer

A convolutional layer is one of the most basic and important building blocks of a neural network. It has several filters, each of which is small compared to the input volume (the image) – for example, 5x5x3 pixels (a 5x5 filter with 3 channels, because standard images have 3 color channels). During the forward

pass of the network, the filters are moved over the input image. At each position of the filters on the image, a convolution is computed, which is an element-wise matrix multiplication and a sum over the resulting matrix. The result of this operation is an activation map; this map is the output of the convolutional layer.

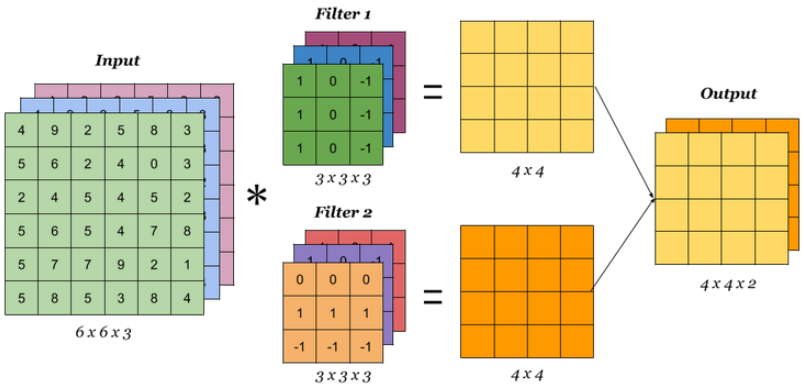


FIGURE 4.3 Convolution with multiple filters
Source: https://idoml.com (03.06.2018)

The size of a filter can be configured, as well as the step size, the stride, and the amount of zero padding around the input image.

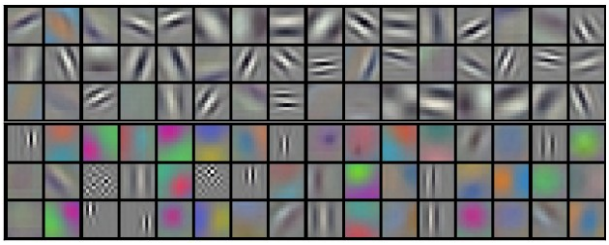


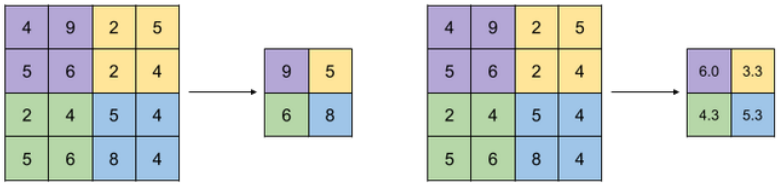
FIGURE 4.4 Example filters learned by AlexNet [8].
Source: http://cs231n.github.io/assets/cnn/weights.jpeg (27.05.2018)

4.4 Pooling Layer

Pooling is a technique which allows reduction in the size of an image by extracting a single value from a region of values. The extracted value depends

on the pooling type that is used. The most common pooling types are maximum (max) pooling for extracting the highest value from the current field, and average (avg) pooling, which extracts the average value of the current region.

A pooling layer has basically three parameters it can be configured with. First, there is the stride s, which is the distance the filter is moved. Second, there is the filter size f, which determines the width and height of the filter that is used to extract the value from the input.



(A) Max Pooling (B) Average Pooling

FIGURE 4.5 Max and average pooling
Source: https://idoml.com (02.06.2018)

As described in the previous chapter, a layer can have many output channels. Pooling can also be performed with multiple channels, as shown in Figure 4.6.

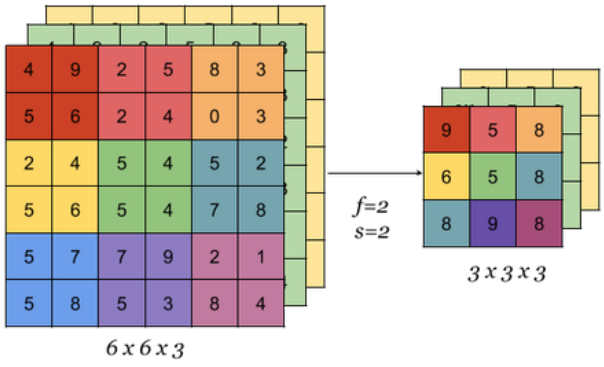
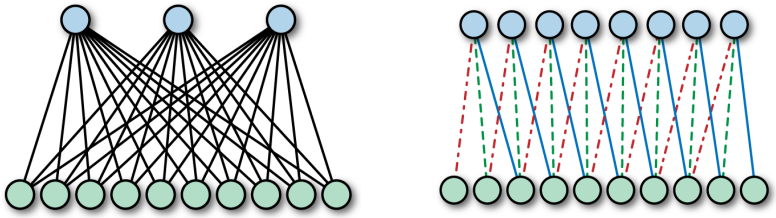


FIGURE 4.6 Max pooling with multiple channels
Source:https://idoml.com (03.06.2018)

4.5 Fully Connected Layer

A fully connected layer is almost identical to a convolutional layer. The only difference is that the output of a convolutional layer is spatially connected to a region of the previous layer, but not to the whole output. Figure 4.7 shows the difference between a fully connected layer and a convolutional layer.



