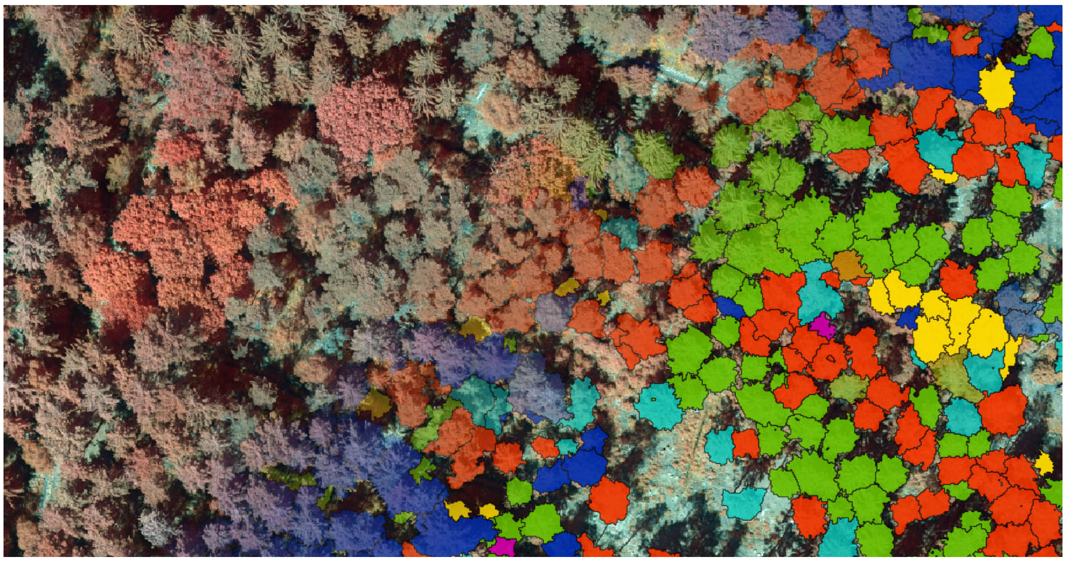
GEORG-AUGUST-UNIVERSITÄT GÖTTINGEN

Master Thesis

# Tree Detection in Remote Sensing Imagery

# Baumerkennung in Fernerkundungs-Bildmaterial



prepared by

**Maximilian Ulrich Freudenberg**

born in Heidelberg, Germany

at the Third Institute of Physics

in the Department of Computational Neuroscience

in cooperation with the Chair of Forest Inventory and Remote Sensing

from the Burckhardt Institute, Faculty of Forest Sciences and Forest Ecology

| | |
|---|---|
| Thesis Period: | 5th November 2018 until 28th March 2019 |
| First Referee: | Prof. Dr. Christoph Kleinn |
| Second Referee: | Prof. Dr. Florentin Wörgötter |

## Abstract

In this master thesis three subjects will be studied: First, the segmentation of tree cover in high resolution satellite imagery from a region in and around Bengaluru, India. Second, the position detection of single palm trees on large areas near Jambi, Indonesia, also on satellite imagery. Third, the classification of tree species in aerial images in a study area near Gartow, Lower Saxony. In order to solve these tasks, *deep learning* is used with two types of neural networks. For the segmentation of tree cover and for the position detection, a so-called “U-Net” is used, and for the classification a “ResNet”. These networks, in conjunction with the approaches used in this thesis, perform equal or better than existing methods. In the segmentation task, the U-Net reaches an intersection over union value of 90%, which cannot be reached by “classical” approaches like support vector machines. The method for individual palm tree detection developed in this thesis is one order of magnitude faster than the state of the art method, while reaching comparable accuracy. Lastly, the ResNet allows to detect and distinguish more tree species groups than methods that do not employ deep learning. Consequently, further research in the area of remote sensing in combination with deep learning is promising.

## Zusammenfassung

In dieser Masterarbeit werden drei Themen behandelt: Erstens die Erkennung von baumbedeckten Gebieten in hochaufgelösten Satellitenbildern in einer Region in und um Bengaluru, Indien. Zweitens, die Positionsbestimmung einzelner Palmen auf großen Flächen nahe Jambi, Indonesien, ebenfalls auf Satellitenbildern. Drittens, die Unterscheidung von Baumarten auf Luftbildern in einem Untersuchungsgebiet bei Gartow, Niedersachsen. Um diese Problemstellungen zu lösen, wird *Deep Learning* mit zwei verschiedenen Typen neuronaler Netze eingesetzt. Zur Kartierung der Baumbedeckung und zur Positionsbestimmung wird ein sogenanntes “U-Net” verwendet, bei der Klassifikation ein “ResNet”. Dabei zeigt sich, dass diese Netzwerke und die hier gewählten Methoden bestehenden Ansätzen überlegen, oder mindestens ebenbürtig sind. Bei der Erkennung von Baumbedeckung erreicht das U-Net eine Genauigkeit von 90%, was von “klassischen” Methoden wie Support Vector Machines nicht erreicht wird. Die hier entwickelte Methode zur Positionsbestimmung von Palmen ist bei vergleichbarer Genauigkeit eine Größenordnung schneller als die bisher beste Methode. Das ResNet schließlich erlaubt, mehr Baumarten bzw. Art-Gruppen zu unterscheiden, als nicht auf Deep Learning basierende Methoden. Dem entsprechend erscheint eine weitergehende Untersuchung im Lichte der hier präsentierten Ergebnisse lohnenswert.

