Master Thesis

Learning City Structures from Online Maps

Author(s):
Kaiser, Pascal

Publication Date:
2016

Permanent Link:
https://doi.org/10.3929/ethz-a-010688765

Rights / License:
In Copyright - Non-Commercial Use Permitted
Master Thesis

Learning City Structures from Online Maps

Pascal Kaiser

22.03.2016

Advisors: Dr. Martin Jaggi, Dr. Aurelien Lucchi, Dr. Jan Dirk Wegner, Dr. Markus Kalisch
Supervisor: Prof. Dr. Thomas Hofmann
Department of Computer Science
ETH Zurich
Switzerland
Preface

First, I want to thank Dr. Martin Jaggi, Dr. Aurlien Lucchi and Dr. Jan Dirk Wegner for being amazing supervisors for my master thesis. I benefited very much from their knowledge and their deep understanding of neural networks and machine learning in general. I would also like to thank Prof. Thomas Hofmann for giving me the chance to write my master thesis in the Data Analytics Lab at ETH Zurich. Writing my master thesis in this dynamic environment was a great experience for me and I could improve my skills on so many levels. I also want to thank Silvano Galliani, Yannic Kilcher and Dimititrios Marmanis for all their help and support with the technical setup. I would like to thank Dr. Markus Kalisch, my supervisor from the ETH mathematics department, for giving me the chance to write my master thesis at the Data Analytics Lab. Furthermore, I would like to thank my wife Akiko for her patience and never-ending support.
Abstract

Huge amounts of remote sensing data are nowadays publicly available with applications in a wide range of areas including the automated generation of maps, change detection in biodiversity, monitoring climate change and disaster relief. On the other hand, deep learning with multi-layer neural networks, which is capable of learning complex patterns from huge datasets, has advance greatly over the last few years.

This work presents a method that uses publicly available remote sensing data to generate large and diverse new ground truth datasets, which can be used to train neural networks for the pixel-wise, semantic segmentation of aerial images.

First, new ground truth datasets for three different cities were generated consisting of very-high resolution (VHR) aerial images with ground sampling distance on the order of centimeters and corresponding pixel-wise object labels. Both, VHR aerial images and object labels are publicly available and were downloaded from online map services over the internet. Second, the three newly generated ground truth datasets were used to learn the semantic segmentation of aerial image by using fully convolutional networks (FCNs), which have been introduced recently for accurate pixel-dense semantic segmentation tasks. Third, two modifications of the base FCN architecture were found that yielded performance improvements. Fourth, an FCN model was trained on huge and diverse ground truth data of the three cities simultaneously and achieved good semantic segmentations of aerial images of a geographic region that has not been used for training.

This work shows that using publicly available remote sensing data can be used to generate new ground truth datasets that can be used to effectively train neural networks for the semantic segmentation of aerial images. Moreover, the method presented here allows to generate huge and in particular diverse ground truth datasets that enable neural networks to generalize their predictions to geographic regions that have not been used for training.
6 Experiments
  6.1 Learning from Single Cities .......................... 43
  6.2 Three Different FCN Architectures .................... 45
  6.3 Learning Simultaneously from Multiple Cities ........ 50

7 Discussion
  7.1 New Ground Truth Data ............................... 53
  7.2 Learning from Single Cities .......................... 55
  7.3 Three Different FCN Architectures .................... 56
  7.4 Learning Simultaneously from Multiple Cities ........ 58

8 Conclusions and Future Work .............................. 61

Bibliography .............................................. 63

A Alternative CNN architecture ........................... 67

B Learning Building Corners ................................ 71

C Supplementary Tables .................................... 75
Chapter 1

Introduction

Huge amounts of remote sensing data are available nowadays. Analyzing remote sensing data to extract relevant information has many applications including the automated generation of maps, city planning, predicting crop yields and disaster relief. For example, the tsunami that hit the northeastern coast of Japan in 2011 left a wage of devastation. Transporting food, medicine and other supplies to where they were needed has been seriously slowed down by collapsed buildings and damaged roads. Automatically extracting intact buildings and roads from satellite and aerial images, which have been taken after the catastrophe (see Figure 1.1), could significantly facilitate the organization of disaster relief.

It is generally difficult for humans to analyze and interpret remote sensing data manually. This is mainly because these datasets can be very large in size and patterns that occur in the data can be difficult to detect [1].

Recently, the automatic analysis of raw image data has advanced greatly; a progress which was mainly brought about by the success of deep neural networks [17], [16], [34]. Deep learning with multi-layer neural networks has become the state-of-the art for various visual learning tasks since the

![Figure 1.1: Aerial image of Japan before and after the Tohoku tsunami disaster in 2011.](image-url)
innovative work of Krizhevsky et al. [16] in 2012. Deep learning does not require an a-priory feature definition. Rather deep learning starts from raw image data and finds suitable features during the course of training, which allows for completely automated, end-to-end learning.

With the advance of deep learning, it was demonstrated that convolutional neural networks (CNNs) are especially effective in image recognition tasks [16], [34]. For most image recognition tasks, CNNs currently outperform all other methods (reviewed in Russakovsky et al. [29]). CNNs have also been applied successfully to the semantic segmentation of aerial images [24], [25], [23] and [30]. Semantic segmentation aims at labelling each pixel in an image with the most likely object class label.

However, when passing an image through a CNN, the spatial resolution of the image gets typically reduced and information about the localization of objects contained in the image gets blurred. Thus CNNs usually output coarse classification maps. To this end, a special form of CNN, so called fully convolutional networks (FCNs), were proposed by Long et al. [20]. FCNs are able to perform in-network upsampling of coarse neural network layers back to the resolution of the original image. In addition, Long et al. [20] introduce a so called skip-architecture which fuses information from shallow neural network layers with deep neural network layers to combine spatial information with semantic information to output accurate pixel-wise object label predictions. FCNs can be trained end-to-end, pixels-to-pixels from raw image data to pixel-wise object labels.

Machine learning algorithms currently use relatively small benchmark datasets to learn the semantic segmentation of aerial images [21]. The size of benchmark datasets is thus a limiting factor for the learning of the semantic segmentation of aerial images. However, it is possible to improve this situation by using more training data that can be acquired from sources that are publicly available.

Billions of remote sensing images of the earth’s surface have been acquired over the past few years as a base for earth observation and are publicly available from online map services. On the other hand, geographic coordinates of buildings and roads, which appear in aerial images, are available from online map services and can be freely downloaded over the internet too. Combining aerial images and geographic coordinates of buildings and roads is thus a way to generate new data that can be used to train neural networks for the semantic segmentation of aerial images. Moreover, online map services recently started to provide very-high resolution (VHR) aerial images with ground sampling distances on the order of centimeters. In VHR aerial images small-scale surface textures become visible, which have not been accessible in aerial images of lower resolutions. The higher resolution of aerial images might be useful for learning the semantic segmentation of aerial images because it provides finer information to the neural networks.

The first purpose of this work is to develop a method to generate new
ground truth datasets that can be used to train neural networks for the semantic segmentation of aerial images. The method proposed in this work aims to overcome the limitations posed by the relatively small size of benchmark datasets to the learning of the semantic segmentation of aerial images. To this end, this work describes a semi-automatic approach, which combines freely available VHR aerial images from Google Maps and geographic coordinates of buildings and roads from OpenStreeMap (OSM), to generate new ground truth datasets. New ground truth data in this context are VHR aerial images and corresponding pixel-wise object label files for the object labels building, road and background. This semi-automatic approach is flexible, since it can be applied to different geographic regions, and scalable, as long as the quality of the data obtained from Google Maps and OSM allows for it.

The second purpose of this work is to examine the applications and limitations of this semi-automatic approach to generate new ground truth data by training FCNs on newly generated ground truth datasets and by comparing their predictive performance to each other. To this end, three new ground truth datasets were generated for the urban areas of Chicago, Paris and Zurich, respectively. For training, a base FCN architecture was used which was described in Long et al. [20].

First, FCNs were trained on new ground truth data of each city separately and their predictive performance was compared to each other. Second, this work found two modifications of the base FCN architecture, which yielded improvements on the semantic segmentation of aerial images. Third, an FCN was trained simultaneously on new ground truth data of Chicago, Paris and Zurich and achieved semantic segmentations for the buildings of a fourth city (which was not used to train any FCN) that were better than the semantic segmentations obtained from the FCNs, which were trained on single cities.

The rest of this work is organized as follows. Chapter 2 reviews related work on semantic segmentation of aerial images. Chapter 3 presents an overview over FCNs. Chapter 4 describes the semi-automatic approach mentioned above to generate new ground truth datasets and describes the newly generated ground truth datasets for Chicago, Paris and Zurich. Chapter 5 explains the deep learning framework which was used in this work. Chapter 6 describes the predictive performances of FCNs which were trained on new ground truth datasets. Chapter 7 discusses the results and summarizes the most important findings. Chapter 8 draws conclusions and discusses what insights future work might yield.
Chapter 2

Related Work

Several methods have been developed for the semantic segmentation of aerial images over the past 30 or so years. These can be classified into two categories: non-neural network methods and neural network methods. This chapter reviews first non-neural network methods for the semantic segmentation of aerial images and then turns to neural network methods for the semantic segmentation of aerial images. At the end, it describes methods, which have been proposed so far, to generate new ground truth datasets of aerial images and pixel-wise object labels.

2.1 Non-neural network methods

It is difficult, in general, to determine the object label of a single pixel in an aerial image if only the spectral intensities of the pixel itself are considered. This is because the spatial neighbourhood of a pixel contains much information about the local context to which a pixel belongs and thus about what the pixel depicts. To include the spatial context of a pixel for determining its object label, much attention has been given to the extraction of hand-crafted local features from a pixel’s neighbourhood like local binary patterns, texture filter banks, Textons and Histograms of oriented gradients [10], [3] and [37]. These features are then used as an input to standard classifiers like support vector machines, logistic regression classifiers or random forests and used to classify each pixel or segment of an aerial image. Sirmaček et al. [35] and Senaras et al.[32] took this approach further by combining different methods to extract local image features and by fusing outputs of different classifiers into one prediction. Other works aimed at including a-priory information about the different object labels in aerial images like shape templates [15] and smoothness [31].

However, choosing suitable a-priory information and/or robust image features is not an easy task. This is because objects that appear in aerial images generally show high intra-class variability, low inter-class differences
and can be correlated to each other [26], [40] and [39]. In very-high resolution (VHR) aerial images, these issues might even be more pronounced because small-scale surface textures become visible that are not accessible in aerial images of lower resolutions.

2.2 Neural network methods

Artificial neural networks, which have been around for 25 years or so, have become very popular over the last few years and have been applied very successfully to various machine learning tasks ranging from medical diagnosis, financial applications and pattern recognition to vehicle control and game-playing. For various visual recognition tasks, deep learning with multi-layer neural networks has outperformed all other competing methods [17], [16]. Deep learning has also been applied to the semantic segmentation of aerial images and yielded classifiers that are able to detect buildings, roads and other object in aerial images on a per-pixel basis with high confidence [25], [23], [30] and [21].

The rise of deep learning with multi-layer neural networks over the last few years was made possible by methodological advances, faster computers and the availability of large data sets to train neural networks on. What distinguishes deep learning from conventional machine learning techniques is that the division into feature extraction and the actual learning task has become largely irrelevant. For image recognition, typically the raw pixel values of an image are passed through a neural network. In the learning phase, backpropagation is used to calculate parameter weight updates, which enables the neural network to automatically learn those features from the data that are most discriminative for that task that is to be learned. Typically all neural network parameters are learned from raw image data and the corresponding labels, which overcomes the need for hand-crafted feature generation.

Convolutional Neural Networks

Convolutional neural networks (CNNs) are neural networks that have received much attention in recent years. CNNs have been applied very successfully to natural language processing [2], recommender systems [38] and image recognition [17], [16], [20] and [9]. CNNs outperform all competing methods on most visual learning tasks and have become increasingly popular also outside the scientific community.

CNNs are artificial neural networks that are inspired by the organization of the animal visual cortex. Hubel and Wiesel [12], [13] showed that the visual cortex of animals contains cell arrangements, so-called receptive fields, which are responsible for detecting light in small, overlapping sub-regions of the visual field. CNNs aim at rebuilding this architecture to some extent
in artificial neural networks. Characteristic for CNNs are neurons that are organized in alternatively stacked convolutional and pooling layers. Each neuron of a convolutional or pooling layer processes a relatively small subregions (the neuron’s receptive field), of its inputs. Often the receptive fields of neighbouring neurons overlap.

One of the earliest successful applications of CNNs was the hand-written check numbers recognition system that was proposed by LeCun et al. [17], [18] (reviving earlier work of Fukushima [6]). With the rise of GPU-accelerated computing a couple of years ago, it became possible to train larger and deeper CNNs and to apply them to more complex machine learning tasks. In 2012 for example, Krizhevsky et al. trained a large and deep CNN for the classification of 1.2 million high-resolution images into 1000 different classes in the ImageNet LSVRC-2010 contest and achieved excellent classification results [16]. Among the first to apply CNNs to the learning of semantic segmentation of aerial images were Mnih et al. [24], [25] and [23].

