Master Thesis

Image Segmentation using Convolutional Neural Networks for Change Detection of Landcover

Deep Learning

Martin Boos

2. June 2018

HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL

FHO Fachhochschule Ostschweiz

This page is intentionally left blank.

Abstract

tbd

Acknowledgments

Put your acknowledgments here.

Prof. Stefan Keller tbd

Contents

Examples 1

This chapter is to demonstrate some of the capabilities of this LaTeX template. Please take a good look at this chapter and try to follow the guidelines.

Goal of this chapter

1.1 Equations

1.1.1 some equations

Equations can easily be written using the *Equation* environment. Inline equations are inserted with $\sqrt{-1} = i$. Equations can also be labeled, so it is possible to reference them. This should be done for all important equations.

Equations

$$x^2 + y^2 = 1 \tag{1.1}$$

There are several environments for multi line equations. A very useful one is *align*, see equation (1.4).

$$\oint \vec{E} \cdot d\vec{A} = \frac{q}{\epsilon_0} \tag{1.2}$$

$$\oint \vec{B} \cdot d\vec{A} = 0 \tag{1.3}$$

$$\oint \vec{E} \cdot d\vec{s} = -\frac{d\Phi_B}{dt} \tag{1.4}$$

Images are always inserted inside a *figure* environment. If possible, it is advisable to use `[tb]` as position. Always remember to add a caption and a label, so you can reference the image like this: Figure 1.1. If possible, images

Figures and Tables

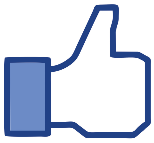


FIGURE 1.1 An example image

some	text	is shown	here
there is more	text here	and here	cool.
and	even	more	here.

TABLE 1.1 A sample table

should be inserted as vector graphics, e.g. `eps` or `pdf` - or even drawn manually in TikZ.

Tables can be used quite similarly. They are inserted inside a *table* environment, as shown in Table 1.1.

Another useful tool is the *tabularx* environment. It lets the user specify the total width of the table, instead of each column. An example is shown in Table 1.2.

Please read the documentation of the *booktabs*[1] package to find information on how to create good tables. Always remember the first two guidelines and try also to stick to the other three:

1. Never, ever use vertical rules.
2. Never use double rules.
3. Put the units in the column heading (not in the body of the table).
4. Always precede a decimal point by a digit; thus 0.1 *not* just .1.
5. Do not use „ditto“ signs to repeate a value. In many circumstances a blank will serve just as well. If it won't, then repeat the value.

PARAGRAPHS Note that each paragraph is ended with an empty line. *Never* use \\ to end paragraphs – this is a new line, not a new paragraph. Also try to keep your source code clean: about 80 characters per line. Using source control makes your life much easier

QUOTES Quotes can easily be made using the „csquotes“ package. Citing text passages is

[1] `http://www.ctan.org/pkg/booktabs`

some	text	is shown here
there is more and	text here even	and here. more here.

Table 1.2 Tabularx example

easily done: „First argument: citation, second argument: terminal punctuation!“ (me) Whole block quotes are also easily possible.

> Formal requirements in academic writing frequently demand that quotations be embedded in the text if they are short but set oV as a distinct and typically indented paragraph, a so-called block quotation, if they are longer than a certain number of lines or words. In the latter case no quotation marks are inserted.

Any numbers and units should by typed using the siunitx package. Numbers are written as 1234 3.45×10^{-5} 1, 2 and 4 4 to 30 10° 5°3′2″. Units are written with $\mathrm{kg\,m/s^2}$ or $14\,\mathrm{F}^4$. Of course also possible is 10 m, 40 m and 12 m or −40 °C to 125 °C Almost every unit you could possibly think of is implemented! SI Units

The bibliography is created using *Bibtex*. The standard format is set to `ieeetr`, which is the IEEE Standard. There are example entries for different types of in the separate bibliography file **article book booklet conference inbook incollection manual mastersthesis misc phdthesis proceedings techreport unpublished**. Bibliography

All glossary entries are made in the separate file `glossary.tex`. They can then be used with Equation. Acronyms are defined as shown there and used similarly. The first time, it will be *support vector machine (SVM)*. The second time: *SVM*. The glossary has to be created manually by invoking `makeindex -s doku.ist -t doku.glg -o doku.gls doku.glo`. The index is simply created by using `index{text}`. It is generated automatically. Index & Glossary

1.1.2 Listings

Listings are created by the `lstlistings` package.