**Keywords:** Deep Learning, Semantic Segmentation, Remote Sensing, Tree Species Detection, Satellite Imagery, U-Net, ResNet

## Acknowledgments

First of all I want to thank my supervisors and referees Professor Kleinn and Professor Wörgötter, who guided me through this thesis and gave me the freedom to explore own ideas. Throughout the entire time, they provided valuable feedback and always had time for answering questions. Their experience was invaluable during the preparation and submission of the publication related to this thesis.

I also want to thank:

Nils Nölke for all the interesting research questions he posed of which I could complete just a fraction. He introduced me to the field of remote sensing and our discussions always ended with new ideas and solutions.

Alejandro Agostini for motivating me to write the publication related to this thesis. He helped me to structure my thoughts and to improve the quality of my writing.

Kira Urban for providing the data used in the publication and for giving me the initial idea of detecting palms.

Paul Magdon, Hans Fuchs and Lutz Fehrmann for discussions on general remote sensing topics, for the time consuming pre-processing of the data I used and for the generation of training data in the field and *in silico*.

Max Vossel and Jan Plank for their comments regarding the text.

Vladimir Belov for the discussions and insights with respect to deep learning.

The entire Department of Forest Inventory and Remote Sensing for an always helpful atmosphere and the interest in my work.

The Department of Computational Neuroscience for being my anchor in the world of physics and for all the answers I received with regard to deep learning.

Annabelle Ritter and my family, who patiently and unconditionally supported me during the writing of this thesis and the years of studying before.

# Contents

# 1. Introduction

This master thesis applies *deep learning* methods from computer vision in the field of remote sensing for forestry. In remote sensing, the subject of study is the entire earth or regions of it. On the size scale of research objects, remote sensing is thus between astrophysics with its huge galaxies and bio- or particle-physics, where single molecules or even the compounds of atoms are studied. By the methods applied, remote sensing is closely related to physics as it takes advantage of satellites, airplanes or drones in order to capture imagery in a spectrum ranging from radar, over infrared and the visible light, to ultraviolet light. Light detection and ranging (Lidar) techniques are used as well. The information captured from above can be fused with measurements taken on the ground in order to derive models for processes taking place in the environment. This is similar to physics, where the subject of interest is probed by different means with the aim of arriving at an insight into the underlying mechanisms. The processes and objects observed in remote sensing range from the tree growth in a certain area [59], over the development of an entire city over several decades [26] to global plant monitoring [14]. The last point is of particular interest, as it enables biomass estimations [51] and thus to predict the nature's capacity to absorb $CO_2$. This in turn improves the accuracy of climate models and their predictions. Therefore, remote sensing indirectly wields an influence over political decisions on a global scale, like the Kyoto Protocol or the Paris Agreement and it plays a key role in creating an awareness for global trends like deforestation or melting glaciers and polar caps.

The global or regional scale of the problems addressed by remote sensing implies capturing large amounts of data. As technology advances and sensor resolution increases, processing all the data becomes more and more challenging. Consequently, accurate, robust and fast image processing techniques are required. This is where *deep learning* comes into play. Deep learning is a subcategory of machine learning, which has recently gained reputation for its remarkable results in various tasks. In deep learning, so-called neural networks are used to process information. There are several kinds of neural networks, which have their specific strengths and weaknesses. Recurrent Neural Networks (RNNs) and their derivatives (e.g. LSTMs) are able to process text and speech [55, 25]. For example, Google's translate software employs an LSTM network, which allows to translate between any two languages with remarkable accuracy [80]. Convolutional Neural Networks (CNNs) are strong in image, video or audio processing [63, 37]. They are able to classify the contents of an image or to mask objects in it on a pixel level. This masking process is called *semantic segmentation*. In remote sensing, the term "segmentation" traditionally refers to partitioning the image into groups of pixels that share a similar spectral signature. Subsequently, these groups are classified by a traditional machine learning tool like a support vector machine, based on their color or other features. In contrast to that, "segmentation" in computer vision refers to the aforementioned masking of entire physical objects. CNNs handle the delineation of objects and their classification in one step. They are for example used to segment MRI images into benign and malign tissue [16] and another research case is the processing of video data captured for self-driving cars [79]. Lastly, CNNs can of course be used to segment remote sensing imagery, which is the content of the work presented.

We will carry out three different computer experiments, focusing on specific tasks:

1. Tree cover mapping
2. Detecting and counting palms
3. Tree species classification

Land cover mapping, or more specifically, tree cover mapping is a standard task in remote sensing, which is often carried out with various data sources. We will use this as an entry point and introduce the work flow using neural networks. The methods used in this first task will then be developed further in the direction of locating and counting individual palm trees in satellite images, which is the main contribution of this thesis. The section which deals with detecting palms has been published [22] and contents of this publication are included in this thesis. These passages are consequently marked by quotation marks. Lastly, we will apply a CNN for classifying tree species groups in aerial images.

Each of these tasks comes with different sub-tasks and each of them will be further introduced in the respective section. We will now review the work related to the three topics listed above.