**Fully Convolutional Networks**

Fully convolutional networks (FCNs) are deep neural networks for semantic segmentation tasks, which emerged from CNNs. While CNNs are successful in detecting objects in images, repeated convolution and pooling typically blurs spatial information and makes it difficult to trace activations in the CNN back to the regions of the input image, where these activations eventually came from. FCNs overcome this limitation by reinterpreting fully connected CNN layers as convolutional layers and by optionally performing in-network upsampling by so-called deconvolutional layers. A detailed description of FCNs and how CNNs are transformed into FCNs is provided in chapter 3 of this work.

FCNs were first described in Long et al. [20], who achieved state-of-the-art segmentation results on the PASCAL VOC [5], NYUDv2 [33] and SIFT Flow [19] datasets. Moreover, FCNs have been applied to the semantic segmentation of aerial images by Marmanis et al. who trained FCN models on the ISPRS semantic labeling benchmark dataset for aerial images and achieved excellent segmentation results [21].

**2.3 Generating New Ground Truth Datasets**

Mnih et al. [24] suggested an approach to generate new ground truth datasets of aerial images and corresponding pixel-wise object label files in 2010. They suggested to combine aerial images, which they obtained from publicly available sources like libraries, with geographic coordinates of buildings and roads from OpenStreetMap (OSM). OSM is a collaborative project for creating free and editable maps of the world. The approach they describe is similar to some extent to the approach described in this work to generate
new ground truth data. However, there are several methodological differences. First, Mnih et al. used aerial images with ground sampling distances on the order of 1 meter, which they did not obtain from online map services. This work uses aerial images with ground sampling distances on the order of centimeters, which are publicly available and can be downloaded over the internet from online map services. Second, they manually readjusted pixel-wise object labels to get accurate ground truth data. The approach developed in this work does not require to manually readjusting pixel-wise object labels to get accurate ground truth data. Third, to determine widths of roads, Mnih et al. did not make use of the information that OSM provides in the form of highway tags. They rather treated each type of road in the same way. This work makes use of the highway tags, which OSM provides, by specifying a default road width for each highway tag.

An approach to combine VHR aerial images from online map services with geographic coordinates of road centerlines from OSM is also described in Mattyus et al. [22]. Mattyus et al. used VHR aerial images and geographic coordinates of road centerlines to learn to estimate road widths by using a Markov random field. At predicting time, given the coordinates of road centerlines, their approach yields pixel-wise road predictions for aerial images that are very good. This demonstrates that combining freely available aerial images from online map services with geographic coordinates of man-made objects from OSM is suitable to learn complex visual recognition tasks.
Chapter 3

Fully Convolutional Networks

This chapter describes fully convolutional networks (FCNs). FCNs are deep neural networks that emerged from convolutional neural networks (CNNs). Thus this chapter first starts with a description of the base architecture of CNNs. The second section describes then how CNNs are transformed into FCNs and the third section describes how FCNs are used for the semantic segmentation of images.

3.1 Architecture of Convolutional Neural Networks

3.1.1 Base Architecture

CNNs for visual recognition tasks usually contain several layers of neurons. These layers are stacked on top of each other and their neurons are interconnected. Each layer is arranged as a three-dimensional array of neurons of size $H \times W \times D$. $H$ and $W$ are spatial dimensions and $D$ is the number of feature channels (or the dimension) of the layer.

The first layer of a CNN is the image itself with $D$ being the number of color channels of the image. Characteristic for CNNs are alternatively stacked convolutional and pooling layers, which process relatively small sub-regions, so-called receptive fields, of their inputs. The outputs of convolutional and pooling layers are then tiled such that they form again three-dimensional arrays representing the input image in different feature spaces. Often the receptive fields of neighbouring neurons overlap, which means that spatial information is processed multiple times. Repeated convolution and pooling typically reduces the spatial dimension $H \times W$ of the image, when being passed through the CNN, while at the same time increasing the number of feature channels $D$.

The stack of layers that forms a CNN eventually transfers the input
image into an output through a function that is differentiable. The different type of layers that are commonly used in CNNs are discussed bellow.

3.1.2 Convolutional Layers

Convolutional layers are the main building blocks of CNNs. Their parameters are learnable filters, which are usually relatively small in size. All filters of a convolutional layer have the same size. During a forward pass, each filter of a convolutional layer gets convolved across the whole height $H$ and width $W$ of the input to the convolutional layer. Convolution is the dot product between the elements of the filter and the respective region in the input to the convolutional layer. For each filter of a convolutional layer, the convolutions calculated across the input produce a two-dimensional output. These filter-wise, two-dimensional outputs of a convolutional layer are then stacked on top of each other and form the three-dimensional output of the convolutional layer.

Each neuron of a convolutional layer is usually only connected to a small region of the previous layer (its receptive field). This causes filters to learn spatially local input patterns. Stacking multiple convolutional layers on top of each other leads to a local-to-global architecture. First, the CNN learns to recognize and represent small regions of the input image; then, when this process is repeated, the CNN generates representations over ever larger spatial contexts and assembles these to more global, semantic representations of the image. The more convolutional layers are stacked on top of each other, the larger usually the receptive fields get with respect to the input image.

Let a convolutional layer have a fixed number $D$ of convolutional filters, each of which has the same size $h \times w$. A convolutional layer takes data of size $H' \times W' \times D'$ as input and produces data of size $H \times W \times D$ as output (see Figure 3.1). In addition, each convolutional layer specifies a so-called stride parameter $s$ and padding parameters $p_h$ and $p_w$. The stride parameter $s$ specifies the interval at which the filters are applied to the input and the padding parameters $p_h$ and $p_w$ specify the number of white pixels that are added to each side of the input. The convolutional layer thus outputs data of size $\left\lfloor \frac{H + 2p_h - h + 1}{s} \right\rfloor \times \left\lfloor \frac{W + 2p_w - w + 1}{s} \right\rfloor \times D$. Where $\lfloor x \rfloor$ indicates the floor function, which maps a real number to its largest previous integer.

Let $x_{i'j'd'}$ be a neuron at spatial coordinates $(i', j')$ in the $d'$-th feature channel of the input to a convolutional layer. Let $\theta_{pqd'd'}$ be the weight value at position $(p, q)$ of the $d$-th filter of the convolutional layer with respect to the $d'$-th feature channel of the input to the convolutional layer. Let $y_{ijd}$ be a neuron at spatial coordinates $(i, j)$ in the $d$-th feature channel of the output of the convolutional layer. The output of $y_{ijd}$ is calculated by

$$y_{ijd} = \sum_{d'=0}^{D'-1} \sum_{p=0}^{h-1} \sum_{q=0}^{w-1} x_{s+i+p, s+j+q, d'} \cdot \theta_{pqd'd'} + b_d$$

18
Where \( b_d \) is the bias parameter of the \( d \)-th filter of the convolutional layer. The filter weights \( \theta_{pqdd} \) and the bias terms \( b_d \) are the learnable parameters of convolutional layers.

### 3.1.3 Activation Functions

Activation functions determine how strongly the outputs of neurons in a neural network are activated. In a CNN, activation functions usually take the outputs of convolutional layers as input. In its simplest form, an activation function is a binary function which determines whether a neuron is firing or not. A classical activation function, which is often used in neural networks, is the sigmoid activation function \( f(x) = (1 + \exp(-x))^{-1} \).

In modern CNNs however, often rectified linear units (ReLU) are used as activation functions. The effectiveness of ReLU for learning speed and for convergence was described by Nair and Hinton [27]. When a ReLU receives a negative input, it forwards zero. When a ReLU receives a positive input, it forwards this input unchanged (see Figure 3.2). ReLU activation functions don’t have learnable parameters.

Let \( y_{ijd} \) be the input and \( y_{ijd}^* \) be the output of the ReLU activation function. \( y_{ijd}^* \) can be calculated as follows.

\[
y_{ijd}^* = \max(0, y_{ijd})
\]

### 3.1.4 Pooling Layers

A pooling layer subsamples the neurons of a CNN layer and usually reduces the spatial resolution \( H \times W \) of the data. Pooling layers usually follow convolutional layers after the outputs of convolutional layers have been processed by activation functions. The subsampling applied by pooling layers
is a form of non-linear downsampling. Max pooling is most frequently used in CNNs.

A max pooling layer partitions its input data into rectangular sub-regions of size $\tilde{h} \times \tilde{w}$. For each sub-region the max pooling layer outputs the maximum value. Figure 3.3 illustrates a part of a pooling layer where pooling is applied to rectangular sub-regions of size $3 \times 3$. Like in the case of convolutional layers, a stride parameter $\tilde{s}$ can be specified, which determines the interval at which the max pooling operation is applied to the input.

Let $y_{i',j',d}^*$ be a neuron at spatial coordinates $(i', j')$ of the $d$-th feature channel of the input of a pooling layer. Let $\tilde{y}_{ijd}$ be a neuron at spatial coordinates $(i, j)$ of the $d$-th feature channel of the output of a pooling layer. $\tilde{y}_{ijd}$ is calculated as follows

$$\tilde{y}_{ijd} = \max(\{ y_{i*+p, j*+q, d}^* \mid 0 \leq p \leq \tilde{h} - 1, 0 \leq q \leq \tilde{w} - 1 \})$$
The idea behind pooling layers is that once a feature has been found, it’s enough to know the rough location of the feature relative to other features and the exact location of the feature itself is not needed any more. By applying pooling layers, the spatial resolution $H \times W$ of the data is typically reduced, which makes computation faster. There are no learnable parameters in max pooling layers.

### 3.1.5 Fully Connected Layers

Each neuron in a fully connected layer is connected to each neuron of the previous layer (see Figure 3.4). The receptive field of a neuron in a fully connected layer is thus the entire image. This means that after a fully connected layer it’s not possible any more to trace an activation in the CNN back to a particular region of the input image. After a fully connected layer all spatial context information is lost. For semantic segmentation, this poses a problem since after a fully connected layer no information about the localization of the features in an image, is available any more. FCNs, which are discussed below, are one way to keep spatial information in convolutional neural networks.

In the local-to-global architecture of CNNs, fully connected layers are usually the top layers, responsible for the "high-level reasoning", and only followed by dropout, softmax and/or loss layers. Technically, fully connected layers are convolutional layers with filters that have the same size as the data that act as input to the fully connected layers. Parameters are thus learned like in the case of convolutional layers.
3.1.6 Dropout Layers

Fully connected layers occupy many parameters in a CNN and are prone to overfitting [11]. To prevent overfitting, the dropout method was suggested by Hinton et al. [11]. Dropout is implemented in dropout layers that usually take the outputs of fully connected layers as inputs.

At each training stage, individual input neurons of a dropout layer are either dropped out of the net or kept in the net with probability $p$ or $1 - p$, respectively. Incoming and outgoing edges to dropped-out neurons are also removed from the net. By doing this, at each training stage, only a reduced network is trained on the data. Before starting the next training stage, the removed neurons and their incoming and outgoing edges are reinserted into the network.

At testing time, it would be ideal to calculate an average of all $2^n$ possible dropped-out networks, where $n$ is the number of the input neurons to the dropout layers. However, for large CNNs, this is computationally not feasible. But an approximation of this average can be calculated by using the complete network and weighting each neuron’s output by a factor of $p$. By doing this, the expected value of any neuron’s output is the same at training and at testing time.

Dropout reduces overfitting greatly in large feedforward neural networks and leads to a significant improvement of training speed [11].

3.1.7 Loss Layers

The kind of loss functions that is used for learning with a CNN depends on the kind of task that is learned. For predicting a single class out of $N$ mutually exclusive classes, a softmax loss or a multinomial logistic loss can be used. If the aim is to predict $N$ independent probability values in $[0, 1]$, a sigmoid cross-entropy loss is a suitable loss function. For regressing to real-values labels, an Euclidean loss function can be used for learning.

In this work, all neural networks are trained by using a multinomial logistic loss function applied on a per-pixel basis. This is, for each of the $n$ pixels of an input image to a CNN a multinomial logistic loss is calculated. These pixel-wise multinomial logistic losses are then summed up to one multinomial logistic loss which evaluates the “cost” associated to the entire image.

Let $N$ be the number of pixels in the image that is used as an input to the CNN and let $l_n \in \{0, \ldots, K-1\}$ be the correct object class label for each of the $n \in N$ pixels of the input image. $K$ indicates the number of different object labels. Then the multinomial logistic loss of the entire image is calculated by

$$L = \sum_{n=1}^{N} \log(\hat{p}_{n,l_n})$$
where $\hat{p}_{n,l_n}$ is the estimated probability that pixel $n$ belongs to object label $l_n$ (the correct object label).