(A) Fully Connected Layer (B) Convolutional Layer

FIGURE 4.7 Fully Connected and Convolutional Layers
Source: https://www.safaribooksonline.com/library/view/learning-tensorflow/9781491978504/ch04.html (07.06.2018)

4.6 Mask R-CNN

A massive amount of research would be required to develop or improve an existing state-of-the-art neural network architecture. This was beyond the scope of the study, so we decided to use Mask R-CNN [2]. As the name implies, this is a recurrent, convolutional neural network.

Practical Challenges 5

5.1 Training data

There are two options when selecting a data set that can be used to train the neural networks. Either an already available data set is used, whether on a paid or free basis, or a new data set is created. Currently, machine learning and deep learning are popular topics, especially deep learning. As a result, more data sets are available publicly and at no cost, especially in the area of image processing – such as segmentation, facial recognition, and object detection. Free data sets consisting of aerial imagery are provided by several sources ([9], [10], [11], [12], [13], [14]). Despite this availability of data sets, we decided to make our own. The data set would consist solely of open data, using Microsoft Bing for the imagery and OpenStreetMap for the vector data. Due to this, a tool named Airtiler [6] was developed and is described in detail in Chapter 6.

It can be assumed that in the future, more Swiss cantons[1] will make high-resolution orthophotos publicly available. At the time of writing, the canton of Zurich played a pioneering role and made several of its data sources publicly and freely available[2]. However, at the time of this writing it was not an option to use these images for our work because it would have been a rather small data set.

[1] Switzerland consists of 26 cantons, which form the Swiss Confederation
[2] https://geolion.zh.ch/ (15.06.18)

5.2 Prediction accuracy

5.2.1 Class probability

After the first training of the neural network, the results were not quite as good as expected. Although most of the buildings were predicted as buildings, other classes – like tennis courts – were also predicted as buildings. The network was therefore retrained using the additional and incorrectly determined classes (like tennis courts). However, instead of the distinction between buildings and tennis courts becoming more accurate, the overall prediction accuracy was even worse. This might have been the result of the network having to solve a more complex task, by deciding which class an object belonged to, instead of a simple yes-no decision. In addition, the training data were highly imbalanced as the images contained many more buildings than tennis courts.

A solution could be to train the network separately several times to obtain several models, each trained for a specific class. Another solution could be to weight the loss according to the relative amount of the specific class, based on the size of the whole data set.

5.2.2 Outline

Figure 5.1 shows that the predictions were generally rather too small, compared to the corresponding orthophoto. This might have been the result of slightly misaligned masks, since the masks and the images were generated separately.

FIGURE 5.1 Too small predictions

5.3 Regularization of building outlines

Once a network has been trained, it can be used to make predictions on images it has never "seen" before. However, in certain situations the predictions are far from perfect, especially if the building is partially covered by trees or has unclear outlines. Examples are shown in Figure 5.2. The predictions in such cases are mere approximations of the actual ground truths.

(A) Actual ground truth (B) Predicted building masks

FIGURE 5.2 Predicted masks and actual ground truth

Because of the inaccuracies in predicted building masks, the contours of such predictions cannot directly be used to create the vectorized outlines. Instead, the predictions must be regularized. The approach we used is shown in Figure 5.3 and was similar to that described in literature [15]. The single steps are described in detail in the following sections.

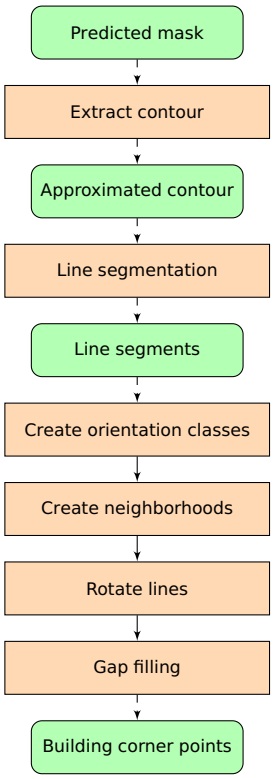


FIGURE 5.3 Rectangularization procedure

5.3.1 Contour extraction

The first step of the building outline regularization procedure consists of obtaining the contour from the predicted mask, which covers the whole building. The extraction is performed using the marching squares algorithm [16]. In this algorithm, a square consisting of four cells is moved (marched) along the contour in such a way that at least one cell always covers the object to be contoured. Additionally, the square always has a state that is derived from the content of its cells, according to (5.1). The cells are traversed in counter-clockwise order.