This is a simple ToDo note ToDo

This is a small note [Citation needed] ToDo

ToDo

Management Summary 2

Overview 3

3.1 Was ist in welchem Kapitel etc.

4 State of the art

There are various different techniques and technologies which allow to build neural networks for machine learning purposes. Due to the increasing computation power of current hardware, especially graphic cards, more complex and therefore computationally heavy models can be built. As a result of this, the active research leads to a number of improvements of existing networks or completely new networks each year.

At the time of this writing, the most promising candidate is Mask R-CNN [1], which is based on Faster R-CNN [2]. Another well performing architecture can be found with U-Net [3] that has been created for medical purposes. Nonetheless, it can also be used for semantic segmentation of all kind of data.

One big challenge in the area of computer vision is instance segmentation, which not only tries to detect occurences of a specific class, but also tries to find single instances of it. In other words, it tries to find the contour. In order to compare various architectures built for such a purpose, there are different challenges published on the internet. One of those leaderboards is from the Microsoft COCO challenge [4], which yearly tries to find the most competitive candidates for different tasks, such as object detection, instance segmentation or keypoint detection.

Introduction 5

5.1 Technical Background - PNF / AV

tbd

5.2 Architecture

Figure 5.1 shows an overview of the architecture. Note, that this diagram shows a combination of the learning and the prediction architecture. Step 2, which is loading the satellite imagery from Microsoft Bing is only required in the learning phase, that is for the generation of the training data, which is required to train the neural network.

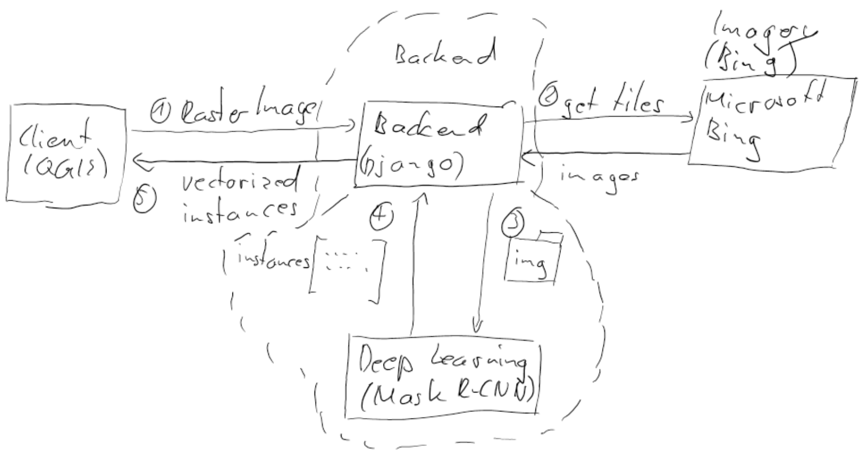


FIGURE 5.1 Architecture overview

Figure 5.2 shows the data flow during the training of the neural network. For this step, satellite imagery from Microsoft Bing maps as well as OpenStreetMap (OSM) data is used. The satellite imagery is downloaded tile per tile for a predefined boundingn box and zoom-level and at the same time, binary images are created from the OSM data, which represent the ground truth. To simplify this step and make it available to the public, a tool called Airtiler [5] has been created and published.