### 3.2 Transforming CNNs into FCNs

In visual learning tasks, CNNs were originally used for assigning a single object label like "dog" or "car" to an entire image. Repeated convolution and pooling leads to a reduction of resolution and a blurring of spatial information (see illustration in the left side of Figure 3.6). Fully connected layers, finally, combine information from the entire image and throw away all spatial information. Without any modification, a typical CNN architecture is thus not suitable for learning the semantic segmentation of images.

The so-called shift-and-stitch method is one possibility to use CNNs for the semantic segmentation of images. The shift-and-stitch method uses shifted sub-regions of an image as input to the CNN. Each output of the CNN is then interpreted as a prediction for the pixels in the center of one of the shifted input images. The outputs are stitched together and, if necessary, upsampled to obtain pixel-wise predictions for the original input image. However, this procedure is computationally expensive and was found to be less efficient than using fully convolutional networks (FCNs) for obtaining pixel-wise object label predictions [20].

FCNs are first described in Long et al. [20]. Long et al. reinterpret fully connected layers as convolutional layers with filters that cover the entire inputs of the fully connected layers. Outputs of the fully connected layers, this is the fully connected neurons, are then arranged as three-dimensional arrays with two spatial dimensions and one feature dimension. By doing so, spatial information is reintroduced into the neural network and fully connected layers can output classification maps. By reinterpreting fully connected layers as convolutional layers, CNNs are transformed into fully convolutional networks. FCNs are built of convolutional layers, pooling layers and activation layers as basic components [20].

When transforming a CNN into an FCN, however, it is not necessary to interpret all fully connected layers as convolutional layers with filters that cover the entire region of the inputs to the fully connected layers. It is common to transform fully connected layers into convolutional layers that have smaller filter sizes. Some fully connected layers are even transformed into convolutional layers that have filters of size $1 \times 1$ [20].
3.3 FCNs for the Semantic Segmentation of Images

Deconvolutional Layers

An FCN is able to output a classification map. However, these classification maps usually have a rather coarse spatial resolution. This is because convolutional and pooling layers in the lower parts of the FCN downsample the data with regard to the spatial resolution of the original image (see left side of Figure 3.6).

One way to upsample the data back to a higher resolution is by backward convolution, sometimes also called deconvolution [20] and implemented in so-called deconvolutional layers. This in-network upsampling is technically not the inverse or a generalized inverse of the function defined by a convolutional layer. It is rather backwards strided convolution in the sense that a deconvolutional layer multiplies inputs to the deconvolutional layer by a filter element-wise and sums over the resulting output windows to produce an upsampled output (see Figure 3.5). This operation is simple to implement because it just reverses the forward and the backward passes of a convolutional layer. This makes the optimization of the learnable parameters of a deconvolutional layer straightforward. Deconvolutional layers are usually inserted into the upper parts of FCNs.

Deconvolutional layers specify the same set of learnable parameters as convolutional layers. These are in particular, the filter size $h \times w$ and the striding parameter $s$. It’s the combination of the filter size $h \times w$ and the striding parameter $s$ that determine the upsampling factor. Let $y_{ijd}$ be the neuron at spatial coordinates $(i, j)$ in the $d$-th feature channel of the output of a deconvolutional layer. Let $x_{i'}$ be any neuron in the input to the deconvolutional layer that is contained in the receptive field of $y_{ijd}$. Let $\theta_{i',d'}$ be a filter weight that is multiplied with $x_{i'}$ for calculating the output of $y_{ijd}$. Where $D'$ is the number of feature channels of the input to the deconvolutional layer and $d' \in D'$. Let further $I'$ be the set that contains the indices of those $x_{i'}$ that are in the receptive field of $y_{ijd}$. The output of $y_{ijd}$ is then calculated as follows

$$y_{ijd} = \sum_{i' \in I'} \sum_{d'=0}^{D'-1} x_{i'} \theta_{i',d'}$$

Combining Semantic Information with Spatial Information

Deconvolutional layers enable in-network upsampling of the data back to the resolution of the original image. However, the deconvolutional layers have to use data for upsampling that typically have a rather coarse resolution. To recover finer details, Long et al. proposed a neural network model, which
Figure 3.5: Part of a deconvolutional layer. $f_{d,d'}$ depicts the $d'$-th slice of the $d$-th filter of the deconvolutional layer, which uses data from the $d'$-th feature channel of the input to the deconvolutional layer to produce the $d$-th feature channel of the output of the deconvolutional layer. $f_{d,(d'+1)}$ depicts the $(d'+1)$-th slice of the $d$-th filter of the deconvolutional layer, which uses data from the $(d'+1)$-th feature channel of the input to the deconvolutional layer to produce the $d$-th feature channel of the output of the deconvolutional layer. (a) and (b) show the same input feature channels ($d'$ and $(d'+1)$) and the same output feature channels ($d$) but with the filters $f_{d,d'}$ and $f_{d,(d'+1)}$ applied at different spatial locations of the input data.
they called skip-architecture [20]. The skip-architecture combines semantic information from deep FCN layers with spatial information from shallow FCN layers. These additional network connections bypass the deep neural network layers, in which spatial information is blurred, and add localization information to the final prediction (See Figure 3.6).

Transforming a CNN into an FCN, which additionally performs in-network upsampling by deconvolutional layers and which fuses information from different neural network layers, yields a differentiable function, which transforms the input image into pixel-wise object class scores. Thus FCNs are multi-layer neural networks that can be trained end-to-end, pixels-to-pixels for semantic segmentation tasks.

Figure 3.6: Conceptional illustration of the data flow through a fully convolutional network (FCN), which is used for the semantic segmentation of aerial images. Three "skips" are highlighted by pale red, pale green and pale blue, respectively. Figure is based on Figure 1 in Long et al. [20].
Chapter 4

Generating New Ground Truth Data

This chapter describes a semi-automatic approach to generate new ground truth datasets of very-high resolution (VHR) aerial images in RGB format and corresponding pixel-wise object label files for the object label classes building, road and background. Aerial images are downloaded from Google Maps and geographic coordinates of buildings and roads are downloaded from OpenStreetMap (OSM). In this work, three large, new ground truth datasets for the cities of Chicago, Paris and Zurich, respectively, were generated. This chapter describes in the following sections the semi-automatic approach to generate new ground truth datasets and how the three new ground truth datasets for Chicago, Paris and Zurich were generated.

4.1 Aerial Images from Google Maps

To generate new ground truth datasets, aerial images were downloaded from Google Maps via the Google Static Maps API\(^1\).

Zoom level

The Google Static Maps API requires the user to specify a zoom level, a natural number between 0 and 22, before downloading aerial images in RGB format. The zoom level specifies the resolution of the aerial image that is returned to the user. For each zoom level, Google Maps divides the world’s surface into a set of square-shaped, so-called tiles that consist of \(256 \times 256\) pixels. At zoom level 0, the whole world is contained in one single tile. Increasing the zoom level by 1 leads to a magnification by a factor of 2. This is, if \(z\) is the zoom level, the number of tiles into which Google Maps divides the world’s surface is \(2^z \times 2^z\). The maximal zoom level, which is available for

---

\(^1\)https://developers.google.com/maps/documentation/static-maps/
downloading aerial images, depends on the geographic location. For large parts of North America and Europe, zoom level 20 is the maximal zoom level available. A tile of Chicago at zoom level 20 depicts a square with side length 28.416 meter, contains $256 \times 256 = 65,536$ pixels and has a ground sampling distance of $28.416/256 = 0.111$ meter per pixel.

**Downloading aerial images**

In this work, aerial images of the cities of Chicago, Paris and Zurich were downloaded from Google Maps at zoom level 20. For each city, a rectangular geographic area covering the city’s center is specified. These cover about 46.1 km$^2$, 50.9 km$^2$ and 31.4 km$^2$ of Chicago, Paris and Zurich, respectively. For each city, this large rectangular area is divided into smaller rectangles, which cover roughly 0.08 km$^2$ each. For each of those rectangles, all Google Maps aerial image tiles, which cover the rectangle, are downloaded from Google Maps and assembled to one large aerial image in RGB format.

Those large aerial images obtained by the assembly of Google Maps aerial images tiles cover, like the rectangles from above, about 0.08 km$^2$ each. However, the large aerial images are usually somewhat larger than their corresponding rectangles because the rectangles are completely contained in the large aerial images. Moreover, the large aerial images usually overlap to some extent. This is, because the Google Maps aerial image tiles, which cover the border between two adjacent rectangles, are usually included in both large aerial images that correspond to the two adjacent rectangles, respectively. The maximal overlap between two large aerial images is roughly 10%. However, the large aerial images will be cropped into images patches later, which are then used as FCN inputs (explained in section 5.2). This cropping will eliminate most of the redundant parts among the large aerial images.

### 4.2 Building and Road Coordinates from OpenStreetMap

Geographic coordinates of buildings and roads were downloaded from OpenStreetMap (OSM). OSM is a collaborative project for creating free and editable maps of the world. In contrast to other online map services, however, the geographic coordinates of all objects in the OSM databases, such as e.g. building corners, can be accessed freely and retrieved via the OpenStreetMap API$^2$. In addition, OSM provides so-called highway tags, which are assigned to each road in the OSM database. These highway tags can be used to specify widths for each road. This is useful because most online map services encode roads as lines that don’t have a constant, pre-defined

---

$^2$http://wiki.openstreetmap.org/wiki/API_v0.6
width. For buildings, OSM provides the geographic coordinates of the buildings’ corners. For roads, OSM provided the geographic coordinates of roads’ centerlines.

Geographic coordinates of buildings and roads were downloaded for each large aerial image of Chicago, Paris and Zurich from OSM.

### 4.3 Generating Pixel-wise Object Label Files

To generate pixel-wise object label files, the geographic coordinates of building corners and road centerlines, which were downloaded from OSM for each large aerial image of Chicago, Paris and Zurich, were transformed into pixel coordinates with respect to that large aerial image in which they are included. Next, for each large aerial image an empty image of the same size was used as a framework for generating a pixel-wise object label file. For each building in the large aerial image, a polygon was plotted into the empty image with the polygon corners corresponding to the building corners.

In the case of the roads, the situation was more complicated. This is because OSM only provides the geographic coordinates of road centerlines and no detailed information about road widths. However, OSM provides a highway tag for most roads that are included in the OSM database. This is, for each point, which is used to encode a road centre line, the type of the road to which the point belongs is specified. In this work, for each highway tag, a default road width was specified a-priory. This is, before aerial images were downloaded from Google Maps. These default road widths were then used to calculate road edges from road centerlines. Road edges were plotted as polygons into the framework images like in the case of the building corners. If a pixel in the object label file was labeled as both, building and road, the pixel was always classified as building. This happened occasionally at borders between buildings and roads but was rare in general. If a pixel was labeled neither as building nor as road, it was classified as background.

The method described here yielded pixel-wise object label files for the VHR aerial images obtained from Google Maps. The a-priory specification of the default road widths is the only aspect of this method to generate new ground truth data which has to be done manually.

### 4.4 New Ground Truth Datasets

In this work, three new ground truth datasets of aerial images and corresponding pixel-wise object labels files were generated for the cities of Chicago, Paris and Zurich, respectively. Coverage, number of pixels and ground sampling distance of each of the three datasets is depicted in Table 4.1. Google Maps uses the web Mercator projection to project the world’s surface to maps, which distorts the world’s surface as the latitude increases.
Table 4.1: Comparison of newly generated ground truth datasets for Chicago, Paris and Zurich and the ISPRS benchmark datasets of Potsdam and Vaihingen.

<table>
<thead>
<tr>
<th>City</th>
<th>Coverage</th>
<th>No. of pixels</th>
<th>Ground sampling distance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chicago</td>
<td>46.1 km$^2$</td>
<td>$3.7 \times 10^9$</td>
<td>11.1 cm</td>
</tr>
<tr>
<td>Paris</td>
<td>50.9 km$^2$</td>
<td>$5.3 \times 10^9$</td>
<td>9.8 cm</td>
</tr>
<tr>
<td>Zurich</td>
<td>31.4 km$^2$</td>
<td>$3.1 \times 10^9$</td>
<td>10.1 cm</td>
</tr>
<tr>
<td>ISPRS-Potsdam</td>
<td>3.4 km$^2$</td>
<td>$1.4 \times 10^9$</td>
<td>5 cm</td>
</tr>
<tr>
<td>ISPRS-Vaihingen</td>
<td>1.4 km$^2$</td>
<td>$1.7 \times 10^8$</td>
<td>9 cm</td>
</tr>
</tbody>
</table>

from the Equator to the poles. This causes slight differences in ground truth sampling distances among the three cities. Examples of aerial images and overlay pixel-wise object labels are shown in Figure 4.1, Figure 4.2 and Figure 4.3 for Chicago, Paris and Zurich, respectively.

For comparison, information about the ISPRS benchmark datasets for semantic segmentation [28] is shown as well in Table 4.1. The ISPRS semantic segmentation benchmark contains two image datasets for the two towns of Vaihingen and Potsdam in Germany. Each dataset contains VHR aerial images and corresponding pixel-wise object label files.