$$\begin{aligned} s &= \sum_{i=0}^{3} 2^i f(c_i) \\ &= 2^0 * f(c_0) + 2^1 * f(c_1) + 2^2 * f(c_2) + 2^3 * f(c_3) \\ &= f(c_{bottomLeft}) + 2 * f(c_{bottomRight}) + 4 * f(c_{topRight}) + 8 * f(c_{topLeft}) \end{aligned} \tag{5.1}$$

where:

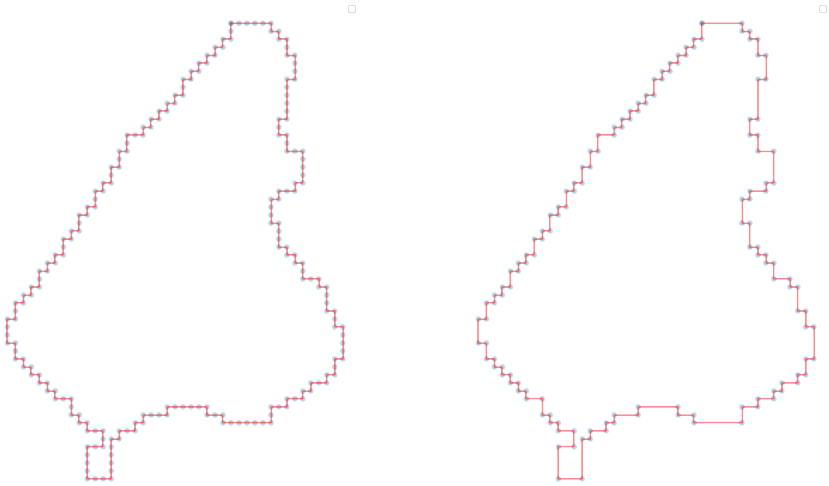
c_i: The value of the cell i

c_0: The bottom left cell

and

$$f(c_i) = \begin{cases} 0 & \text{if } c_i \leq 0 \\ 1 & \text{if } c_i > 0 \end{cases}$$

After the contour has been extracted, its number of points is reduced using a Douglas-Peucker algorithm [17]. The reason for this is that the contour has pixel accuracy. That means there may be several points on the same horizontal or vertical line, although the startpoint and endpoint of each line would be enough to represent the line. In addition, the fewer the points that are represented on a contour, the faster the processing will be.



(A) Original contour (B) Approximated contour

FIGURE 5.4 Contour before and after approximation. It can be seen, that the contour on the right consists of much lesser points.

5.3.2 Line segmentation

Once the contour has been extracted, it is split into many line segments. To do so, the main direction of the building is determined using the Hough Transformation [18]. The result of the Hough Transformation is a data-structure which contains an angle, a distance and a number. The combination of the angle and the distance, from a predefined reference point, leads to a line. The number is the number of points which lie on the constructed line. This algorithm can be used to detect the main building orientation, that is, the longest contour-line of any orientation. The angle of the line that is found is called the main building orientation.

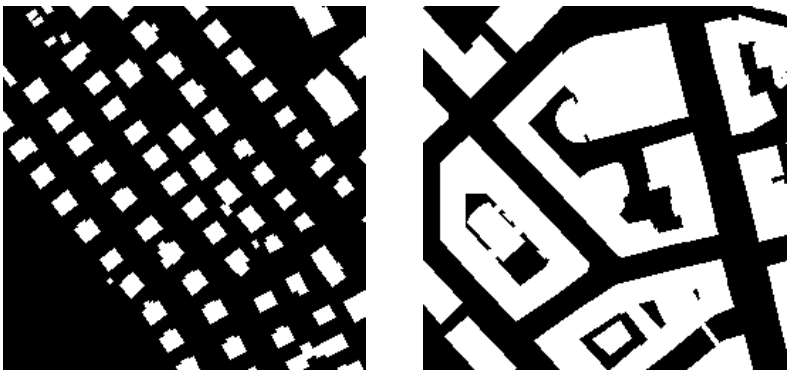
Once the main building orientation is known, the line segmentation starts at the point which has the smallest distance from the line. The whole procedure is depicted in algorithm 1.

```
Data: Contour points, startpoint
Result: Lines
rearrange points so that startpoint == points[0]
lines = []
while any points left do
    segment = remove first 3 elements from points
    while points is not empty do
        p = points.first()
        err = root mean square error of distance between segment.last() and
         p
        if err > threshold then
            break
        end
        segment.append(p)
        points.remove(p)
    end
    if segment.length() >= 3 then
        line = fit line to points of segment
        lines.append(line)
    end
end
```

Algorithm 1: Line segmentation

5.3.3 Create orientation classes

After the line segmentation, an orientation is assigned to each line. Generally, most buildings consist of mainly right angles, although some buildings do exist for which this assumption is untrue. Hence, orthogonality is preferred but other angles remain possible. Algorithm 2 shows the procedure which assigns a main orientation class to each line, and defines the line's nature in terms of it being parallel or orthogonal to the longest line in the same orientation class.