# 2. Related Work

## 2.1. Tree Cover Mapping

Several methods exist for the task of land cover mapping - and thereby tree cover detection. There are two main approaches: Pixel based methods and object based image analysis (OBIA). The spectral methods employ a classifier which works on a per-pixel basis. In contrast, for object based methods the pixels are grouped according to local spectral similarity and these *objects* are classified in a second step. Common classification methods include the k-Nearest Neighbor algorithm (KNN), Random Forests (RF) or Support Vector Machines (SVM). Decision trees and simple neural networks, such as the multilayer perceptron (MLP), are used as well.

A comparison of KNN, RF and SVM on Sentinel-2A imagery with 20 m resolution was carried out by Than Noi et al. [76]. They employed the pixel-wise classification approach and found that the SVM performed best in a variety of scenarios, achieving accuracies higher than 93%. Another application of the SVM can be found in [20], where the pixel-wise land cover classification of the Ethiopian highlands is carried out using Landsat images with a resolution of 30 m. Here, the SVM achieves a precision of 59% and a recall of 78% in the classification of tree cover (equations (3.12) and (3.13) explain precision and recall, which are identical to the user's and producer's accuracy). Chen et al. [10] produced a global land cover map, including tree cover, which is also based on Landsat data. They combine several of the methods mentioned above, achieving a precision of 84.1% and a recall of 92.4%.

Object based methods allow to include properties such as local spectral variance or however defined texture metrics into the classification. These approaches often exhibit higher performance than the pixel-wise classification algorithms. A review of OBIA-approaches can be found in [52].

However, pixel- and object-based methods have recently been outperformed by deep learning approaches [11, 9]. Deep learning had a high impact on remote sensing in general [85, 83] and, more specifically, on land cover classification [43, 70]. The advantage of deep learning approaches is, that they are able to leverage the information contained in the texture *and* in the spectrum, without relying on handcrafted features or classification rules. For example, Iglovikov et al. [30] train a so-called U-Net [65] in order to perform the classification of WorldView-3 imagery with a resolution of 30 cm. The U-Net indeed takes textural information into account and directly generates a probability map for different classes. As we are going to use the U-Net in this thesis, it will be further explained in Section 4.1.

## 2.2. Palm Detection

The following passage is taken from the publication made in the course of this thesis [22]:

"The following studies explicitly dealt with detecting individual palms: Srestasathiern and Rakwatin [73] used Quickbird and WorldView-2 images with 60 cm spatial resolution and four spectral bands (R, G, B, NIR) in Thailand. They derived palm positions from a selected vegetation index, by using a data transform and maximum extraction. Using this approach they reached an $F_1$-Score between 89.7% and 99.3%. However, it is important to remark that they applied their algorithm to plantations where individual palms were well separated without overlapping crowns, and where the plantation borders had previously been delineated.

Li et al. [48, 47] and Cheang et al. [40] both used similar approaches: they trained a convolutional neural network (CNN) classifier. The network receives a small image patch with (or without) a palm in its center as an input and calculates a probability for this patch containing a palm. The small input window is moved over the whole image and at each position, the corresponding probability is recorded. By that, a “palm probability map” was created. Using non-maximum suppression, the palm positions were determined. This approach yielded very good results. However, Cheang et al. claim to reach an accuracy of 94.5% on images with overlapping crowns, but without providing an overview of their test dataset, nor their definition of accuracy or precise network performance. Li et al. [48, 47] train their CNN on several thousand image patches of $17 \times 17$ pixels with a resolution of 60 cm, of which 5000 image patches contained a palm. They reached $F_1$-Scores between 96.1% and 98.8% in [48] and between 92.2% and 97.1% in [47]. In both publications Li et al. use manually selected image sections for training and validation.

In their study, Li et al. [48] compare their deep learning based method with earlier methods, based on local maximum filtering [62] or template matching [35]. They show that the CNN classifier outperforms the conventional methods by 3 to 7 percentage points in terms of the $F_1$-Score. These conventional methods, and those based on spectral profiles [73, 71], require prior knowledge about the plantation borders, which makes them inadequate for large area plantation detection problems. In non-plantation areas, which also appear in our datasets, spectral methods deliver many false positives; the outcome of such a comparison would clearly be in favor of deep learning based models, which do not require the delineation of plantation borders in advance. An example can be found in [73] (Figures 5 & 6). Another alternative to deep learning approaches would be tree crown detection based on height profiles [39]. This method, however, is not applicable to the data used in this [thesis], as no height information is available.” [22]

## 2.3. Tree Species Classification

The detection of single trees, including their position and species is a task that requires high resolution images and, if possible, height information captured by Lidar. Most approaches employ the object based classification methods described above, involving the grouping of pixels according to spectral similarity and subsequent classification. A review of the most common methods can be found in [21].

Li et al. [45] perform the classification of urban trees in WorldView-3 images with a resolution of 30 cm. They segment the imagery into super-pixels (polygons) with a region-growing algorithm. Then, several features are selected per polygon, such as mean, variance or spectral indices, followed by their classification using an SVM. The authors report an accuracy of 80%-92% for five different species, but they were not able to detect single tree positions.

Rezaee et al. [64] use imagery from the same satellite to classify trees in Canada. They perform the same image segmentation as Li et al. [45], but classify the polygons into four species using a VGG16 CNN. Their training dataset was acquired on-ground and the accuracy was reported to be 82% over all. However, the authors do not describe their dataset in detail, so that its extent and therefore the extent of their achievements remains unclear.