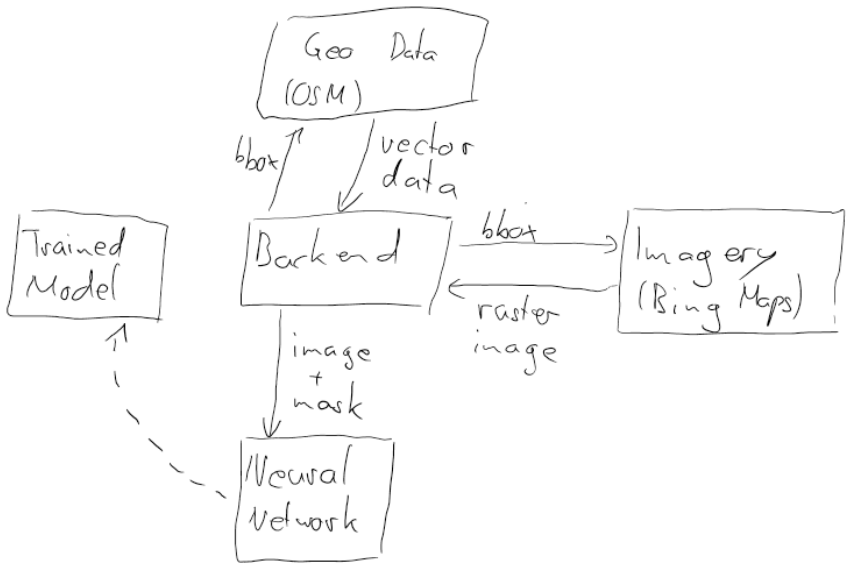


FIGURE 5.2 Learning phase

As soon as the training is completed, the prediction can be done. For this thesis, this has been split into two phases. Figure 5.3 shows the data flow of phase 1, which passes 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 is then used, to predict all instances on the current image. In the next step, the predicted instances are georeferenced using the boundingbox information that was sent to the backend at the beginning of the prediction phase. Finally, the georeferenced instances are sent back to the frontend client (the QGIS plugin), which then visualizes the data on a new layer in QGIS.

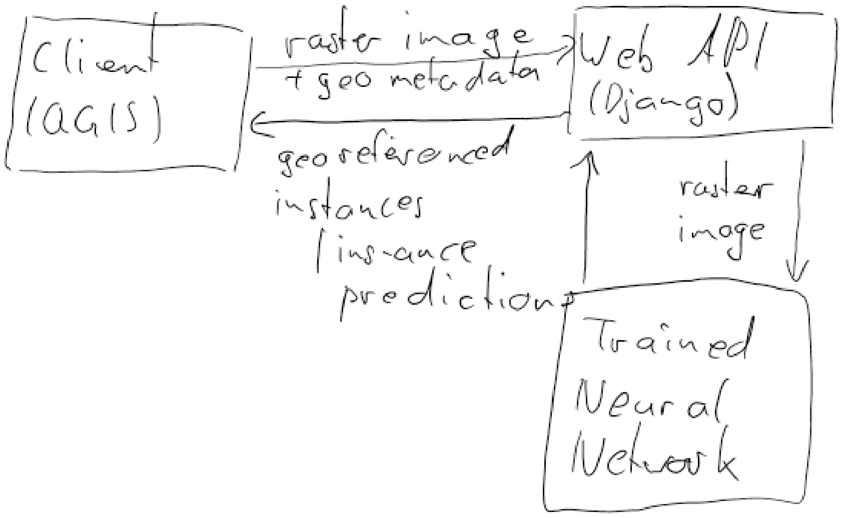


FIGURE 5.3 Prediction phase 1

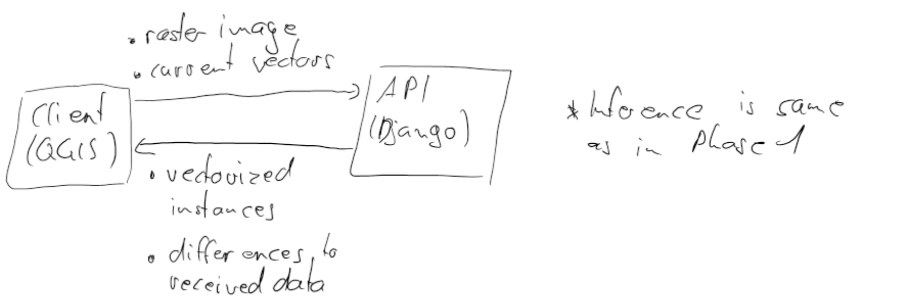


FIGURE 5.4 Prediction phase 2

Image Segmentation with Convolutional Neural Networks (CNN) 6

6.1 Introduction

With the increasing computational power that comes with recent graphic cards, increasingly complex neural networks can be used to achieve more challenging tasks than before. Such tasks can be object detection, where the goal is to determine whether an object of a specified class (for example 'car') is visible on a image.