For a discussion about quality, applications and limitations of the newly generated ground truth datasets for Chicago, Paris and Zurich, refer to section 7.1.
Figure 4.1: Representative example of new ground truth data of Chicago. Left side shows an aerial image from Google Maps with one sub-region of the aerial image magnified. Right side shows the same aerial images overlay with the pixel-wise building and road labels obtained from OSM. Pixels that are labeled as building or road are overlay with red or blue color, respectively. Pixels that are labeled as background are not overlay with any color.
Figure 4.2: Representative example of new ground truth data of Paris. Left side shows an aerial image from Google Maps with one sub-region of the aerial image magnified. Right side shows the same aerial images overlay with the pixel-wise building and road labels obtained from OSM. Pixels that are labeled as building or road are overlay with red or blue color, respectively. Pixels that are labeled as background are not overlay with any color.
Figure 4.3: Representative example of new ground truth data of Zurich. Left side shows an aerial image from Google Maps with one sub-region of the aerial image magnified. Right side shows the same aerial images overlay with the pixel-wise building and road labels obtained from OSM. Pixels that are labeled as building or road are overlay with red or blue color, respectively. Pixels that are labeled as background are not overlay with any color.
Chapter 5
Deep Learning Framework

This chapter describes the deep learning framework which was used to learn the semantic segmentation of aerial images from the new ground truth datasets, which were generated in this work. This chapter is organized as follows. The first section describes the neural network architectures which are relevant for this work and which were used to learn the semantic segmentation of aerial images. The second section explains how the newly generated ground truth data was pre-processed before it was used for learning with fully convolutional networks (FCNs). The third section describes how the FCNs used in this work learned parameters. The fourth section describes the hyperparameters that were used for learning. The fifth section summarizes the metrics which were used to evaluate FCN performance and the sixth section, finally, describes how the deep learning framework was implemented.

5.1 Neural Network Architectures

Three different FCN architectures were used for the learning of the semantic segmentation of aerial images in this work. These are named FCN-8s, FCN-4s-1 and FCN-4s-2. FCN-8s is a neural network architecture which was proposed in Long et al. [20] in 2015. FCN-4s-1 and FCN-4s-2 are neural network architectures that expand the architecture of FCN-8s by adding new connections and new neural network layers to the base architecture of FCN-8s. FCN-4s-1 and FCN-4s-2 are introduced in this work.

All FCN models, which are used in this work, are eventually based on the famous VGG-16 network [34]. VGG-16 is a CNN architecture, which was proposed by Simonyan et al. [34] in 2014 for object recognition tasks. Characteristic for VGG-16 is an architecture that uses relatively small convolutional filters, mainly of size 3×3. These small filters lead to convolutional layers with relatively few learnable parameters. This allows to stack convolutional layers on top of each other while still yielding neural network
models that can be trained in reasonable time. VGG-16 is build of 16 layers and achieved state-of-the art classification results in the ImageNet Challenge 2014 [29].

**FCN-8s**

Long et al. [20] transformed VGG-16 into FCN-8s, a fully convolutional network. They did so by transforming VGG-16’s fully connected layers into convolutional layers and by adding deconvolutional layers, which upsample the data back to the resolution of the original input image. In addition, they added three so-called ”skips” to the FCN architecture, which fuse spatial information from shallow FCN layers with semantic information from deep FCN layers to obtain better semantic segmentations of images. Figure 5.1 illustrates the architecture of FCN-8s. The name of FCN-8s refers to the neural network layer Pool_3, which has the highest spatial resolution among those three neural network layers of FCN-8s which are used in a ”skip”. Pool_3 has a spatial resolution that is roughly 8 times lower than the spatial resolution of the original input image.

When evaluated on a subset of the PASCAL VOC 2011 semantic segmentation dataset [4], FCN-8s achieved state-of-the-art segmentation results [20]. The number of feature channels, which are used in the upper parts of FCN-8s, correspond to the number of different object labels, which are learned. In Long et al. [20], the number of feature channels is either 21, 33, 40 or 60 depending on the dataset which was used for learning. In this work, the number of different object labels is always 3 (building, road and background). For more information about the number of learnable parameters of FCN-8s, refer to Table C.1.

**FCN-4s-1**

FCN-4s-1 expands the FCN-8s architecture by adding one more ”skip” to fuse spatial information from the shallow neural network layer Pool_2. Figure 5.2 illustrates the architecture of FCN-4s-1. Pool_2 has a spatial resolution that is roughly 4 times lower than the spatial resolution of the original image that is used as FCN input. FCN-4s-1 is introduced in this work. For more information about the number of learnable parameters of FCN-4s-1, refer to Table C.2.

**FCN-4s-2**

FCN-4s-2 has an architecture, which is similar to the architecture of FCN-4s-1. But in contrast to FCN-4s-1, FCN-4s-2 does not convolve data from layers Pool_2, Pool_3, Pool_4 and Dropout_2 to outputs with $D = 3$ feature channels. FCN-4s-2 rather keeps a high number of feature channels in the process of in-network upsampling. It’s only the last deconvolutional layer
Figure 5.1: FCN-8s architecture. The neurons of each layer are arranged as a three-dimensional array. The size of each layer is indicated by the numbers in brackets $(D, H, W)$. $D$ refers to the number of feature channels and $H$ and $W$ refer to the spatial dimension of the layer. FCN-8s contains 134,277,737 learnable parameters.
Figure 5.2: FCN-4s-1 architecture. The neurons of each layer are arranged as a three-dimensional array. The size of each layer is indicated by the numbers in brackets \((D, H, W)\). \(D\) refers to the number of feature channels and \(H\) and \(W\) refer to the spatial dimension of the layer. FCN-4s-1 contains 134,276,540 learnable parameters.
Figure 5.3: FCN-4s-2 architecture. The neurons of each layer are arranged as a three-dimensional array. The size of each layer is indicated by the numbers in brackets \((D, H, W)\). \(D\) refers to the number of feature channels and \(H\) and \(W\) refer to the spatial dimension of the layer. FCN-4s-2 contains 290,867,008 learnable parameters.

(Deconv4) which eventually downsamples the number of feature channels from 1920 to 3. Figure 5.3 illustrates the architecture of FCN-4s-2. FCN-4s-2 is introduced in this work like FCN-4s-1. For more information about the number of learnable parameters of FCN-4s-2, refer to Table C.3.

5.2 Pre-processing of Newly Generated Ground Truth Data

All FCN models were trained with new ground truth data, which was generated in this work. As an input to all FCNs, RGB image patches of size 500 \times 500 pixels were used, which were cropped out of the large aerial images described in section 4.1. Cropping was done by laying a grid with cell size 500 \times 500 pixels over the large aerial images. Cropping always started from the pixel in the top left corner and ended in the bottom right region of a large aerial image. Only image patches of size 500 \times 500 were cropped out of large aerial images. This means that margins at the right or bottom edge of a large aerial image were not included in the set of image patches, which were cropped out of a large aerial image. For each image patch, a pixel-wise
Table 5.1: Number of image patches and corresponding pixel-wise object label files used for training, validation and testing.

<table>
<thead>
<tr>
<th>City</th>
<th>Training</th>
<th>Validation and Testing</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chicago</td>
<td>12,300</td>
<td>1410</td>
<td>13,710</td>
</tr>
<tr>
<td>Paris</td>
<td>21,420</td>
<td>1080</td>
<td>22,500</td>
</tr>
<tr>
<td>Zurich</td>
<td>11,422</td>
<td>602</td>
<td>12,024</td>
</tr>
</tbody>
</table>

An object label file was generated accordingly. Before an image patch was used as an input to an FCN, its pixel values were normalized by subtracting the per-channel RGB means (calculated for each city separately).

A total of 13,710, 22,500, and 12,024 image patches and corresponding pixel-wise object label files were generated for Chicago, Paris, and Zurich, respectively. For each city, the image patches and the corresponding object label files were divided into two sets. This is, (1) a set, which was used for FCN training, and (2) a set, which was used for validating and testing (see Table 5.1). Image patches and corresponding pixel-wise object label files in the validation/test sets were not further divided.

5.3 Learning Parameters by Stochastic Gradient Descent

FCNs are feedforward neural networks that are trained by backpropagation. In this work, stochastic gradient descent was used for learning parameters. The learnable parameters $\theta$ of an FCN at training iteration $t + 1$ were updated by a linear combination of the negative gradient $\frac{\partial L(\theta_{t+1})}{\partial \theta}$ and the previous parameter update $\Delta \theta_t$, where $L(\theta)$ indicates the loss as a function of $\theta$. The weight update for the learnable parameters $\theta$ at iteration $t + 1$ is thus calculated by

$$\Delta \theta_{t+1} = \mu \Delta \theta_t - \alpha \frac{\partial L(\theta_{t+1})}{\partial \theta}$$

where $\alpha$ and $\mu$ are hyperparameters for training the FCN model. $\alpha$ is the learning rate, which specifies the weight of the negative gradient, and $\mu$ is the momentum, which is the weight of the previous parameter update.

5.4 Hyperparameters Used for Learning

All model parameters were learned in this work by using a multinomial logistic loss, which was calculated on a per-pixel basis and summed up to a loss for the entire image patch of size 500 $\times$ 500, which acted as input to the FCN. All models were trained using stochastic gradient descent with a momentum of 0.9. A minibatch size of 1 image was used. Learning rates
always started from $5 \times 10^{-9}$ and were divided by a factor of 10 two times when loss and $F_1$ average scores reached a plateau. The learning rates for biases of convolutional layers were doubled with respect to learning rates of the filter weights. A weight decay of $5 \times 10^{-4}$ was used. Between 45,000 and 140,000 iterations were used to train models, which corresponds to training between 3.4 and 6.5 epochs. All pooling layers performed max pooling and only rectified linear units were used as activation functions. The probability that a certain neuron was dropped in a dropout layer was always 0.5.

Models were either learned from scratch or pre-trained weights were used to initialize parameters of FCN layers. When models were learned from scratch, weights were initialized as in Glorot et al. [7]. When models were initialized with pre-trained weights, only pre-trained weights were used that have been obtained in this work by training on newly generated ground truth data.

5.5 Evaluation Metrics

To evaluate FCN performance at training and testing time, for each of the three object labels building, road and background and for an average across them, scores for precision, recall and $F_1$ were calculated on a per-pixel basis. Precision and recall are common metrics for evaluating the performance of neural network models that predict pixel-wise object labels in aerial images [30], [23] and [25]. For each object label, precision is defined as the pixel ratio of true positives to true positive plus false positives. Recall is defined as the pixel ratio of true positives to true positives plus false negatives. The $F_1$ score is a weighted average of precision and recall. For each object label, the $F_1$ score is calculated by

$$F_1 = \frac{2 \cdot \text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$$

To evaluate FCN performance during training, precision, recall and $F_1$ scores were calculated every 500 iterations. For each validation cycle, 100 image patches were used, which amounts to $2.5 \times 10^7$ pixels. To evaluate FCN performance after training has finished, this is at testing time, precision, recall and $F_1$ scores were calculated using all image patches in the corresponding validation/test sets. However, a slightly modified approach was used to calculate precision, recall and $F_1$ scores at testing time compared to when these measures were calculated at training time. Why and how this was done is explained in the next paragraph.

When predicting pixel-wise object labels in an aerial image patch of size $500 \times 500$, usually the predictive performance gets worse at the image boundaries. This is because at image boundaries less spatial context is available for the FCN to estimate the correct pixel-wise object labels. However, when
aiming at the semantic segmentation of an aerial image, which is bigger than 500 × 500 pixels, image patch boundaries inside the aerial images can be avoided. This is possible by cropping the aerial image such that the image patches overlap with each other. The image patches are then used as an input to an FCN and the pixel-wise class scores obtained from the FCN for the different image patches are combined to pixel-wise class scores for the aerial image. This approach allows to generate better semantic segmentations of aerial images. At testing time, scores for precision, recall and $F_1$ were calculated based on weighted averages of pixel-wise class scores.

In this work, each large aerial image used at testing time was cropped into image patches four times using four different grids. The grids overlapped with each other such that the edge of an image patch in one grid was at the center of an image patch in another grid. Figure 5.4 illustrates the four different grids that were used to crop large aerial images into image patches. To predict pixel-wise object labels of a large aerial image, then, a weighted average of the four scores was calculated, which were obtained from the four different grids. A Gauss function was used to calculate the weights of the scores. The closer a pixel was to the center of an image patch, the higher was its weight.

Figure 5.4: Example of the four different grids (indicated by black lines) which were used to crop large aerial images into image patches of size 500 × 500 pixels.

5.6 Implementation

All models used in this work were trained and tested using single NVIDIA GeForce GTX TITAN X GPUs and the deep learning framework Caffe [14].
Chapter 6

Experiments

This chapter is divided into three sections. The first section describes how FCN models were trained for the semantic segmentation of aerial images using newly generated ground truth data of individual cities (Chicago, Paris and Zurich) and what predictive performance these FCN models achieved. The second section describes the predictive performance of the three different FCN architectures, which were used to learn the semantic segmentation of aerial images in this work. The third section describes how an FCN model was trained simultaneously on combined data of Chicago, Paris and Zurich and what predictive performance this FCN model achieved. The specific learning rates, which were used to train all FCN models in this work, can be found in Table C.5.