Nevalainen et al. [59] have acquired an extensive on-ground validated set of 4100 reference trees in a study area in Finland. They captured hyperspectral data of their study site with a resolution of 5 cm and 33 bands using a UAV. From this data, a three-dimensional digital surface model (DSM) was created by automatic image matching. The watershed transform was then used to derive individual segments for each tree crown. These were then classified into four species by a MLP or a Random Forest, reaching an over all accuracy of 95%. Altogether, this is a very convincing result, but it demands complex preprocessing to obtain the DSM and also lots of manual labor for the operation of the UAV.

# 3. Theoretical Background

## 3.1. Neural Networks

The theory section of this thesis is divided into two parts: In this section we discuss neural networks and in the next, remote sensing is addressed. LeCun, Bengio and Hinton, famous pioneers in deep learning, define that neural networks are "computational models that are composed of multiple processing layers [able] to learn representations of data with multiple levels of abstraction" [44]. There are four key terms in this quote: computational model, layer, learning and abstraction. Neural networks are computational models, which implies that they receive an input, process it and produce an output. The model's internal structure determines which tasks it can solve. There the second keyword comes into play: deep learning models are composed of *layers*, which are themselves built up by artificial *neurons*. These neurons are inspired by their biological counterpart and hence give the term "neural network" its name. Each processing layer applies a certain mathematical operation to its input. These operations depend on the connections between the neurons and their connection strengths; the *weights* of the connections. For each model input $x$, there is a desired network output (*ground truth*) $y_t$ and the actual (predicted) output $y_p$. As the actual output depends on all the weights inside the network, there is a set of weights, for which $y_p$ is closest to $y_t$. In order to find this optimum, a technique called *backpropagation* is used - this is the learning procedure corresponding to the second last key term. The input to such a network could be for example an image and the task to classify the displayed content. To refer to the last key term of the quote, the network in this example abstracts from the pixel values of the image to a class like "car" or "tree". In between, it creates "multiple levels of abstraction": In the beginning this could be for example a version of the input with enhanced edges or filtered colors. The output of later layers becomes harder and harder to interpret by humans, which is why neural networks are often criticized as nontransparent [53]. This is the key difference between deep learning and traditional machine learning; in the traditional case, the features a model uses to make a prediction are handcrafted by humans and therefore understandable. In the case of deep learning, they are a product of the backpropagation process and a result of numerical optimization. The term "deep learning" refers to the fact, that the neural networks employed can have many layers (100 or even more [28]). There are many different types of neural networks; in this thesis we will concentrate on so-called convolutional neural networks (CNNs), as they are best suited for the classification and segmentation of images. The following figure depicts the structure of a generic fully convolutional network used for image classification. We will later revisit each of its components and explain the training procedure in detail.

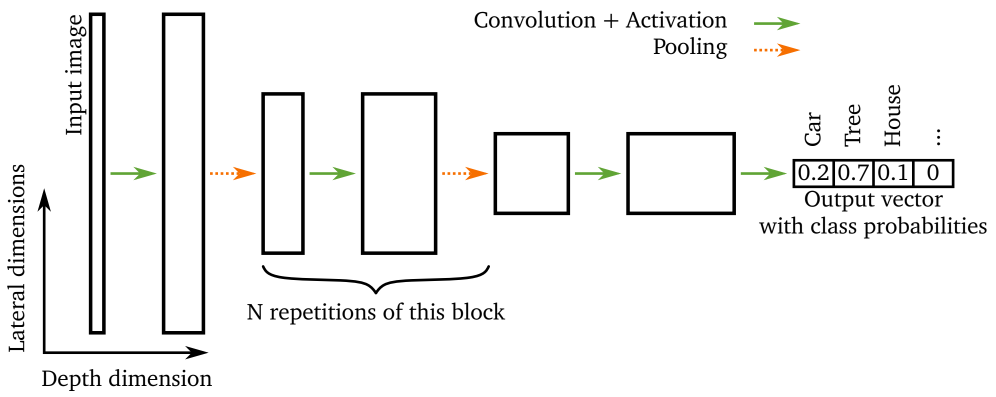


**Figure 3.1.:** Structure of a generic fully convolutional network for classification (FCNN). First, a convolution acts on the input image, followed by a non-linear activation and a so-called *pooling* operation, which halves the lateral dimensions (height and width). Convolution, activation and pooling follow in an alternating manner, until the output is small enough to be converted into a vector containing probabilities for each class by a final convolution.

The input to the network is an image, which usually consists of three channels (red, green, blue) in the depth dimension, but in remote sensing satellite images can have many more channels or *bands*. In Figure 3.1 the lateral dimensions (image height and width) are shown condensed into just one dimension. First, a convolution acts on the input image, which is then followed by an activation function and a pooling operation. This pattern is repeated, until a final convolution produces a vector containing class probabilities. The convolution learns the features necessary for classification, the activation function introduces a non-linearity into the network, which is essential for the learning process and the pooling reduces the lateral dimension and filters out the most relevant information. As this is a widespread sequence in the structure of neural networks, these operations will later be explained in this order.