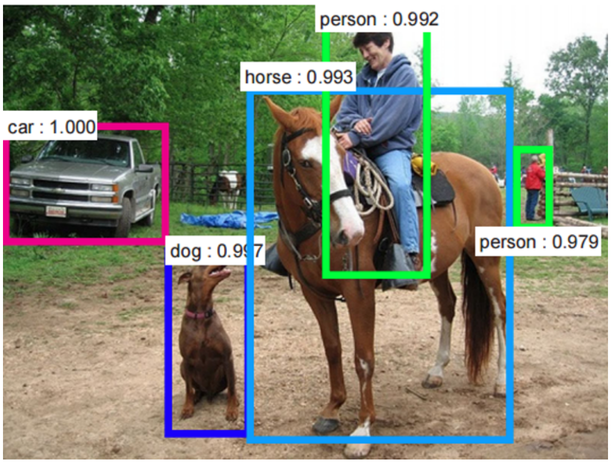


FIGURE 6.1 An example of object detection, showing the detected classes and the confidence of each prediction.
Source: https://dius.com.au/2016/12/06/the-cleverness-of-deep-learning/ (27.05.2018)

In contrast to object detection, the target of image segmentation is not only to state whether an object could be found, but to label each pixel of an image with a class. The Different types of image segmentation can be seen in Figure 6.2.

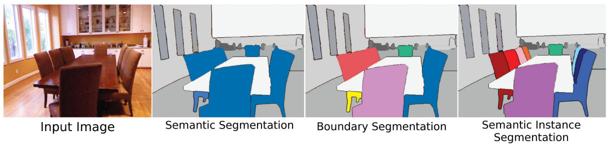


Figure 6.2 Different types of image segmentation.
Source: https://i.stack.imgur.com/mPFUo.jpg (27.05.2018)

6.2 Convolutional Layer

A convolutional layer is one of the most important and basic building blocks of a neural network. It has a number of 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 being moved over the input image and at each position of the filters on the image, a convolution is being computed, which is an element wise matrix multiplication and a sum over the resulting matrix. The result of this operation is an activation map, which is also the output of the convolutional layer.

Obviously, 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.

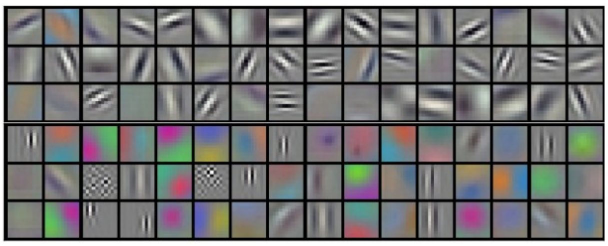


Figure 6.3 Example filters learned by [6].
Source: http://cs231n.github.io/assets/cnn/weights.jpeg (27.05.2018)

6.3 Pooling Layer

6.3.1 Min / Max Pooling

6.3.2 Unpooling

6.4 Fully Connected Layer

6.5 Mask R-CNN

7 Theoretical and Practical Challenges

7.1 Training data generation

In order to train the neural network a data set was required, which could then be used for training and validation. Due to this, a tool [5] has been developed which uses publicly available vector data from OpenStreetMap and satellite imagery Microsoft Bing.