(A) Regular buildings (B) Non-orthogonal buildings

FIGURE 5.5 Various building types with regard to corner angles

Data: Lines, angleThreshold
while *any line without orientation* **do**
  line = longest of unprocessed lines
  line.orientation = angle between line and horizontal-line
  foreach *line li without orientation* **do**
    a = angle between line and li
    if $a \leq$ *angleThreshold* **then**
      li.orientation = line.orientation li.orthogonal = line.orthogonalTo(line)
    end
  end
end

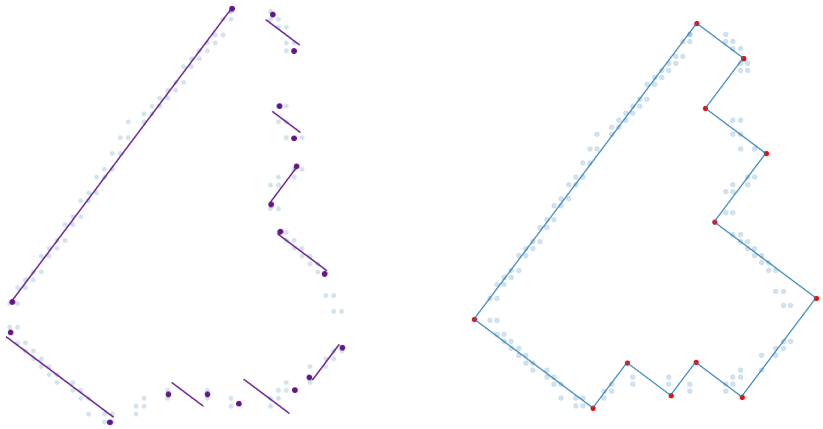
Algorithm 2: Orientation assignment

5.3.4 Create neighborhoods

At this point, each line segment belongs to an orientation class, and the parallelity or orthogonality of each line to the orientation class's main line is known. However, the spatial location of each line has not been considered yet and this must be performed in the current step. Clusters of neighboring lines are created within each orientation class. These clusters make it possible to identify lines which may be better placed in another orientation class. The assumption underlying this step is that it is unlikely for a line k of the orientation class x to be surrounded by lines of the orientation class y. In this case, line k will be assigned to orientation class y.

5.3.5 Update line orientation

Finally, the lines are adjusted to their orientation class with regard to each line's parallelity or orthogonality. The result of such an adjustment is shown in Figure 5.6a. The figure illustrates a single orientation of class and lines, which have been adjusted either parallel or orthogonal to the orientation class. To create the final building outline, all that remains is to fill in the gaps between line segments.



(A) Adjusted lines

(B) Final building outline

FIGURE 5.6 Building outline. Left panel: adjusted lines; right panel: final outline with gaps filled in.

6 Theoretical and Experimental Results

6.1 Training data

6.1.1 Airtiler - A data set generation tool

For the training, we wanted to use publicly and freely available data. Not only do high-resolution orthophotos cost a fair deal; we also wanted to ensure that other researchers would be able to replicate our study. We thus used OpenStreetMap to obtain the vector data and Microsoft Bing Maps for the imagery.

A data set consisting of satellite imagery and images of the ground truths can be created using the Python module Airtiler [6]. This tool was developed by the author during this master thesis. It allows for configuring one or more bounding boxes, together with several other options such as zoom level and OpenStreetMap attributes. Listing 6.1 shows a sample configuration used by Airtiler.

```
{
  "options": {
    "target_dir": "",
    "zoom_levels": [18,19],
    "separate_instances": false
  },
  "query": {
    "tags": [
        "highway",
        "building",
        "leisure=swimming_pool",
        "sport=tennis",
        "landuse=vineyard"
    ]
  },
  "boundingboxes": {
    "giswil_1": [
        8.1566193073,
        46.7783248574,
        8.1681635349,
        46.7905387509
    ],
    "giswil_2": [
        8.1388001434,
        46.7778439103,
        8.1562840923,
        46.7880064196
    ],
    "goldach_rorschacherberg": [
        9.440673825,
        47.4534664711,
        9.5168056457,
        47.4896115582
    ]
  }
}
```

Listing 6.1 Sample configuration for Airtiler

In the next step, the configured bounding boxes are iterated, and for each box,

all tiles according to the tile map service (TMS) specification [19] are processed. This includes the download of the corresponding orthophoto from Bing Maps and the creation of at least one binary image. The binary image contains all ground-truth instances for the current class – such as building or highway – that was specified in the configuration. This step creates one satellite image and one or more binary images per tile. If, for the current tile, no ground truths according to the configuration are found, no images for this tile are created. Figure 6.1 shows the output of a single tile, generated by Airtiler. An important point is the file names. These names are generated from the content of the file, which makes the loading and interpretation of the data set rather simple. Image names include the file extension *.tiff* whereas the masks have the extension *.tif*.

(A) 18_138079_170377.tiff (B) 18_138079_170377_building.tif

(C) 18_138079_170377_highway.tif (D) 18_138079_170377_pool.tif

FIGURE 6.1 Output of Airtiler for the TMS tile (x,y)=(138079,170377) at zoom level 18.