The following flowchart (Fig. 3.2) depicts the general training scheme of neural networks. Basically, the training consists of calculating the error the network has made during the prediction, followed by updating the internal state of the network in a way that minimizes this however defined error. We will now explain the different components of a CNN in more detail, including the convolution and pooling operations introduced above, as well as the training procedure.

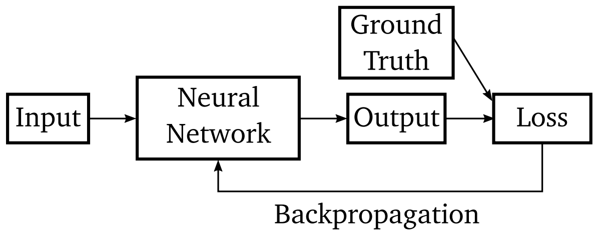


**Figure 3.2.:** The loss function calculates the error the network has made. Then the backpropagation adjusts the internal state of the network so that future predictions are closer to the ground truth.

### 3.1.1. Neurons

In the following we will draw parallels between biological neurons and their computational equivalents. Due to the high complexity of the natural processes, these explanations have to remain superficial. An introduction to the field of neuroscience can for example be found in [7].

The neuron is the smallest functional unit in the brain. Among other parts, it is composed of the *cell body*, the *dendrites* and the *axon*. The dendrites constitute the "receiving end" of the cell, the axon is the "transmitting end". Neurons mostly have several dendrites, but only one axon (though there are exceptions). The axon of one cell connects to the dendrites of another via *synapses*. These connections have a certain strength, which determines how well a signal is transmitted from one cell to the next. Simply speaking, incoming signals from the dendrites are aggregated in the cell body and if the integrated signal exceeds a certain threshold, it is propagated to the next cell through the axon.

Since decades, researchers are modeling this behavior in computer experiments, with the most notable result being the *perceptron*, invented by Rosenblatt in 1958 [66]. The perceptron is a network of artificial neurons. Figure 3.3 depicts a simplified biological neuron with its artificial counterpart, which constitutes the perceptron.

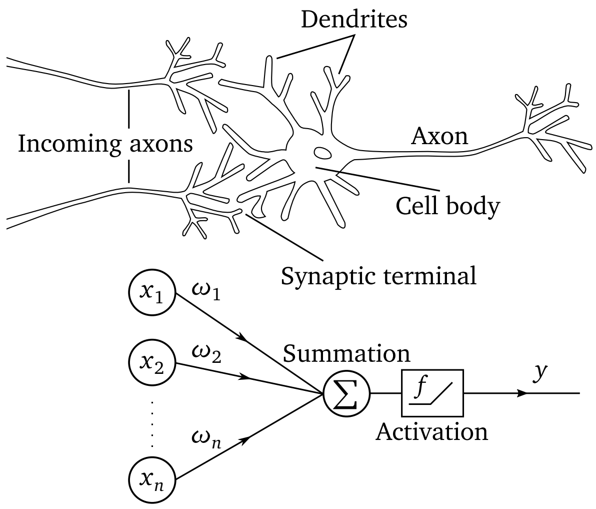


**Figure 3.3.:** Biological and artificial neuron. Information from the incoming axons is represented by $x_i$. The connection strength of the synapses is modeled by a weight parameter $\omega_i$. A weighted summation mimics the integration process in the cell body, followed by an activation function $f$, which determines whether the information is passed on or not. The final output $y$ is analogous to the axon and thereby the input $(x)$ to the next neuron.

The artificial neuron can be modeled by equation (3.1), which consists of a weighted summation, followed by a non-linear activation function (see Sec. 3.1.3) that mimics the threshold applied by the cell body:

$$y = f\left(\sum_i \omega_i x_i + \beta_i\right) \tag{3.1}$$

$x_i$: Input to the neuron, $\omega_i$: Weight parameter, $\beta_i$: Additive bias or offset, $f$: Activation function (see Sec. 3.1.3), $y$: Neuron output

The connection weights $\omega_i$ are *learned* in the backpropagation process, which we will discuss in Section 3.1.7. Several neurons can be concatenated to form *layers*. By connecting layers in series, simple logic operations can be performed. With increasing complexity, the network can also perform more complex tasks. Figure 3.4 gives an example of a *multilayer perceptron* (MLP) [66], which is comprised of *fully connected layers*. Each neuron of a fully connected layer is connected to all neurons in the layer before.

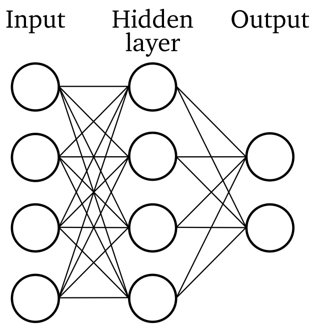


**Figure 3.4.:** The multilayer perceptron (MLP). Each circle represents a neuron, which performs summation and activation. Each line is a weighted connection to the neurons before. This example depicts fully connected layers. Layers inside the network are called *hidden layers*.

This is one of the simplest neural network architectures, but already incorporates the most important attributes (integration, activation, variable weights). We will later employ fully connected layers in networks used for classification.