wir wissen, dass es immer besser aufgelöste Bilddaten einzelner Kantone gibt. Zum Zeitpunkt unserer Arbeit waren diese aber weniger verfügbar, was ein Problem ist, weil für das Training eine möglichst grosse Datenmenge (ideal > 100'000 Bilder) benötigt wird

tbd

7.2 Prediction accuracy

7.2.1 Class probability

After the first training the neural network, the results were not quite as expected. Even though, buildings were predicted as buildings in most cases, other classes, like tennis courts, were predicted as buildings as well. Due to this, the network has been retrained with the additional, incorrectly predicted, classes like tennis courts. However, instead of correctly making a distinction between buildings and tennis courts, the overall prediction accuracy got worse. This might be the result of the network which has to solve a more complex task now, by deciding which class it is, instead of a simple yes-no decision. Additionally, the training data is highly imbalanced, as there are lot more samples of buildings than tennis courts. As a result of this, a solution could be to train the network several times seperatly, to get multiple models, each trained for a specific class. Another solution could be to weight the loss according to the relative amount of the specific class according to the size of the whole dataset.

7.2.2 Outline

The predictions are in most cases a bit to small when compared to the corresponding orthophoto. This might be the result of slightly misaligned masks, since the masks and the images are generated seperately. tbd: bild einfügen

7.3 Building outline regularization

Once the network has been trained, it can be used to make predictions on images, it has never "seen" before. However, there are situations in which the predictions are far from perfect, especially then if the building is partially covered from trees or has unclear outlines. This can be seen in Figure 7.1.

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

FIGURE 7.1 Predicted masks and actual ground truth

As a result of the inaccuracies in the predicted building masks the contours of these predictions can not directly be used to create the vectorized outlines. Instead, the predictions have to be regularized. The approach used is similar to [7] and described in Figure 7.2.

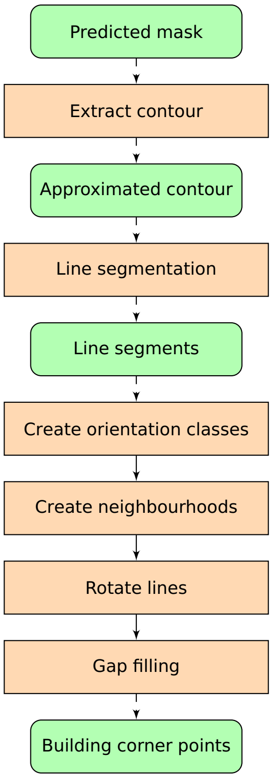


FIGURE 7.2 Rectangularization procedure

The single steps are described in detail in the following sections.

7.3.1 Contour extraction

The first step of the building outline regularization procedure consists of getting the contour from the predicted mask, which covers the whole building. The extraction is done using the marching squares algorithm [8]. 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, which is derived from the content of its cells, according to (7.1). The cells are traversed in counter clockwise order.

$$
\begin{aligned}
s &= \sum_{i=0}^{3} 2^i f(c_i) \\
&= 2^0 * f(c_0) + 2^1 * f(c_1) + 2^2 * f(c_2) + 2^3 * f(c_3) \\
&= f(c_{bottomLeft}) + 2 * f(c_{bottomRight}) + 4 * f(c_{topRight}) + 8 * f(c_{topLeft})
\end{aligned}
\tag{7.1}
$$

where:

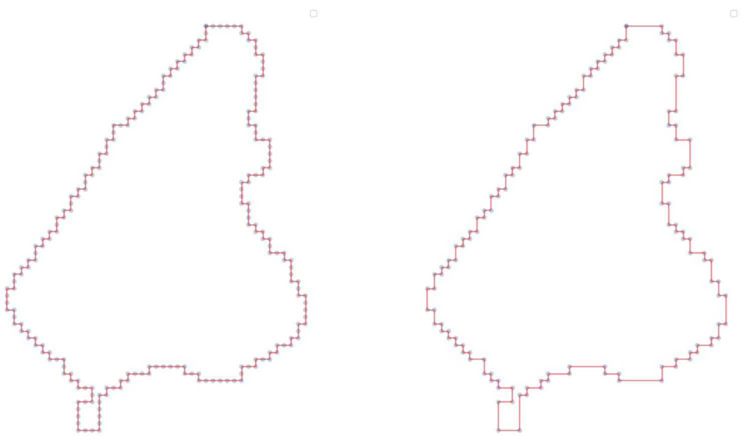
c_i: The value of the cell i

c_0: The bottom left cell

and

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

As soon as the contour has been extracted, its number of points will be reduced using a Douglas-Peucker algorithm [9]. 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, even though, the startpoint and endpoint of each such line would be enough, to represent the line. Additionally, the lower the number of points per contour is, the faster the following processing will be.