6.1 Learning from Single Cities

Neural networks with FCN-8s, FCN-4s-1 and FCN-4s-2 architecture were trained on the newly generated ground truth datasets of Chicago, Paris and Zurich, respectively. For each city, an FCN-8s, FCN-4s-1 and FCN-4s-2 model was trained separately, yielding a total of nine different, fully trained FCN models. All neural networks with FCN-8s architecture were trained from scratch. FCN-4s-1 shares 18 of its 19 convolutional layers and 2 of its 4 deconvolutional layers with FCN-8s. Before training an FCN-4s-1 model on data of a single city, the parameter weights of those neural network layers of FCN-4s-1 that are shared with FCN-8s were initialized with the weights of that FCN-8s model, which previously had been fully trained on data of the same city. FCN-4s-2 shares all 15 of its 15 convolutional layers and none of its 4 deconvolutional layers with FCN-8s. Before training an FCN-4s-2 model on data of a single city, the parameter weights of those layers of FCN-4s-2 that are shared with FCN-8s were initialized like in the case of FCN-4s-1.

Each of the nine fully trained FCN models was used to predict the whole
test/validation set of the city it was trained on and $F_1$ scores were calculated for each of the object labels building, road and background and for an average over these three object labels. For precision and recall scores, which were calculated along with $F_1$ scores, refer to Table C.6, Table C.7 and Table C.8, for Chicago, Paris and Zurich, respectively. $F_1$ scores are shown in Table 6.1 for Chicago, in Table 6.2 for Paris and in Table 6.3 for Zurich. Figure 6.1 shows the semantic segmentations of an aerial image of Chicago predicted by those FCN models, which were trained on Chicago data. Subfigures (c), (d) and (e) in Figure 6.1 show the semantic segmentations obtained from FCN-8s, FCN-4s-1 and FCN-4s-2 models, respectively. Figure 6.2 and Figure 6.3 show analogous figures for Paris and Zurich. Comparisons of predictive performances of fully trained FCN models hereafter are based on $F_1$ average scores unless stated otherwise.

**Chicago, Paris and Zurich**

Each fully trained FCN (FCN-8s, FCN-4s-1 and FCN-4s-2) achieved the best predictions for Chicago, followed by Zurich and Paris (see Table 6.1, Table 6.2, Table 6.3). The predictive performances for Chicago were around 5% better than for Zurich and the predictive performances for Zurich were around 4% better than for Paris (for each of FCN-8s, FCN-4s-1 and FCN-4s-2). This is also apparent from the semantic segmentations shown in Figure 6.1, Figure 6.2 and Figure 6.3. Semantic segmentations for Chicago were more accurate than for Paris and for Zurich. In the case of Paris, roads were often predicted to be more narrow than they actually are in the aerial image (see Figure 6.2). However, the widths of the predicted roads were similar to the widths of the roads in the ground truth data, which were also too narrow.

**Buildings and Roads**

In the case of Chicago and Zurich, each model (FCN-8s, FCN-4s-1 and FCN-4s-2) predicts buildings better than roads. Building predictions are between 4.0% and 8.9% better than road predictions. In the case of Paris however, it’s the other way round and predictions for roads are between 4.9 and 5.8% better than predictions for buildings (see Table 6.1, Table 6.2, Table 6.3).

**Cross-Predictions**

To investigate how well fully trained FCN models predict aerial images of different geographic regions, "cross-predictions" were made. This is, FCN models trained on ground truth data of one city were used to predict aerial images of the other two cities. Results are similar for all three FCN architectures FCN-8s, FCN-4s-1 and FCN-4s-2. For simplicity, just the results
obtained from the three FCN-4s-1 models are shown, which were trained on the ground truth datasets of Chicago, Paris and Zurich, respectively.

For each image, which was "cross-predicted", those RGB channel means were subtracted which were used to train the FCN model used for the prediction. Subtracting the RGB channels means of the city, to which the predicted aerial image belongs, produced predictions which were similar or worse in quality.

The semantic segmentations obtained from "cross-predictions" of FCN-4s-1 models are shown in Figure 6.4. Most building predictions in "cross-predictions" were fragmentary and patchy. Road predictions were mostly missing. The city structure of Chicago appears to be especially difficult to predict for the FCN-4s-1 models trained on data of Paris and Zurich, respectively.

6.2 Three Different FCN Architectures

All FCN-4s-1 models achieved better predictions than the corresponding FCN-8s models. FCN-4s-1 was 0.9%, 0.3% and 1.4% better than FCN-8s in the case of Chicago, Paris and Zurich, respectively (see Table 6.1, Table 6.2, Table 6.3). The better semantic segmentations of FCN-4s-1 compared to FCN-8s are also apparent from Figure 6.1 for Chicago and in Figure 6.3 for Zurich. It can also be seen that FCN-4s-1 was able to connect road segments better than FCN-8s for Chicago and for Zurich. For Paris the performance improvement of FCN-4s-1 over FCN-8s is more difficult to see and with 0.3% relatively small.

FCN-4s-2 achieved better predictions than FCN-8s in the case of Chicago (1.3%) and in the case of Zurich (1.7%). For Paris, however, FCN-4s-2 and FCN-8s showed the same predictive performance. FCN-4s-2 yielded better predictions than FCN-4s-1 in the case of Chicago (0.2%) and Zurich (0.3%). For Paris, however FCN-4s-2 was 0.3% worse than FCN-4s-1. From the semantic segmentations shown in Figure 6.1, Figure 6.2, Figure 6.3, FCN-4s-2 seems to be able to connect road fragments even better than FCN-4s-1 in the case of Chicago and Zurich. For Paris, the semantic segmentations obtained from FCN-4s-2, FCN-4s-1 and FCN-8s appear to be very similar to each other.
Table 6.1: $F_1$ scores for Chicago. FCN-8s, FCN-4s-1 and FCN-4s-2 were trained and evaluated on data of Chicago. FCN-4s-1-c was simultaneously trained on combined data of Chicago, Paris and Zurich and evaluated on data of Chicago. FCN-residual* is a neural network that aims at residual learning as described in He et al. [9]. FCN-residual yielded predictions that were slightly worse than the predictions of the other FCN-models. FCN-residual is described and discussed in Appendix A.

<table>
<thead>
<tr>
<th></th>
<th>FCN-8s</th>
<th>FCN-4s-1</th>
<th>FCN-4s-2</th>
<th>FCN-4s-1-c</th>
<th>FCN-residual*</th>
</tr>
</thead>
<tbody>
<tr>
<td>$F_1$ average</td>
<td>0.816</td>
<td>0.827</td>
<td><strong>0.829</strong></td>
<td>0.825</td>
<td>0.813</td>
</tr>
<tr>
<td>$F_1$ building</td>
<td>0.842</td>
<td>0.849</td>
<td><strong>0.850</strong></td>
<td>0.848</td>
<td>0.842</td>
</tr>
<tr>
<td>$F_1$ road</td>
<td>0.801</td>
<td><strong>0.809</strong></td>
<td><strong>0.809</strong></td>
<td>0.805</td>
<td>0.792</td>
</tr>
<tr>
<td>$F_1$ background</td>
<td>0.805</td>
<td>0.823</td>
<td><strong>0.828</strong></td>
<td>0.821</td>
<td>0.805</td>
</tr>
</tbody>
</table>

Table 6.2: $F_1$ scores for Paris. FCN-8s, FCN-4s-1 and FCN-4s-2 were trained and evaluated on data of Paris. FCN-4s-1-c was simultaneously trained on combined data of Chicago, Paris and Zurich and evaluated on data of Paris.

<table>
<thead>
<tr>
<th></th>
<th>FCN-8s</th>
<th>FCN-4s-1</th>
<th>FCN-4s-2</th>
<th>FCN-4s-1-c</th>
</tr>
</thead>
<tbody>
<tr>
<td>$F_1$ average</td>
<td>0.764</td>
<td><strong>0.767</strong></td>
<td>0.764</td>
<td>0.754</td>
</tr>
<tr>
<td>$F_1$ building</td>
<td>0.751</td>
<td>0.755</td>
<td><strong>0.759</strong></td>
<td>0.742</td>
</tr>
<tr>
<td>$F_1$ road</td>
<td>0.809</td>
<td><strong>0.810</strong></td>
<td>0.808</td>
<td>0.801</td>
</tr>
<tr>
<td>$F_1$ background</td>
<td>0.732</td>
<td><strong>0.737</strong></td>
<td>0.725</td>
<td>0.718</td>
</tr>
</tbody>
</table>

Table 6.3: $F_1$ scores for Zurich. FCN-8s, FCN-4s-1 and FCN-4s-2 were trained and evaluated on data of Zurich. FCN-4s-1-c was simultaneously trained on combined data of Chicago, Paris and Zurich and evaluated on data of Zurich.

<table>
<thead>
<tr>
<th></th>
<th>FCN-8s</th>
<th>FCN-4s-1</th>
<th>FCN-4s-2</th>
<th>FCN-4s-1-c</th>
</tr>
</thead>
<tbody>
<tr>
<td>$F_1$ average</td>
<td>0.796</td>
<td>0.810</td>
<td><strong>0.813</strong></td>
<td>0.805</td>
</tr>
<tr>
<td>$F_1$ building</td>
<td>0.892</td>
<td><strong>0.904</strong></td>
<td><strong>0.904</strong></td>
<td>0.902</td>
</tr>
<tr>
<td>$F_1$ road</td>
<td>0.823</td>
<td>0.815</td>
<td><strong>0.826</strong></td>
<td>0.820</td>
</tr>
<tr>
<td>$F_1$ background</td>
<td>0.672</td>
<td><strong>0.711</strong></td>
<td>0.708</td>
<td>0.694</td>
</tr>
</tbody>
</table>

46
Figure 6.1: Chicago. Aerial image, ground truth labels and out-sample predictions for a representative area of Chicago. One sub-region of the aerial image is magnified. FCN-8s, FCN-4s-1 and FCN-4s-2 were trained on Chicago data only. FCN-4s-1-c was simultaneously trained on combined data of Chicago, Paris and Zurich. Pixels that are labeled as building or road are overlay with red or blue color, respectively. Pixels that are labeled as background are not overlay with any color.
Figure 6.2: Paris. Aerial image, ground truth labels and out-sample predictions for a representative area of Paris. One sub-region of the aerial image is magnified. FCN-8s, FCN-4s-1 and FCN-4s-2 were trained on Paris data only. FCN-4s-1-c was simultaneously trained on combined data of Chicago, Paris and Zurich. Pixels that are labeled as building or road are overlay with red or blue color, respectively. Pixels that are labeled as background are not overlay with any color.
Figure 6.3: Zurich. Aerial image, ground truth labels and out-sample predictions for a representative area of Zurich. One sub-region of the aerial image is magnified. FCN-8s, FCN-4s-1 and FCN-4s-2 were trained on Zurich data only. FCN-4s-1-c was simultaneously trained on combined data of Chicago, Paris and Zurich. Pixels that are labeled as building or road are overlay with red or blue color, respectively. Pixels that are labeled as background are not overlay with any color.
Figure 6.4: Cross-predictions of FCN-4s-1 models trained on data of Chicago, Paris and Zurich separately. FCN-4s-1 models trained on data of one city were used for the semantic segmentation of aerial images of the other two cities. (b) and (c) are predictions of FCN-4s-1 trained on Chicago data, (a) and (f) are predictions of FCN-4s-1 trained on Paris data, (d) and (e) are predictions of FCN-4s-1 trained on Zurich data. Pixels that are labeled as building or road are overlay with red or blue color, respectively. Pixels that are labeled as background are not overlay with any color.

6.3 Learning Simultaneously from Multiple Cities

A neural network model with FCN-4s-1 architecture (named FCN-4s-1-c) was simultaneously trained on combined data of Chicago, Paris and Zurich. The FCN-4s-1 model architecture was chosen because FCN-4s-1 appears to be the most suitable model architecture to learn the semantic segmentation of aerial images (regarding predictive performance and time needed for training).

Before training, the parameter weights of FCN-4s-1-c were initialized with the parameter weights of the FCN-4s-1 model which had been fully trained on Chicago data previously. The fully trained FCN-4s-1-c model was used to predict the whole test/validation set of Chicago, Paris and Zurich, respectively, and $F_1$ scores were calculated for each of the object labels building, road and background and for an average over these three object labels. $F_1$ scores are shown in Table 6.1 for Chicago, in Table 6.2 for
Paris and in Table 6.3 for Zurich. Subfigures (f) in Figure 6.1, Figure 6.2 and Figure 6.3 show the semantic segmentations obtained from FCN-4s-1-c for a representative area of Chicago, Paris and Zurich, respectively. Comparisons of the predictive performances of fully trained FCN models are based on $F_1$ average scores unless stated otherwise.