6.1.2 Publicly available data sets

Several data sets are publicly available from several source ([9], [10], [11], [12], [13], [14]).

6.2 Mapping Challenge

At the time of writing, the platform crowdAI hosted a challenge called Mapping Challenge [20], which focused on detecting buildings from satellite imagery. To gain additional knowledge regarding the performance of Mask R-CNN, we decided to participate in the challenge. Table 6.1 shows the changes made to the Mask R-CNN config and their impact on the prediction accuracy.

# Epochs	# Steps / Epoch	# Validation Steps	Config Change	AP@0.5	AR@0.5
100	2500	150	-	0.798	0.566
100	2500	150	Image mean RGB updated	0.799	0.564
100	2500	150	Mini mask disabled	0.807	0.573
100	5000	200	+ validation steps	0.821	0.599
100	10000	200	+ steps / epoch	0.833	0.619
100	20000	300	+ Validation steps, + steps / epoch	0.853	0.885

TABLE 6.1 Mapping challenge results

Generally, the results indicated that fine-tuning the hyperparameters improved the accuracy and that the impact was intensified merely through longer training. However, in longer training it must be ensured that the network does not overfit. In the case of an existing architecture, such as Mask R-CNN, this precaution has already been applied. If one develops a new architecture, overfitting must be taken care of, for example using a technique called Dropout [21]. Generally, Dropout randomly disables some units during the training. As a result, the model constantly changes and such change makes overfitting less likely.

6.3 Microsoft COCO Annotation Format

For the crowdAI Mapping Challenge [20], the instances were represented in Microsoft COCO annotation format [22]. Unfortunately, using this format, it is impossible to represent polygons that have holes in them[1]. Because there were many buildings that could require such representation, we decided not to use this format and instead used images for the representation of the ground truths.

(A) A building with multiple holes (B) Predicted building masks

FIGURE 6.2 The corresponding ground truth

[1] https://github.com/cocodataset/cocoapi/issues/153,21.05.2018

7 Practical Results

7.1 QGIS Plugin

A QGIS plugin was created in this research. It allows the processing of currently displayed imagery in a corresponding backend server. Figure 7.1 shows an image of the imagery layer overlaid with the changes generated by the backend. The red dots on the upper left indicate deleted objects – that is, buildings which were not predicted in the image but existed at the same location in the cadastral survey data. However, in this case the neural network was incorrect because the actual buildings can be seen clearly. Additionally, in the lower middle area, a blue region can be seen which indicates a change in the survey data. A prediction is classified as a change as soon as it fully covers at least one existing object in the survey data. Figure 7.2 shows the same changes on the cadastral data layer, and Figure 7.3 shows the predictions returned by the neural network.