## 3.1.2. Convolutional Layers

The mathematical operation of convolution is the central building block of CNNs, hence giving them their name. In the case of discrete raster data, such as images, the convolution receives two inputs: the image $I$ and a kernel $K$, with which the image is convolved. Both, image and kernel are in this case three dimensional arrays with $I$ being $H$ pixels high, $W$ pixels wide and having $C$ channels in depth. The kernel is a matrix with dimensions $H_k \times W_k \times C$, so image and kernel share the number of channels. In lateral direction (height and width), the kernel is much smaller than the image, being usually $3 \times 3$ or $5 \times 5$ pixels in size. The output of the convolution is a new array with dimensions $(H - H_k + 1) \times (W - W_k + 1)$, so the depth dimension is "consumed" by the convolution and the lateral dimensions are reduced as well. The convolution operation can be seen as sliding the kernel over the input image, computing the element-wise product at each position $i, j$. Following equation gives the mathematical definition of the convolution, as used in deep learning:

$$(I * K)_{i,j} = \sum_{l=0}^{H_k-1} \sum_{m=0}^{W_k-1} \sum_{n=1}^{C} I_{i+l,j+m,n} \, K_{l,m,n} \tag{3.2}$$

$I$: Input image, $K$: Convolution kernel, $i, j$: Indices of convolution output array
$l, m, n$: Spatial and channel indices

The convolution operation can be re-written in a form that matches the neuron-equation (3.1). Consequently, each element of the kernel can be seen as *weight* and is *learned* by the network in the backpropagation process, which we will discuss in Section 3.1.7.

Figure 3.5 demonstrates how the convolution is computed for a grayscale image. In order to achieve an output with the same lateral extent as the input, the input is padded, which means that the border values are mirrored with an appropriate width (here one).

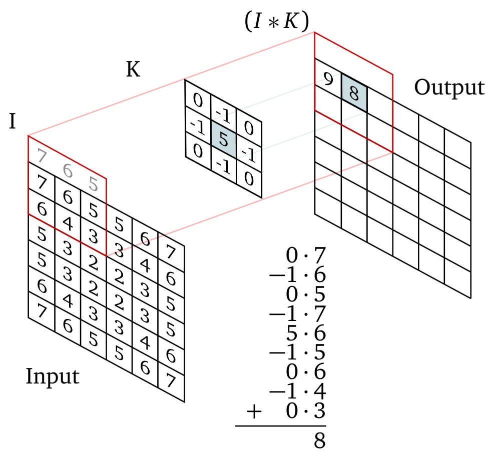


**Figure 3.5.:** The graphic [61] shows how the second element of the first row is calculated. After this step, the computation window slides one step to the right and after finishing one row, it advances to the next.

Depending on the kernel, the convolution can for example enhance edges or blur the image, as Figure 3.6 demonstrates:

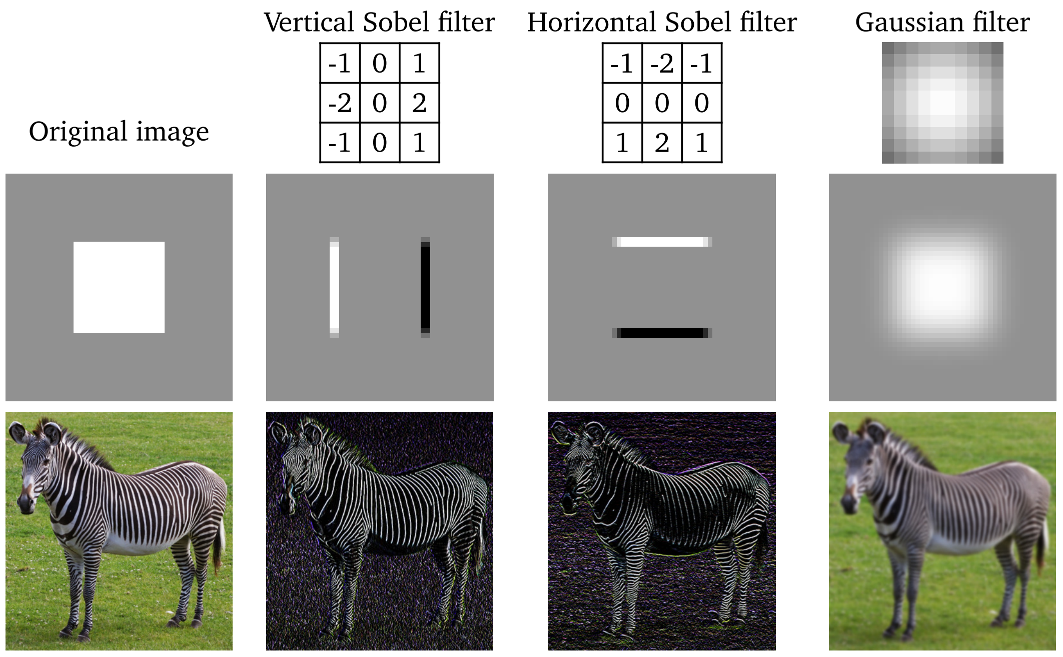


**Figure 3.6.:** The top row shows the used kernel, the second and third rows the results. Gray corresponds to a value of zero, black to negative and white to positive values. In the Zebra image [78], all color channels are processed independently. Here, one can nicely see how the filters highlight vertical or horizontal stripes differently.