**Predicting Chicago, Paris and Zurich**

The fully trained FCN-4s-1-c model yielded predictions for all three cities that are comparable to the predictions obtained from the three FCN-4s-1 models which were only trained on single-city data of Chicago, Paris and Zurich, respectively. FCN-4s-1-c achieved predictions that are 0.2%, 1.3% and 0.5% worse compared to the predictions of the FCN-4s-1 models, which were trained on single-city data of Chicago, Paris and Zurich, respectively (see Table 6.1, Table 6.2, Table 6.3). The semantic segmentations obtained from FCN-4s-1-c are also comparable to the semantic segmentations obtained from the three FCN-4s-1 models trained on single-city data (see Figure 6.1, Figure 6.2 and Figure 6.3). Moreover, FCN-4s-1-c appears to be able to connect some road fragments better than the FCN-4s-1 model trained on Paris data (see subfigure (f) in Figure 6.2) and the FCN-4s-1 model trained on Zurich data (see subfigure (f) in Figure 6.3).

**Predicting Tokyo**

The fully trained FCN-4s-1-c model and the three fully trained FCN-4s-1 models (trained on single-city data of Chicago, Zurich and Paris) were also used to predict aerial images of Tokyo. Aerial images of Tokyo were not included in any ground truth data used for training FCN models. Figure 6.5 shows the semantic segmentations for a representative aerial image of Tokyo from FCN-4s-1-c (subfigure (b)) and the three FCN-4s-1 models trained on single-city data of Chicago (subfigure (c)), Paris (subfigure (d)) and Zurich (subfigure (e)), respectively.
Figure 6.5: Predictions of FCN-4s-1 models for an aerial image of Tokyo. (a) shows the aerial image. (b), (c) and (d) show the semantic segmentation obtained from FCN-4s-1 trained on single-city data of Chicago, Paris and Zurich, respectively. (e) shows the semantic segmentation obtained from FCN-4s-1-c trained simultaneously on combined data of Chicago, Paris and Zurich. Neither of the FCN-4s-1 models was trained on data of Tokyo. Pixels that are labeled as building or road are overlay with red or blue color, respectively. Pixels that are labeled as background are not overlay with any color. Ground sampling distance of the aerial image of Tokyo is 12.1 cm.
Chapter 7

Discussion

This chapter first discusses the quality, applications and limitations of the new ground truth datasets, which were generated by the semi-automatic approach discussed earlier in this work. Second, this chapter discusses the predictive performances of the fully convolutional networks (FCNs), which were used to learn the individual city structures of Chicago, Paris and Zurich. Third, the performance improvements of the FCN-4s-1 models and FCN-4s-2 models over the FCN-8s models are discussed. Fourth, the predictive performance of FCN-4s-1-c model is discussed, which was trained simultaneously on combined ground truth data of Chicago, Paris and Zurich.

7.1 New Ground Truth Data

The new ground truth datasets, which were generated in this work, appear to be largely accurate and precise for all three cities Chicago, Paris and Zurich (see Figure 4.1, Figure 4.2 and Figure 4.3). Each of the ground truth datasets generated in this work is between 2 and 30 times bigger than the Vaihingen and/or Potsdam datasets of the ISPRS benchmark [28], which are currently being used to learn the semantic segmentation of aerial images (see Table 4.1). Moreover, the newly generated ground truth data can be used to effectively learn the semantic segmentation of aerial images with neural networks for all three cities (see Table 6.1 for Chicago, Table 6.2 for Paris and Table 6.3 for Zurich).

Thus, as a result, the semi-automatic generation of new ground truth data is suitable for the generation of large, accurate and diverse new ground truth datasets of very-high resolution (VHR) aerial images and pixel-wise object labels. The new ground truth data can be effectively used to learn the semantic segmentation of aerial images. Ground truth data generated by this semi-automatic approach overcomes the limitations posed by the relatively small size of benchmark datasets to the learning of the semantic segmentation from VHR aerial images. Furthermore, this approach enables
the generation of diverse ground truth datasets that can be used to train neural networks that generalize well to data from geographic regions that have not been used to train these models.

However, there are two aspects that should be put in consideration when this semi-automatic approach is used to generate new ground truth datasets. First, buildings that are pictured in aerial images can appear from the side. Aerial images provided by online map services have been taken by airborne platforms, which fly over a certain area and capture one aerial image after another until eventually the whole area of interest has been photographed. But usually it’s only the point captured by the middle of the lens of a camera that is photographed from a truly orthogonal perspective. Objects that are pictured at the edge of an aerial images usually appear from the side. Figure 7.1 (a) illustrates this issue by showing four buildings in the same aerial image that all appear from different angles. This "angle issue" rarely occurs in the case of roads and is not pronounced in the case of low buildings. However, in general, the "angle issue" is rather common. Figure 7.1 (b) shows an aerial image of Chicago that suffers from the "angle issue.

The "angle issue" is difficult to treat. Even if it was known at which angle a building in an aerial image is depicted, it is not clear how to make use of this information. If the pixel-wise object labels of the building were adjusted, the ground truth for this building would not be correct any more in the sense that it would not correspond to the geographic coordinates of the layout of the building at ground level. But the layout of the building at ground level is important for example for the generation of maps.

Second, for generating new ground truth data, this work determines the widths of roads by specifying an a-priory default road width for each highway tag that is used by OpenStreetMap (OSM). Because the widths of real roads are not really constant, this leads to ground truth data that can
be inaccurate at road edges. As a consequence, the newly generated ground truth data in this work is less accurate for roads than it is for buildings. In particular, this is the case for the newly generated ground truth datasets for Paris and Zurich. Another method to infer road widths is described in Mattyus et al. [22]. Mattyus et al. infer road widths from VHR aerial images and geographic coordinates of road centerlines by using a Markov random field.

Interestingly, although the newly generated ground truth data suffers from the "angle issue" issue and from inaccurate road labels, it is still possible to learn the semantic segmentation of buildings and roads in aerial images effectively. Minh et al. reported that missing or inaccurate object labels do not seem to affect the predictive power of the neural networks severely, which they trained for the semantic segmentation of aerial images, as long as there are enough correct object labels available in the ground truth data [24]. This seems to correspond to the findings in this work that it is possible to effectively learn the semantic segmentation of aerial images even in the presence of some inaccurate object labels.

7.2 Learning from Single Cities

Each model (FCN-8s, FCN-4s-1 and FCN-4s-2), after having been trained on single-city data, yielded reasonable semantic segmentations for aerial images of the same city (see Figure 6.1, Figure 6.2, Figure 6.3 and see also $F_1$ scores in Table 6.1, Table 6.2 and Table 6.3).

Chicago, Paris and Zurich

Each model (FCN-8s, FCN-4s-1 and FCN-4s-2) achieved the best predictive performance for Chicago, followed by Zurich and Paris. The city structure of Chicago thus appears to be easier to learn and predict than the city structure of Paris and Zurich. This might be because Chicago has a more regular city structure than Paris and Zurich, because the ground truth data for Chicago is better than the ground truth data for Paris and Zurich or a combination of both. The differences in predictive power between Paris and Zurich were mainly due to the predictions for buildings, which were some 15% better for Zurich than for Paris (comparison based on $F_1$ building scores). This might be because the buildings of Paris have more complex architectures than the buildings of Zurich. Many buildings of Zurich have rather simple, rectangular shapes, which might be easier to predict than the complex layouts of the buildings of Paris.

All FCN models had difficulties to connect some road fragments to each other. This happened for example if roads in aerial images were covered partly by large trees. Example of such situations can be seen in subfigure (c), (d) and (e) in 6.3.
Learning Buildings vs. Learning Roads

In the case of Chicago and Zurich, each model (FCN-8s, FCN-4s-1 and FCN-4s-2) predicted buildings better than roads with building predictions being between 4.0% and 8.9% better than road predictions. This might be because, in general, the newly generated ground truth data in this work is more accurate for buildings than for roads. In the case of Paris however, it’s the other way round and predictions for roads were between 4.9% and 5.8% better than predictions for buildings (see Table 6.1, Table 6.2, Table 6.3).

This is surprising since the newly generated ground truth data for Paris appears to be very accurate for buildings and less accurate for roads (see Figure 4.2). In particular road widths are not constant in Paris which frequently leads to ground truth road labels that are too narrow. Apparently, the neural networks trained on data of Paris learned to label roads as “too narrow” based on the ground truth data (see Figure 6.2). This is an interesting result since it shows how powerful neural network models can be in extracting those features from raw image data that are discriminative for the task that they are given to learn.

When training from scratch, interestingly, it seems that FCNs learn buildings faster than roads. This trend was observed in all FCN models that were trained in this work. Figure 7.2 shows out-sample predictions of an FCN-8s for a representative area of Zurich during the first 20,000 training iterations while learning on data of Zurich from scratch. Obviously buildings are learned faster than roads. For the first 10,000 or so training iterations, the FCN-8s made virtually no road predictions. After 10,000 or so training iterations the FCN-8s started to predict pixels as building, road and background. Learning buildings might be faster than learning roads because building labels are more accurate than road labels or buildings might be easier to distinguish in an aerial image since they have more compact shapes than roads. Figure 7.3 shows the corresponding $F_1$, precision and recall scores obtained during learning for each of the object labels building, road and background and for an average over the three labels (in addition, the training loss is depicted on a logarithmic scale in the bottom of the figure).

In total, the semantic segmentations obtained from the fully trained FCN models, which were trained on single-city data for each of the cities Chicago, Paris and Zurich separately, demonstrate that it is possible to effectively learn the semantic segmentation of aerial images with neural networks from the new ground truth datasets, which were generated in this work.

7.3 Three Different FCN Architectures

All FCN-4s-1 models, trained on data of single cities, yielded better predictions than the respective FCN-8s models, which were trained on data of
Figure 7.2: Out-sample predictions of FCN-8s during training on Zurich data for an aerial image of Zurich. Numbers at the bottom of subfigures indicate after how many training iterations a prediction was obtained. Pixels that are labeled as building or road are overlay with red or blue color, respectively. Pixels that are labeled as background are not overlay with any color.

Figure 7.3: Learning curves of an FCN-8s model trained on Zurich data.
the same city. Thus the FCN-4s-1 architecture appears to be better suited than the FCN-8s architecture to learn the semantic segmentation of aerial images. This is probably due to the additional network connection from the shallow layer Pool2 that FCN-4s-1 has and FCN-8s doesn’t, which is the main difference between these two neural networks (see Figure 5.1 and Figure 5.2). When FCN-4s-1 upsamples data back to the resolution of the original aerial image, it fuses data from the shallow neural network layer (Pool2) with a deep neural network layer to improve localization. Fusing spatial information from layer Pool2 obviously helps FCN-4s-1 to achieve better semantic segmentation than FCN-8s. The better prediction of the FCN-4s-1 models are also apparent from Figure 6.1, Figure 6.2 and Figure 6.3. FCN-4s-1 was in particular able to draw roads more accurately than FCN-8s (see subfigures (c) and (d) in Figure 6.1 and Figure 6.3).

All FCN-4s-2 models, trained on data of single cities, yielded better or equal predictions compared to the respective FCN-8s models, which were trained on data of the same city. FCN-4s-2 thus appears to be better or equally suited to learn the semantic segmentation of aerial images as FCN-8s. FCN-4s-2 also yielded better predictions than FCN-4s-1 in the case of Chicago and Zurich. For Paris, however FCN-4s-2 predicted worse than FCN-4s-1. FCN-4s-2 also fuses data from layer Pool2 like FCN-4s-1. In contrast to FCN-4s-1 however, FCN-4s-2 uses a very high number of feature channels when performing in-network upsampling. However, whether these additional feature channels help for the semantic segmentation of aerial images is not clear. FCN-4s-2 did outperform FCN-4s-1 when trained on data of Chicago and Zurich, respectively. But the improvements were rather small with 0.2% and 0.3%, respectively. In the case of Paris, it was FCN-4s-1, which achieved better predictions than FCN-4s-2. FCN-4s-2 is with about 291 million learnable parameters significantly larger than FCN-4s-1 and FCN-8s (both contain about 134 million learnable parameters). One training iterations takes accordingly more time for FCN-4s-2 than for FCN-8s or FCN-4s-1.

Regarding the performance improvements achieved by FCN-4s-1 relative to FCN-8s and the relatively short time it needs for one training iteration, FCN-4s-1 is the most efficient FCN out of the three FCN architectures, which were investigated in this work. However, overall the performance improvements of FCN-4s-1 and FCN-4s-2 over FCN-8s are between 0.0% and 1.7% and thus relatively small.

7.4 Learning Simultaneously from Multiple Cities

FCN-4s-1-c, which was trained simultaneously on combined ground truth data of Chicago, Paris and Zurich, achieved semantic segmentations of aerial images of all three cities (Chicago, Paris and Zurich) which are comparable
to the semantic segmentations obtained from the respective models trained on single-city data (see Table 6.1, Table 6.2, Table 6.3 and Figure 6.1, Figure 6.2, Figure 6.3). In contrast, whenever an FCN-4s-1 model was trained on data of one city and used to predict aerial images of another city, the predictions were sketchy and inaccurate (see Figure 6.4). Moreover, for buildings, FCN-4s-1-c achieves semantic segmentations of aerial images of Tokyo (which were not use to train FCN models in this work) that are clearly better than the semantic segmentations obtained by the three other FCN-4s-1 models, which were trained on single-city data only.