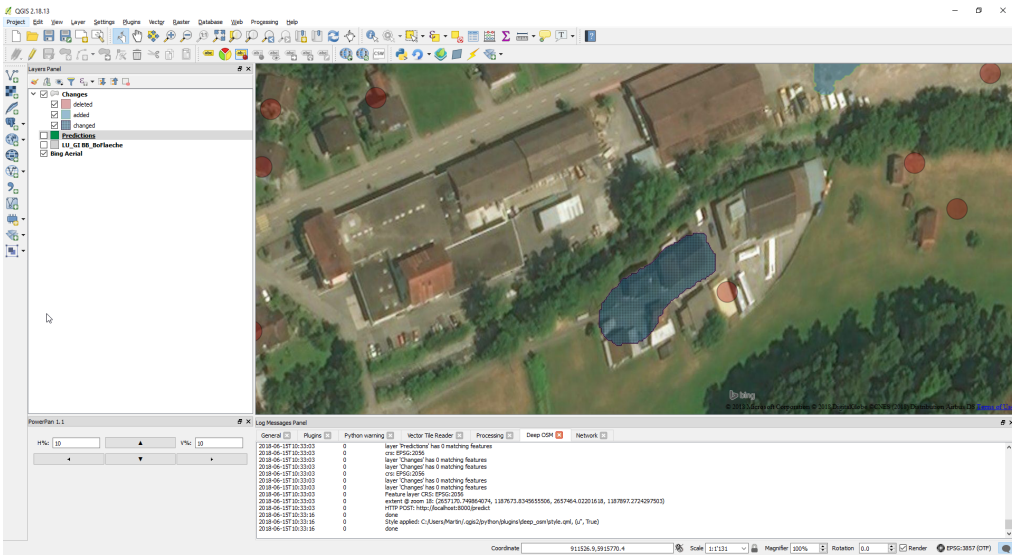


FIGURE 7.1 Changes in QGIS

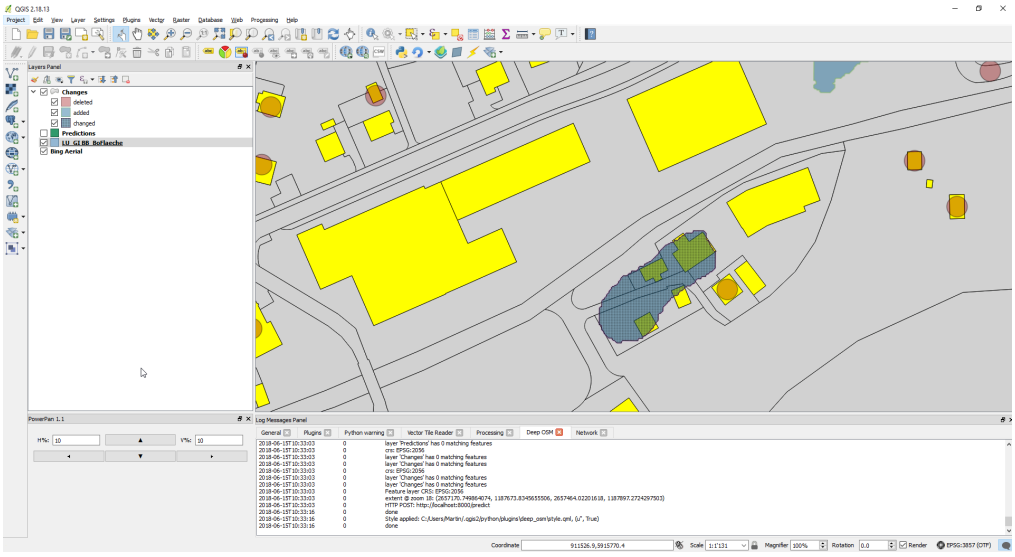


FIGURE 7.2 Changes on cadastral survey data layer

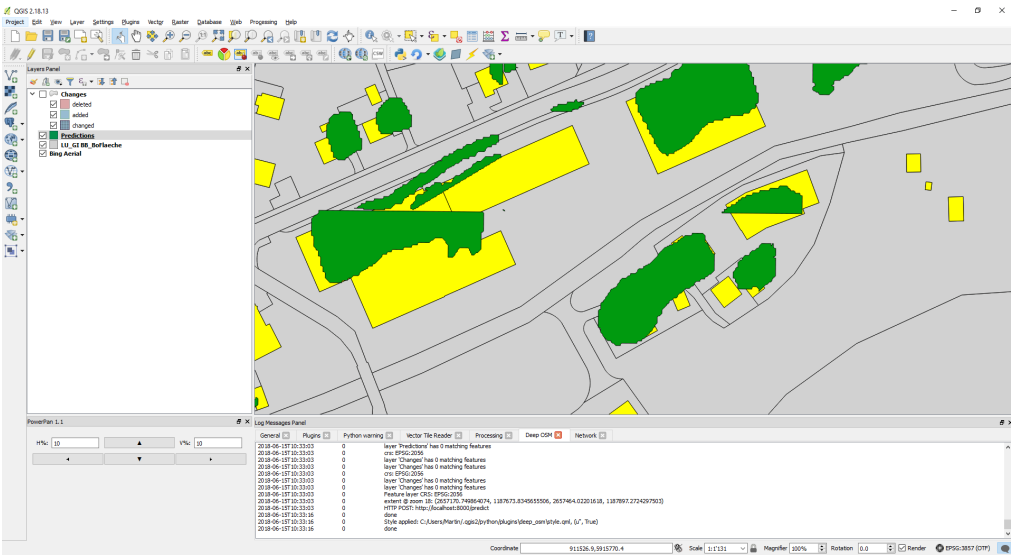


FIGURE 7.3 Predictions (green) generated by the neural network

FIGURE 7.4 Prediction attributes show the predicted class (here, buildings)

FIGURE 7.5 Changes have attributes showing the predicted class and type of change (added, deleted, or changed).

7.2 Prediction Accuracy

The accuracy of predictions for objects on orthophotos is measured using a method called "intersection over union" (IoU), also known as the Jaccard coefficient [23]. The coefficient is a measure of similarity between objects and is calculated as shown in Figure 7.6.

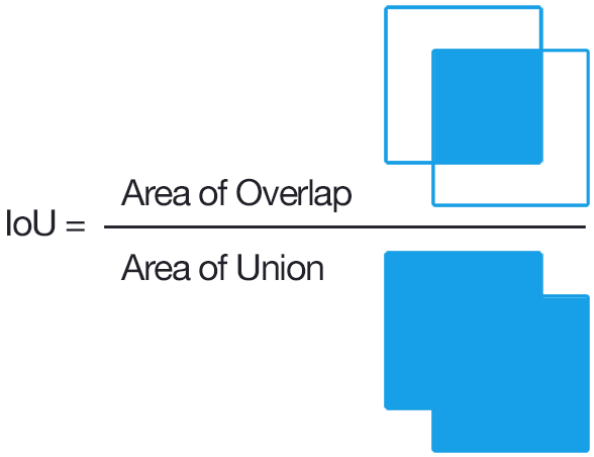


FIGURE 7.6 The calculation of intersection over union (IoU)
Source: https://www.pyimagesearch.com/2016/11/07/intersection-over-union-iou-for-object-detection/ (23.06.2018)