### Convolutional Layer

In one convolutional layer, there are $N$ convolutions acting in parallel on the same input, each one with a different kernel. As an intermediate result, there are $N$ different two-dimensional images, the so-called *feature maps*. These are stacked in order to form the output of the convolutional layer, which now has the dimension $(H - H_k + 1, W - W_k + 1, N)$. Each convolutional layer is then usually followed by an *activation* function, which introduces a non-linearity, and *pooling*, which reduces the lateral extent of the feature maps. These operations will be described in Sections 3.1.3 and 3.1.4.

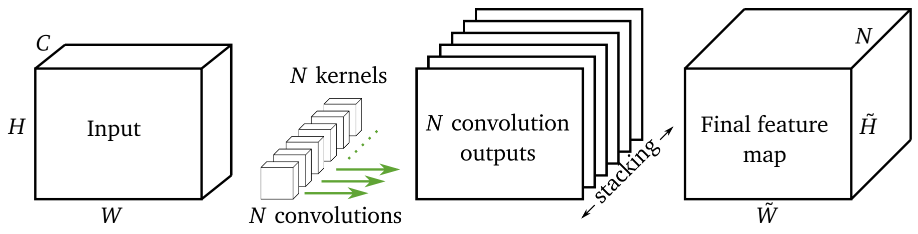


**Figure 3.7.:** In a convolutional layer, $N$ convolutions act on the input with different kernels, resulting in $N$ two-dimensional outputs. These are stacked in order to generate the final feature map, which is the output of the layer.

### Padding, Stride and Dilation

Padding, stride and dilation are means of controlling the output size of a convolution operation and how it extracts information from its input. When *padding* an image or feature map, additional values are inserted around its border. Padding can be done with a constant value (usually zero) or a reflection of the image border, as we already saw in Figure 3.5. The larger the pad width $p$, the larger the output after the convolution. Padding can therefore be used to retain a feature map's lateral size during the convolution operation.

The kernel can be slided over the input with a certain step width, the *stride*. A larger stride $s$ results in a smaller output image; a stride of two e.g. halves the output size. Above we assumed a stride of one, so no aggressive reduction of information took place. Figure 3.8 illustrates the combination of *padding* and *stride*:

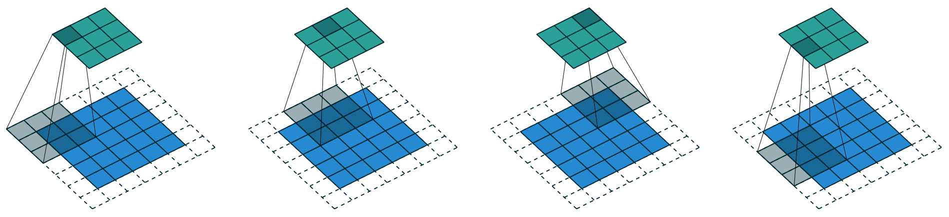


**Figure 3.8.:** In this example [19], the input (blue) is padded with $p = 1$. The kernel size is 3 and the stride is $s = 2$. As a consequence, the output (cyan) has half the size of the padded input.

The term *dilation* is used to describe the insertion of zeros into either the kernel or the convolution input. Dilating the input, while using a non-dilated kernel, increases the output size compared to a non-dilated input. If a dilated kernel is used, its size is effectively larger than its non-dilated equivalent, resulting in a smaller convolution output. Dilating the kernel allows to incorporate long-range correlations into the feature map, as each resulting pixel receives information from a larger area, which can be seen in Figure 3.9:

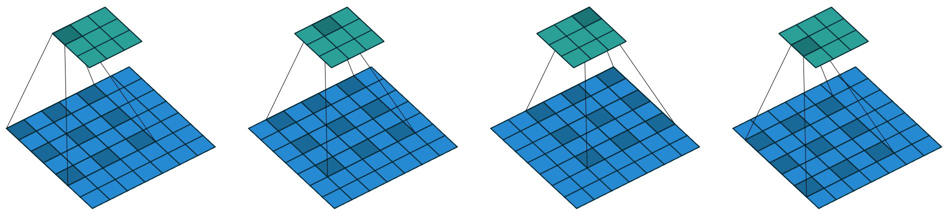


**Figure 3.9.:** Dilated convolution with zeros inserted into the kernel. Only the grayed values contribute to the output. As a result of the larger kernel (here 5 x 5 instead of 3 x 3), the output becomes smaller. Image taken from [19].

**Transposed Convolutions**
As we have seen so far, a convolutional layer can be used to downsample its input. The opposite direction is also possible: Padding, stride and dilation can be combined in a way that allows to upsample the input via convolution, using a learnable interpolation method. The resulting operation is called *transposed convolution* (not to confuse with deconvolution).

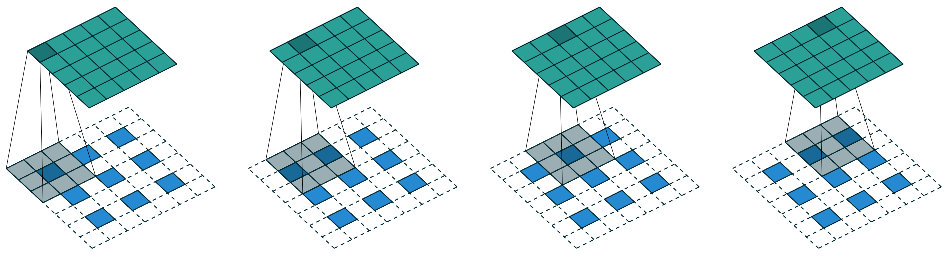


**Figure 3.10.:** Transposed convolution [19]: The input of size 3x3 is dilated with zeros and padded to size 7x7. Then a regular convolution is performed, resulting in an enlarged output of size 5x5. Image taken from [19].