These results demonstrate that FCN-4s-1-c, which is trained on a large and diverse ground truth dataset composed of aerial images of different cities, can be used for the semantic segmentation of aerial images of different geographic regions that have not been used to train FCN-4s-1-c.
Chapter 8

Conclusions and Future Work

This work shows that combining publicly available very-high resolution (VHR) aerial images and publicly available geographic coordinates of buildings and highways can be used to generate new ground truth datasets that can be used to effectively train neural networks for the semantic segmentation of aerial images. Moreover, generating huge and in particular diverse ground truth datasets led to a neural network which was able to make reasonable semantic segmentations for buildings in aerial images of a city that has not been used to train this model. Furthermore, this work suggests two modifications to the base FCN architecture described by Long et al. [20]; each of which yield a minor performance improvement relative to the base FCN architecture when learning the semantic segmentation of aerial images (regarding $F_1$ average scores).

Considering the results obtained in this work, it is possible to interpret that the size and the diversity of ground truth data, which is used for learning, is critically important for the generalization ability of neural network models that learn the semantic segmentation of aerial images. This work hopes thus to contribute to the automatic generation of maps from raw aerial image data by providing a simple, semi-automatic approach to generate large, accurate and diverse ground truth datasets that can be used to train neural networks for the semantic segmentation of aerial images.

It might be worth analysing the newly generated ground truth datasets with other machine learning methods that aim at the semantic segmentation of images. At the same time it might be worth investigating how the size and diversity of new ground truth data affects the generalization performance of neural networks to aerial images of different geographic regions.

In conclusion, this work shows that publicly available remote sensing data can be used to effectively train neural networks to recognize complex patterns in this data. Huge amounts of remote sensing data are available
nowadays and analyzing this data to extract relevant information surely adds value to many fields of application including automatic map generation, cartography, climate change monitoring, biodiversity monitoring, flood prediction and disaster warning or disaster relieve.
Bibliography


Appendix A

Alternative CNN architecture

This section describes and discusses the neural network model FCN-residual. FCN-residual is introduced in this work and was used to learn the semantic segmentation of aerial images of Chicago. However, it yielded predictions that were slightly worse than the predictions of other FCN models trained on Chicago. For this reason, FCN-residual is described in the Appendix of this work and not in one of the main chapters.

The architecture of FCN-residual is based on the architecture of so-called residual networks, which were first described in He et al. [9]. Building on work of Srivastava et al. [36], He et al. observed a degradation of training accuracy in very deep neural networks when these networks started to converge. To ease training of very deep neural networks, He et al. suggested deep residual nets. They reformulate neural network layers to explicitly learn residual functions instead of the original functions and call such neural network architectures residual networks. The deep residual nets of He et al. were very successful and won 1st places in the 2015 ImageNet detection, 2015 ImageNet localization\(^1\), 2015 COCO detection and 2015 COCO segmentation\(^2\).

In this work, a combination of the residual network architecture suggested by He et al. [8] and the neural network architecture proposed by Long et al. [20] is used to learn the semantic segmentation of aerial images of Chicago. For this purpose the architecture of FCN-4s-1 was transformed into an FCN-residual architecture. Whenever it was possible, two subsequent convolutional layers, each followed by a rectified linear unit, were transformed into the building blocks of residual learning as described in He et al. In addition, the number of convolutional layers was increased like in He et al. Figure A.1 illustrates the architecture of FCN-residual.

\(^1\)http://image-net.org/challenges/LSVRC/2015/

\(^2\)http://mscoco.org/dataset/#detections-challenge2015
Figure A.1: FCN-residual architecture. The neurons of each layer are arranged as a three-dimensional array. The size of each layer is indicated by the numbers in brackets \((D, H, W)\). \(D\) refers to the number of feature channels and \(H\) and \(W\) refer to the spatial dimension of the layer. FCN-8s contains 170,559,420 learnable parameters.
Owing to the skip architecture of FCNs, it was not possible to convert all convolutional layers into the building blocks of residual learning. This is because the skip architecture of the FCNs used in this work fuses data from different neural network layers. Before fusing two data layers, the layers need to have the same size. So usually one layer is cropped with regard to the size of the other layer. This can cause the receptive fields of the two data layers that will be fused to be shifted or distorted with respect to each other. Transforming all convolutional layers of FCN-residual into the building blocks of residual neural networks would cause such mismatches of receptive fields. Thus five layers of FCN-residual were not converted into the building blocks of residual networks (see Figure A.1). These layers are learnt as usual, this is, in a non-residual-learning fashion. FCN-residual is learned from scratch on newly generated ground truth data of Chicago.

FCN-residual yields predictions that are 0.3%, 1.4% and 1.7% worse than the predictions of FCN-8s, FCN-4s-1 and FCN-4s-2 trained on Chicago data (regarding $F_1$ scores) (see Table 6.1). FCN-residual is thus the model with the worst performance, which was trained and evaluated on Chicago. However, the performance differences between FCN-residual and the other FCN models is rather small (see also Figure A.2).
One possible explanation for the comparably low performance of FCN-residual is that not all layers of FCN-residual have been transformed into the units of residual learning as described in He et al. [8]. Having “usual” and residual FCN layers mixed in one neural network architecture might hamper optimization by gradient descent.
Appendix B

Learning Building Corners

Semantic segmentation of aerial images means to assign an object label like building or road to each pixel in an aerial image. Maps, however, are usually stored in a vector format. This is, objects that appear in maps are typically not encoded by pixel-wise object labels but rather by the coordinates of their corner points. When displaying a map, for example on a computer screen, these corner points are then connected to silhouettes of the objects they belong to by the software which is used to draw the map.

Eventually, it would be desirable to transform the semantic segmentations of aerial images into vector maps. Vector maps have the advantage of capturing the contours of objects like buildings and roads more accurately. This is, because in the semantic segmentations of aerial images, object contours are typically somewhat blurred. One approach to generate vector maps from aerial images would be to learn the corner points of objects like buildings and streets. If corner points can be predicted with high confidence, they could be assigned to objects in aerial images and eventually they could be stored as vectors in a vector map representation of the aerial image.

In this work, a preliminary work was conducted, which aims at detecting corners of buildings in aerial images of Chicago. For this purpose, a small training dataset of very-high resolution (VHR) aerial images and corresponding pixel-wise object label files was generated (consisting of roughly 75 million pixels). VHR aerial images of Chicago have already been downloaded from Google Maps and coordinates of building corners have already been downloaded from OpenStreetMap. Pixel-wise object labels were generated by plotting building corners as circles of predefined size into an empty image, which had the same size as the aerial image for which the building labels were aimed for. A fully convolutional network with FCN-8s architecture was then trained on this newly generated ground truth data from scratch for 30,000 iterations using a learning rate of $5 \times 10^{-9}$. The pixel-wise building corner predictions obtained from this FCN-8s model are shown in Figure B.1. Subfigure (a) shows an aerial image of Chicago, subfigure (b)
shows the aerial image overlay with the building corner ground truth labels and subfigure (c) shows the building corner prediction of FCN-8s.

Figure B.1: Aerial image (a), ground truth building corner labels (b) and building corner prediction by an FCN-8s model (c) for Chicago. One sub-region of the aerial image is magnified. Pixels that are labeled as building corner are overlay with red. Pixels that are labeled as background are not overlay with any color.

The predictions of building corners were surprisingly good. Many building corners were captured by this FCN-8s prediction and could possibly be used to try to determine buildings in the depicted aerial image. But not all building corners have been captured and some corner predictions were rather blurry. However, the predictive performance of this FCN model could probably be improved by changing hyperparameters for learning. In the preliminary work conducted in this work, exactly one learning configuration was used, which worked out-of-the-box and led to the predictions shown in Figure B.1.

Assigning building corners to buildings in aerial images is another task and probably not trivial. To assign a building corner to a building, one has to know roughly where the buildings in an aerial image are located. Otherwise it is not clear how to connect the building corners with each other such as to enclose the buildings. To get information about the location of buildings in aerial images, one could use the semantic segmentation approach of aerial images, which is described and discussed in this work to detect buildings and roads in aerial images. Combining predictions of building corners with
predictions of the location of buildings could facilitate the generation of vector maps from aerial images. FCNs might be helpful in this context because they can generate predictions for both, location of buildings and location of building corners.
Appendix C

Supplementary Tables
<table>
<thead>
<tr>
<th>Parameter layer</th>
<th>No. of filters</th>
<th>Filter sizes</th>
<th>No. of bias parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>conv</td>
<td>64</td>
<td>(3, 3, 3)</td>
<td>64</td>
</tr>
<tr>
<td>conv</td>
<td>128</td>
<td>(64, 3, 3)</td>
<td>128</td>
</tr>
<tr>
<td>conv</td>
<td>256</td>
<td>(128, 3, 3)</td>
<td>256</td>
</tr>
<tr>
<td>conv</td>
<td>512</td>
<td>(256, 3, 3)</td>
<td>512</td>
</tr>
<tr>
<td>conv</td>
<td>1024</td>
<td>(512, 3, 3)</td>
<td>1024</td>
</tr>
<tr>
<td>conv</td>
<td>2048</td>
<td>(1024, 1, 1)</td>
<td>2048</td>
</tr>
<tr>
<td>deconv</td>
<td>3</td>
<td>(3, 4, 4)</td>
<td>-</td>
</tr>
<tr>
<td>conv</td>
<td>3</td>
<td>(512, 1, 1)</td>
<td>3</td>
</tr>
<tr>
<td>deconv</td>
<td>3</td>
<td>(3, 16, 16)</td>
<td>-</td>
</tr>
</tbody>
</table>

Table C.1: Learnable parameters of FCN-8s. Learnable parameters are contained in convolutional layers and deconvolutional layers. The second column of the table indicates the number of filters of each layer. The number of filters corresponds to the number of feature channels ($D$), which each layer outputs. The third column indicates the filter sizes ($D' \times h \times w$) of each layer. $D'$ is the number of input feature channels and $h \times w$ is the two-dimensional, spatial dimensions of the filters. The total number of learnable parameters in FCN-8s is 134,277,737.

<table>
<thead>
<tr>
<th>Parameter layer</th>
<th>No. of filters</th>
<th>Filter sizes</th>
<th>No. of bias parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>conv</td>
<td>64</td>
<td>(3, 3, 3)</td>
<td>64</td>
</tr>
<tr>
<td>conv</td>
<td>128</td>
<td>(64, 3, 3)</td>
<td>128</td>
</tr>
<tr>
<td>conv</td>
<td>256</td>
<td>(128, 3, 3)</td>
<td>256</td>
</tr>
<tr>
<td>conv</td>
<td>512</td>
<td>(256, 3, 3)</td>
<td>512</td>
</tr>
<tr>
<td>conv</td>
<td>1024</td>
<td>(512, 3, 3)</td>
<td>1024</td>
</tr>
<tr>
<td>conv</td>
<td>2048</td>
<td>(1024, 1, 1)</td>
<td>2048</td>
</tr>
<tr>
<td>deconv</td>
<td>3</td>
<td>(3, 4, 4)</td>
<td>-</td>
</tr>
<tr>
<td>conv</td>
<td>3</td>
<td>(512, 1, 1)</td>
<td>3</td>
</tr>
<tr>
<td>deconv</td>
<td>3</td>
<td>(3, 16, 16)</td>
<td>-</td>
</tr>
<tr>
<td>conv</td>
<td>3</td>
<td>(128, 1, 1)</td>
<td>3</td>
</tr>
</tbody>
</table>

Table C.2: Learnable parameters of FCN-4s-1. Learnable parameters are contained in convolutional layers and deconvolutional layers. The second column of the table indicates the number of filters of each layer. The number of filters corresponds to the number of feature channels ($D$), which each layer outputs. The third column indicates the filter sizes ($D' \times h \times w$) of each layer. $D'$ is the number of input feature channels and $h \times w$ is the two-dimensional, spatial dimensions of the filters. The total number of learnable parameters in FCN-4s-1 is 134,276,540.
### Table C.3: Learnable parameters of FCN-4s-2

Learnable parameters are contained in convolutional layers and deconvolutional layers. The second column of the table indicates the number of filters of each layer. The number of filters corresponds to the number of feature channels ($D$), which each layer outputs. The third column indicates the filter sizes ($D' \times h \times w$) of each layer. $D'$ is the number of input feature channels and $h \times w$ is the two-dimensional, spatial dimensions of the filters. The total number of learnable parameters in FCN-4s-2 is 290,867,008.