However, because of its non-differentiability, the IoU cannot directly be used as a loss-coefficient during the training of the neural network. Nonetheless, there are options for using IoU during training, as reported in literature ([24], [25]). The goal of this thesis was not to obtain the most accurate predictions but rather to reduce the false positives and false negatives as much as possible. Hence, it did not matter so much if the prediction was highly accurate but rather that all objects for their corresponding classes should be found. We therefore introduced a new metric, hit rate, which simply counted whether an object was found (hit) or not. It was calculated as follows:

$$Precision = \frac{|TP|}{|TP| + |FP|} \tag{7.1}$$

and

$$Recall = \frac{|TP|}{|TP| + |FN|} \tag{7.2}$$

where:

TP: True positive prediction

FP: False positive prediction

FN: False negative prediction

Finally, accordingly to this metrics, our predictions had a **Precision of 95.33%** and a **Recall of 88.96%**. These values were obtained using a randomly selected batch of 150 images from the test data set.

8 Conclusion and Future Work

8.1 Training Data

A more accurate and balanced data set would probably lead to higher prediction quality. Using images with higher resolution might even enable the neural network to learn the differences between asphalt and gravel roads. Additionally, the predicted masks would probably also be more accurate and fewer wrong predictions would occur. Generally, there might be an increase in the overall quality of predictions.

Additionally, instead of training with a data set consisting of images from a high zoom level (like 18 or 19), using images from a lower zoom level (16 or 17) might result in overall faster prediction as a larger area could be covered at once. Because only the actual changes are of interest, not the predicted contours themselves, it would not be problematic that in this case the prediction accuracy would suffer slightly through a reduced detail lever per image.

8.2 QGIS Plugin

Instead of sending an image and corresponding vectors to the backend, an option could be to send two images – an old and a new one – to the backend. All changes can then be derived from the differences in the predictions of those two images. However, often access to old and updated imagery is lacking, which is a problem for the use case described in the introduction.

Another improvement would be to give the user the option to manually choose a perimeter. The perimeter would then be automatically processed, although it might take a longer time.

Appendices

List of Figures

List of Tables

API

```yaml
swagger: "2.0"
info:
  description: Change Detector from Image to Vector
  version: "1.0.0"
  title: Change Detector from Image to Vector
  contact:
    name: Github repository
    url: https://github.com/mnboos/osm-instance-segmentation
  license:
    name: GNU General Public License v3.0
    url: https://www.gnu.org/licenses/gpl-3.0.en.html
host: localhost
basePath: /
schemes:
- http
paths:
  /predict:
    post:
      consumes:
      - application/json
      produces:
      - application/json
      parameters:
      - in: body
        name: body
        required: true
        schema:
          $ref: "#/definitions/InferenceRequest"
      responses:
        200:
```

```
            description: successful operation
            schema:
              $ref: "#/definitions/InferenceResponse"
definitions:
  InferenceRequest:
    type: "object"
    properties:
      rectangularize:
        type: "boolean"
      x_min:
        type: "number"
        description: Minimum longitude of the extent that is sent
      x_max:
        type: "number"
        description: Maximum longitude of the extent that is sent
      y_min:
        type: "number"
        description: Minimum latitude of the extent that is sent
      y_max:
        type: "number"
        description: Maximum latitude of the extent that is sent
      image_data:
        type: "string"
        description: Base64 encoded image that shall be predicted
      reference_features:
        type: "array"
        items:
          type: "string"
          description: WKT encoded features
  InferenceResponse:
    type: "object"
    properties:
      features:
        type: "array"
        items:
          type: "string"
          description: GeoJSON encoded predictions
      deleted:
        type: "array"
        items:
```

```
      type: "string"
      description: GeoJSON encoded objects that were deleted
  added:
    type: "array"
    items:
      type: "string"
      description: GeoJSON encoded objects that were added
  changed:
    type: "array"
    items:
      type: "string"
      description: GeoJSON encoded objects that were changed
```

Bibliography

[1] T.-Y. Lin, M. Maire, S. Belongie, L. Bourdev, R. Girshick, J. Hays, P. Perona, D. Ramanan, C. L. Zitnick, and P. Dollár, *Microsoft coco: Common objects in context*, 2014. [Online]. Available: http://arxiv.org/pdf/1405.0312.

[2] K. He, G. Gkioxari, P. Dollár, and R. Girshick, *Mask r-cnn*, 2017. [Online]. Available: http://arxiv.org/pdf/1703.06870.

[3] S. Ren, K. He, R. Girshick, and J. Sun, *Faster r-cnn: Towards real-time object detection with region proposal networks*, 2015. [Online]. Available: http://arxiv.org/pdf/1506.01497.

[4] O. Ronneberger, P. Fischer, and T. Brox, *U-net: Convolutional networks for biomedical image segmentation*, 2015. [Online]. Available: http://arxiv.org/pdf/1505.04597.

[5] D. Engelbart, „Augmenting human intellect: A conceptual framework“, 1962.