(A) Original contour (B) Approximated contour

FIGURE 7.3 Contour before and after approximation

7.3.2 Line segmentation

Once the contour has been extracted, it is split into multiple line segments. For this, the main direction of the building is determined using the Hough Transformation [10]. The result of the Hough Transformation is an datastructure, which contains an angle, a distance and a number. The combination of the angle and the distance, from a predefined reference point, lead to a line. The number is the number of points which lie on the constructed line. Therefore, this algorithm can be used to detect the main building orientation, i.e. the longest contour-line of any orientation. The angle of the found line, is called the main building orientation.

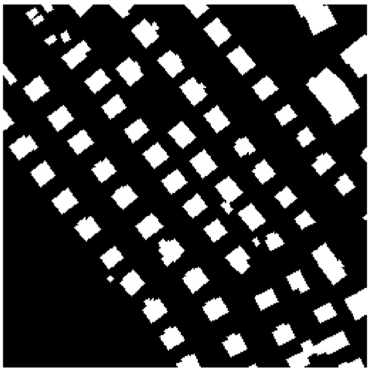
Once the main building orientation is known, the line segmentation starts at the point which has the smallest distance to this line. The whole procedure is depicted in algorithm 1.

```
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
```

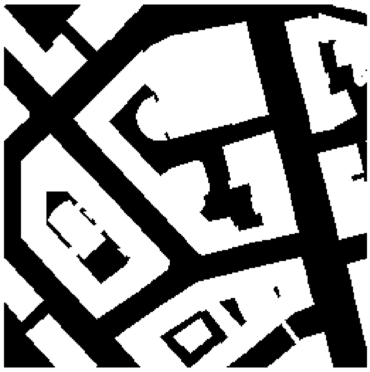
Algorithm 1: Line segmentation

7.3.3 Create orientation classes

After the line segmentation, an orientation will be assigned to each line. Generally, most of the buildings consist of mainly right angles. However, there are still buildings, for which this assumption is not true. Due to this, orthogonality will be preferred, but other angles will still be possible. Algorithm 2 shows the procedure, which assigns a main orientation class to each line and defines the lines parallelity / orthogonality to the longest line of the same orientation class.