<table>
<thead>
<tr>
<th>Parameter layer</th>
<th>No. of filters</th>
<th>Filter sizes</th>
<th>No. of bias parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>conv1,1</td>
<td>64</td>
<td>(3, 3, 3)</td>
<td>64</td>
</tr>
<tr>
<td>conv1,2</td>
<td>64</td>
<td>(64, 3, 3)</td>
<td>64</td>
</tr>
<tr>
<td>conv2,1</td>
<td>128</td>
<td>(64, 3, 3)</td>
<td>128</td>
</tr>
<tr>
<td>conv2,2</td>
<td>256</td>
<td>(128, 3, 3)</td>
<td>256</td>
</tr>
<tr>
<td>conv3,1</td>
<td>256</td>
<td>(128, 3, 3)</td>
<td>256</td>
</tr>
<tr>
<td>conv3,2</td>
<td>256</td>
<td>(256, 3, 3)</td>
<td>256</td>
</tr>
<tr>
<td>conv4,1</td>
<td>512</td>
<td>(256, 3, 3)</td>
<td>512</td>
</tr>
<tr>
<td>conv4,2</td>
<td>512</td>
<td>(512, 3, 3)</td>
<td>512</td>
</tr>
<tr>
<td>conv5,1</td>
<td>512</td>
<td>(512, 3, 3)</td>
<td>512</td>
</tr>
<tr>
<td>conv5,2</td>
<td>512</td>
<td>(512, 3, 3)</td>
<td>512</td>
</tr>
<tr>
<td>conv6,1</td>
<td>4096</td>
<td>(512, 7, 7)</td>
<td>4096</td>
</tr>
<tr>
<td>conv6,2</td>
<td>4096</td>
<td>(4096, 1, 1)</td>
<td>4096</td>
</tr>
<tr>
<td>deconv1,1</td>
<td>1536</td>
<td>(1024, 7, 7)</td>
<td></td>
</tr>
<tr>
<td>deconv2,1</td>
<td>1792</td>
<td>(1024, 4, 4)</td>
<td></td>
</tr>
<tr>
<td>deconv3,1</td>
<td>1920</td>
<td>(3, 8, 8)</td>
<td></td>
</tr>
</tbody>
</table>
Table C.4: Learnable parameters of FCN-residual. Learnable parameters are contained in convolutional layers and deconvolutional layers. The second column of the table indicates the number of filters of each layer. The number of filters corresponds to the number of feature channels ($D$), which each layer outputs. The third column indicates the filter sizes ($D' \times h \times w$) of each layer. $D'$ is the number of input feature channels and $h \times w$ is the two-dimensional, spatial dimensions of the filters. The total number of learnable parameters in FCN-residual is 170,559,420.
Reduction of learning rate to $5 \times 10^{-10}$ & Reduction of learning rate to $5 \times 10^{-11}$ & Total number of learning iterations \\
FCN-8s, Chicago & 3 & 6 & 9 \\
FCN-8s, Paris & 9 & 12 & 14 \\
FCN-8s, Zurich & 6 & 10 & 11.8 \\
FCN-4s-1, Chicago & 3 & 6 & 11 \\
FCN-4s-1, Paris & 1 & 4 & 5.5 \\
FCN-4s-1, Zurich & 1 & - & 4.5 \\
FCN-4s-2, Chicago & 2 & 5 & 7.5 \\
FCN-4s-2, Paris & 1.5 & 4 & 6 \\
FCN-4s-2, Zurich & 1 & 3.5 & 4.5 \\
FCN-4s-1-c & 2 & 7 & 10 \\
FCN-residual, Chicago & 4 & 7 & 8 \\

Table C.5: Learning rates and total number of learning iterations that were used to train FCN models. Training of all FCN models started with a learning rate of $5 \times 10^{-9}$. The second and third column of the table indicate after how many training iterations the learning rate was reduced to $5 \times 10^{-10}$ and $5 \times 10^{-11}$, respectively. The last columns indicates for how many iterations FCN models were trained in total. In the case of FCN-4s-1 trained on Zurich data, reducing the learning rate to $5 \times 10^{-11}$ did not yield a further performance improvement. All numbers have to be multiplied by a factor of $10^4$.

<table>
<thead>
<tr>
<th></th>
<th>FCN-8s</th>
<th>FCN-4s-1</th>
<th>FCN-4s-2</th>
<th>FCN-4s-1-c</th>
<th>FCN-residual*</th>
</tr>
</thead>
<tbody>
<tr>
<td>$F_1$ average</td>
<td>0.816</td>
<td>0.827</td>
<td>0.829</td>
<td>0.825</td>
<td>0.813</td>
</tr>
<tr>
<td>$F_1$ building</td>
<td>0.842</td>
<td>0.849</td>
<td>0.850</td>
<td>0.848</td>
<td>0.842</td>
</tr>
<tr>
<td>$F_1$ road</td>
<td>0.801</td>
<td>0.809</td>
<td>0.809</td>
<td>0.805</td>
<td>0.792</td>
</tr>
<tr>
<td>$F_1$ background</td>
<td>0.805</td>
<td>0.823</td>
<td>0.828</td>
<td>0.821</td>
<td>0.805</td>
</tr>
<tr>
<td>Precision average</td>
<td>0.824</td>
<td>0.838</td>
<td>0.838</td>
<td>0.833</td>
<td>0.825</td>
</tr>
<tr>
<td>Precision building</td>
<td>0.853</td>
<td>0.863</td>
<td>0.867</td>
<td>0.859</td>
<td>0.847</td>
</tr>
<tr>
<td>Precision road</td>
<td>0.790</td>
<td>0.802</td>
<td>0.801</td>
<td>0.801</td>
<td>0.790</td>
</tr>
<tr>
<td>Precision background</td>
<td>0.830</td>
<td>0.848</td>
<td>0.846</td>
<td>0.840</td>
<td>0.838</td>
</tr>
<tr>
<td>Recall average</td>
<td>0.822</td>
<td>0.833</td>
<td>0.836</td>
<td>0.832</td>
<td>0.813</td>
</tr>
<tr>
<td>Recall building</td>
<td>0.834</td>
<td>0.841</td>
<td>0.839</td>
<td>0.841</td>
<td>0.840</td>
</tr>
<tr>
<td>Recall road</td>
<td>0.845</td>
<td>0.856</td>
<td>0.855</td>
<td>0.849</td>
<td>0.821</td>
</tr>
<tr>
<td>Recall background</td>
<td>0.786</td>
<td>0.802</td>
<td>0.813</td>
<td>0.807</td>
<td>0.779</td>
</tr>
</tbody>
</table>

Table C.6: $F_1$ score, precision and recall of different FCN models trained and evaluated on ground truth data of Chicago. FCN-4s-1-c indicates a neural network model which was trained on ground truth data of Chicago, Paris and Zurich mutually and evaluated on ground truth data of Chicago. FCN-residual* is a neural network that aims at residual learning as described in He et al. [9]. FCN-residual trained on Chicago ground truth data yields predictions that are slightly worse than the predictions obtained from the other FCN-models. FCN-residual is described and discussed in Appendix A.
<table>
<thead>
<tr>
<th></th>
<th>FCN-8s</th>
<th>FCN-4s-1</th>
<th>FCN-4s-2</th>
<th>FCN-4s-1-c</th>
</tr>
</thead>
<tbody>
<tr>
<td>$F_1$ average</td>
<td>0.764</td>
<td>0.767</td>
<td>0.764</td>
<td>0.754</td>
</tr>
<tr>
<td>$F_1$ building</td>
<td>0.751</td>
<td>0.755</td>
<td>0.759</td>
<td>0.742</td>
</tr>
<tr>
<td>$F_1$ road</td>
<td>0.809</td>
<td>0.810</td>
<td>0.808</td>
<td>0.801</td>
</tr>
<tr>
<td>$F_1$ background</td>
<td>0.732</td>
<td>0.737</td>
<td>0.725</td>
<td>0.718</td>
</tr>
<tr>
<td>Precision average</td>
<td>0.774</td>
<td>0.777</td>
<td><strong>0.780</strong></td>
<td>0.762</td>
</tr>
<tr>
<td>Precision building</td>
<td>0.787</td>
<td>0.791</td>
<td>0.775</td>
<td>0.767</td>
</tr>
<tr>
<td>Precision road</td>
<td>0.774</td>
<td>0.774</td>
<td><strong>0.781</strong></td>
<td>0.775</td>
</tr>
<tr>
<td>Precision background</td>
<td>0.760</td>
<td>0.767</td>
<td><strong>0.783</strong></td>
<td>0.744</td>
</tr>
<tr>
<td>Recall average</td>
<td>0.774</td>
<td><strong>0.777</strong></td>
<td>0.768</td>
<td>0.765</td>
</tr>
<tr>
<td>Recall building</td>
<td>0.724</td>
<td>0.728</td>
<td><strong>0.749</strong></td>
<td>0.727</td>
</tr>
<tr>
<td>Recall road</td>
<td>0.874</td>
<td><strong>0.878</strong></td>
<td>0.866</td>
<td>0.863</td>
</tr>
<tr>
<td>Recall background</td>
<td>0.723</td>
<td><strong>0.724</strong></td>
<td>0.690</td>
<td>0.705</td>
</tr>
</tbody>
</table>

Table C.7: $F_1$ score, precision and recall of different FCN models trained and evaluated on ground truth data of Paris. FCN-4s-1-c indicates a neural network model which was trained on ground truth data of Chicago, Paris and Zurich simultaneously and evaluated on ground truth data of Paris.

<table>
<thead>
<tr>
<th></th>
<th>FCN-8s</th>
<th>FCN-4s-1</th>
<th>FCN-4s-2</th>
<th>FCN-4s-1-c</th>
</tr>
</thead>
<tbody>
<tr>
<td>$F_1$ average</td>
<td>0.796</td>
<td>0.810</td>
<td><strong>0.813</strong></td>
<td>0.805</td>
</tr>
<tr>
<td>$F_1$ building</td>
<td>0.892</td>
<td><strong>0.904</strong></td>
<td><strong>0.904</strong></td>
<td>0.902</td>
</tr>
<tr>
<td>$F_1$ road</td>
<td>0.823</td>
<td>0.815</td>
<td><strong>0.826</strong></td>
<td>0.820</td>
</tr>
<tr>
<td>$F_1$ background</td>
<td>0.672</td>
<td><strong>0.711</strong></td>
<td>0.708</td>
<td>0.694</td>
</tr>
<tr>
<td>Precision average</td>
<td>0.810</td>
<td><strong>0.827</strong></td>
<td>0.822</td>
<td>0.818</td>
</tr>
<tr>
<td>Precision building</td>
<td>0.896</td>
<td>0.893</td>
<td><strong>0.899</strong></td>
<td>0.895</td>
</tr>
<tr>
<td>Precision road</td>
<td>0.833</td>
<td>0.829</td>
<td>0.830</td>
<td><strong>0.838</strong></td>
</tr>
<tr>
<td>Precision background</td>
<td>0.702</td>
<td><strong>0.758</strong></td>
<td>0.737</td>
<td>0.719</td>
</tr>
<tr>
<td>Recall average</td>
<td>0.799</td>
<td>0.801</td>
<td><strong>0.809</strong></td>
<td>0.805</td>
</tr>
<tr>
<td>Recall building</td>
<td>0.892</td>
<td><strong>0.915</strong></td>
<td>0.910</td>
<td>0.909</td>
</tr>
<tr>
<td>Recall road</td>
<td>0.816</td>
<td>0.805</td>
<td><strong>0.825</strong></td>
<td>0.805</td>
</tr>
<tr>
<td>Recall background</td>
<td>0.688</td>
<td>0.684</td>
<td>0.693</td>
<td><strong>0.701</strong></td>
</tr>
</tbody>
</table>

Table C.8: $F_1$ score, precision and recall of different FCN models trained and evaluated on ground truth data of Zurich. FCN-4s-1-c indicates a neural network model which was trained on ground truth data of Chicago, Paris and Zurich simultaneously and evaluated on ground truth data of Zurich.

80
Declaration of Originality

The signed declaration of originality is a component of every semester paper, Bachelor’s thesis, Master’s thesis and any other degree paper undertaken during the course of studies, including the respective electronic versions.

Lecturers may also require a declaration of originality for other written papers compiled for their courses.

I hereby confirm that I am the sole author of the written work here enclosed and that I have compiled it in my own words. Parts excepted are corrections of form and content by the supervisor.

Title of work (in block letters):

Learning City Structures from Online Maps

Authored by (in block letters):
For papers written by groups the names of all authors are required.

Name(s):

First name(s):

Kaiser

Pascal

With my signature I confirm that
- I have committed none of the forms of plagiarism described in the Citation etiquette information sheet.
- I have documented all methods, data and processes truthfully.
- I have not manipulated any data.
- I have mentioned all persons who were significant facilitators of the work.
- I am aware that the work may be screened electronically for plagiarism.
- I have understood and followed the guidelines in the document Scientific Works in Mathematics.

Place, date:

Zurich, 22.08.2016

Signature(s):

P. Kaiser

For papers written by groups the names of all authors are required. Their signatures collectively guarantee the entire content of the written paper.