[6] Martin Boos, *Airtiler*. [Online]. Available: https://github.com/mnboos/airtiler (visited on 05/15/2018).

[7] *Mopublic*, 2017-06-16. [Online]. Available: https://www.cadastre.ch/content/cadastre-internet/de/manual-av/service/mopublic/_jcr_content/contentPar/tabs_copy_copy/items/dokumente/tabPar/downloadlist/downloadItems/102_1472647336994.download/Weisungen-MOpublic-de.pdf (visited on 06/22/2018).

[8] A. Krizhevsky, I. Sutskever, and G. E. Hinton, „Imagenet classification with deep convolutional neural networks“, 2012, pp. 1097–1105. [Online]. Available: http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf.

[9] Volodymyr Mnih, „Machine learning for aerial image labeling“, Dissertation, 2013. [Online]. Available: https://www.cs.toronto.edu/~vmnih/data/ (visited on 01/06/2018).

[10] *Spacenet on amazon web services (aws)*, 2018-04-30. [Online]. Available: https://spacenetchallenge.github.io/datasets/datasetHomePage.html (visited on 05/15/2018).

[11] International Society for Photogrammetry and Remote Sensing, *Vaihingen*. [Online]. Available: http://www2.isprs.org/commissions/comm3/wg4/2d-sem-label-vaihingen.html (visited on 05/15/2018).

[12] International Society for Photogrammetry and Remote Sensing, *Potsdam*. [Online]. Available: http://www2.isprs.org/commissions/comm3/wg4/2d-sem-label-potsdam.html (visited on 05/15/2018).

[13] P. Helber, B. Bischke, A. Dengel, and D. Borth, *Eurosat: A novel dataset and deep learning benchmark for land use and land cover classification*, 2017. [Online]. Available: http://arxiv.org/pdf/1709.00029.

[14] *Deepsat*. [Online]. Available: http://csc.lsu.edu/~saikat/deepsat/ (visited on 05/15/2018).

[15] T. Partovi, R. Bahmanyar, T. Kraus, and P. Reinartz, „Building outline extraction using a heuristic approach based on generalization of line segments“, *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 10, no. 3, pp. 933–947, 2017, ISSN: 1939-1404. DOI: 10.1109/JSTARS.2016.2611861.

[16] C. Maple, „Geometric design and space planning using the marching squares and marching cube algorithms“, in *2003 international conference on geometric modeling and graphics*, E. e. Banissi and M. Sarfraz, Eds., IEEE Comput. Soc, 2003, pp. 90–95, ISBN: 0-7695-1985-7. DOI: 10.1109/GMAG.2003.1219671.

[17] D. H. Douglas and T. K. Peucker, „Algorithms for the reduction of the number of points required to represent a digitized line or its caricature“, *Cartographica: The International Journal for Geographic Information and Geovisualization*, vol. 10, no. 2, pp. 112–122, 1973, ISSN: 0317-7173. DOI: 10.3138/FM57-6770-U75U-7727.

[18] R. O. Duda and P. E. Hart, „Use of the hough transformation to detect lines and curves in pictures“, *Communications of the ACM*, vol. 15, no. 1, pp. 11–15, 1972, ISSN: 0001-0782. DOI: 10.1145/361237.361242. [Online]. Available: http://dl.acm.org/ft_gateway.cfm?id=361242&type=pdf.

[19] *Tile map service specification*. [Online]. Available: https://wiki.osgeo.org/wiki/Tile_Map_Service_Specification (visited on 06/12/2018).

[20] crowdAI, *Mapping challenge*. [Online]. Available: https://www.crowdai.org/challenges/mapping-challenge (visited on 04/29/2018).

[21] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, „Dropout: A simple way to prevent neural networks from overfitting“, *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014, ISSN: 1532-4435. [Online]. Available: http://dl.acm.org/ft_gateway.cfm?id=2670313&type=pdf.

[22] *Coco data format: Common objects in context*. [Online]. Available: http://cocodataset.org/#format-data (visited on 05/15/2018).

[23] B. Liu, *Web Data Mining: Exploring Hyperlinks, Contents, and Usage Data*, 2nd ed. Berlin / Heidelberg: Springer-Verlag, 2011, ISBN: 9783642194597.

[24] G. Bebis, R. Boyle, B. Parvin, D. Koracin, F. Porikli, S. Skaff, A. Entezari, J. Min, D. Iwai, A. Sadagic, C. Scheidegger, T. Isenberg, M. A. Rahman, and Y. Wang, Eds., *Optimizing Intersection-Over-Union in Deep Neural Networks for Image Segmentation: Advances in Visual Computing*, Springer International Publishing, 2016, ISBN: 978-3-319-50835-1.

[25] J. Yu, Y. Jiang, Z. Wang, Z. Cao, and T. Huang, *Unitbox: An advanced object detection network*, 2016. DOI: 10.1145/2964284.2967274. [Online]. Available: http://arxiv.org/pdf/1608.01471.