(A) Regular buildings



(B) Non-orthogonal buildings

FIGURE 7.4 Different kinds of buildings with regard to their 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 ≤ angleThreshold then
            li.orientation = line.orientation li.orthogonal =
              line.orthogonalTo(line)
        end
    end
end
```

Algorithm 2: Orientation assignment

7.3.4 Create neighbourhoods

At this point, each line segment belongs to a orientation class and the parallelity / orthogonality of each line to the orientation classes main line is known. However, the spatial location of each line has not been taken into account yet, which is done in this step. Clusters of neighbouring lines are created within each orientation class. As a result of this, it is now possible to find lines, which may be better placed in an other orientation class. This is based in the assumption, that it is unprobable, that a line k of the orientation class x is surrounded by lines of the orientation class y. In this case, the line k will be assigned to the orientation class y.

7.3.5 Update line orientation

Finally, the lines will be adjusted to their orientation class with respect to each lines parallelity / orthogonality. The result of such an adjustment, can be seen in Figure 7.5, which shows a single orientation class and lines, which have been adjusted either parallel or orthogonal to the orientation class.

FIGURE 7.5 Adjusted lines

7.3.6 Gap filling

In order to create the final building outline, the only thing left to do is to fill the gaps between the line segments.

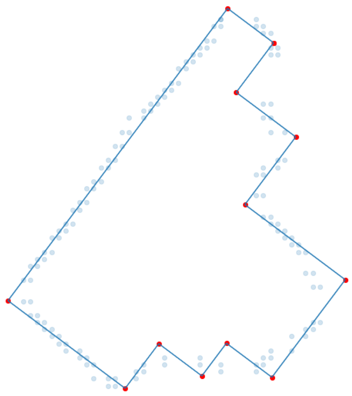


Figure 7.6 Final building outline

8 Theoretical and Experimental Results

8.1 Training data

For the training, we wanted to use publicly and freely available data, which lead to Open Street Map for the vector data and Microsoft Bing Maps for the imagery. A dataset consisting of satellite imagery and images for the ground truths can be created using the Python module Airtiler [5].

Furthermore, there are several different datasets publicly available: [11], [12], [13], [14], [15], [16].

8.2 Microsoft COCO

For the crowdAI Mapping Challenge [17] the instances were represented in the Microsoft COCO annotation format [18]. Unfortunately, using this format, it is not possible to represent polygons with holes in it[1]. As there are many buildings, for which this would be required, we decided not to use this format and instead use images for the representation of the ground truths.

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

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

FIGURE 8.1 The corresponding ground truth

8.3 Building detection

tbd

8.4 Mapping Challenge

At the time of this writin the platform crowdAI hosted a challenge called Mapping Challenge [17] which was about detecting buildings from satellite imagery. In order to gain additional knowledge regarding the performance of Mask R-CNN, we decided to participate in the challenge.

Table 8.1 shows the changes made to the Mask R-CNN config and their impact on the prediction accuracy.

Generally, these results indicate, that finetuning of hyperparameters has an impact. Furthermore, the even bigger impact can be made just by longer training. However, in case of longer training, one has to make sure, that the network will not overfit. In the case of an already existing architecture like Mask R-CNN for example, this has already been done. On the other hand, if one develops a new architecture, overfitting has to be taken care of, for example with a technique called Dropout [19]. Generally, Dropout randomly disables some units during the training. As a result of this, the model constantly changes and overfitting can not happen that easily.

# 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 8.1 Mapping challenge results

Practical Results 9

A QGIS plugin has been created, which allows to process the currently displayed imagery in a corresponding backend server.

9.1 QGIS Plugin

9.1.1 Wie es von den Leuten genutzt wird

9.1.2 Effizienzsteigerung

10 Conclusion and Future Work

was könnte besser / anders sein, mit höher aufgelösten Daten? (zB. unterscheiden zwischen befestigt / unbefestigt, Umrisse besser / genauer, weniger Artefakte (false positive))

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 one does not have access to either old or up-to-date imagery, which is a problem for the use case described in the introduction.

Appendices

List of Figures

List of Tables

Bibliography

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

[2] 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.

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

[4] 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.

[5] Martin Boos, *Airtiler*. [Online]. Available: https://github.com/mnboos/airtiler.

[6] 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.

[7] 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.

[8] 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.

[9] 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.

[10] 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.

[11] Volodymyr Mnih, „Machine learning for aerial image labeling“, Dissertation, 2013. [Online]. Available: https://www.cs.toronto.edu/~vmnih/data/.

[12] *Spacenet on amazon web services (aws)*, 30.04.2018. [Online]. Available: https://spacenetchallenge.github.io/datasets/datasetHomePage.html.

[13] International Society for Photogrammetry and Remote Sensing, *Vaihingen*. [Online]. Available: http://www2.isprs.org/commissions/comm3/wg4/2d-sem-label-vaihingen.html.

[14] International Society for Photogrammetry and Remote Sensing, *Potsdam*. [Online]. Available: http://www2.isprs.org/commissions/comm3/wg4/2d-sem-label-potsdam.html.

[15] 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.

[16] *Deepsat*. [Online]. Available: http://csc.lsu.edu/~saikat/deepsat/.

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

[18] *Coco data format: Common objects in context*. [Online]. Available: http://cocodataset.org/#format-data.

[19] 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.

Index