A rigorous treatment of convolution arithmetics, including padding, stride, dilation and transposed convolutions, can be found in [19].

### 3.1.3. Activation Functions

There are many different activation functions available for neural networks [27, p. 67-70, p. 81]. Their most important feature is that they are non-linear functions. In contrast, the convolution is a linear operation. This has the effect that, if one applies many convolutions after each other to form an output, there exists a single convolution that forms the very same output. This means that a neural network without non-linearities can essentially perform only linear regression tasks. Therefore, non-linearities play a key role in neural networks. As we will see later, the derivatives of the activation functions are important as well, since they determine how the different weights in the convolutional layers get updated in the backpropagation process. The following is not a comprehensive list, but rather gives an overview of the most important activation functions. More can easily be found on the internet or in books [27].

**ReLU activation**
The rectifying linear unit (ReLU) activation was used to model the behavior of biological neurons as early as 1975 [23]. Biological neurons start to fire, when their cumulated input exceeds a certain threshold and mostly sit idle the remaining time [7]. The ReLU function mimics this behavior by mapping all negative inputs to zeros and by letting all positive input pass through, so the threshold value is zero. It has proved to allow for efficient training of neural networks [41]. The derivative of the ReLU function is the Heaviside- or step-function.

$$\mathrm{ReLU}(x) = \max(0, x) \tag{3.3}$$

**Sigmoid activation**
The sigmoid activation or logistic function is a widely used activation function and has a characteristic “S” shape. It has continuous derivatives and is bounded between 0 and 1. This is why it is used to map arbitrary values to this interval. The sigmoid function can for example be used as last activation in binary classification tasks, but not for categorical classification and has the advantage that its first derivative is simple to compute.

$$S(x) = \frac{1}{1+e^{-x}} \tag{3.4}$$

$$S'(x) = S(x)(1-S(x)) \tag{3.5}$$

**Softmax activation**
The softmax activation is a generalization of the sigmoid function that can handle multi-dimensional input. Let $\boldsymbol{z}$ be a $C$-dimensional input vector to the function. It then returns $C$ values between zero and one, that add up to one. The function attenuates the highest value of the input vector, which turns out closest to one. This makes the softmax function ideal for categorical classification tasks, where the network has to “decide” which output value is most relevant.

$$\sigma(\boldsymbol{z})_j = \frac{e^{z_j}}{\sum_{k=1}^{C} e^{z_k}} \quad \text{for } j = 1, \dots, C \tag{3.6}$$

### 3.1.4. Pooling

Pooling is a technique for downsampling, which reduces the amount of information passed through the network. There are different types of pooling, with *max pooling* being the most common one. Max pooling takes an array as input and replaces every block of $n \times n$ values by their maximum. Other pooling variants for example take the average of each block. Pooling acts on each depth-layer independently, so only the lateral dimensions are changed. Max pooling with a size of $2 \times 2$

for example discards 75% of the incoming information, which reduces the number of parameters in later layers. This reduces the risk of overfitting and speeds up the computation. An alternative to pooling are strided convolutions, as described in Section 3.1.2.

$$\begin{array}{|cc|cc|}\hline 1 & 2 & 3 & \mathbf{9} \\ 3 & \mathbf{4} & 7 & 2 \\ \hline 6 & 0 & 1 & 2 \\ \mathbf{8} & 3 & \mathbf{6} & 5 \\ \hline \end{array} \quad \Rightarrow \quad \begin{array}{|cc|}\hline 4 & 9 \\ 8 & 6 \\ \hline \end{array}$$

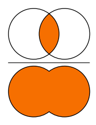
**Figure 3.11.:** Example for 2 × 2 max pooling. Each 2 × 2 block is replaced by its maximum value.

## 3.1.5. Metrics

In order to measure the performance of a network, we will define different *metrics*. Depending on the task, a suitable metric has to be chosen. When we use a classifier to differentiate between $C$ classes, the network returns a normalized vector $\boldsymbol{y}_p \in \mathbb{R}^C$. Each entry of the vector corresponds to the probability of the respective class and will be denoted with $y_{p,c}$. When a network for image segmentation is used, it returns an entire probability map or a matrix $\boldsymbol{Y}_p \in \mathbb{R}^{H\times W\times C}$, respectively. This matrix is $H$ rows high, $W$ columns wide and has $C$ channels in depth. For the calculation of the metrics, this matrix is reshaped by concatenating its rows ($\mathbb{R}^{H\times W\times C} \rightarrow \mathbb{R}^{H\cdot W\times C}$). Essentially, it now holds $C$ vectors with $H \cdot W$ entries, which will be denoted with $\boldsymbol{y}_{p,c}$ (bold means vector). These vectors will then be compared to their ground truth counterparts $\boldsymbol{y}_t$.

### Intersection over Union

The intersection over union (IoU) or Jaccard index ($J$) is used to measure the quality of a segmentation. It ranges from zero to one. The closer it is to one, the better the predicted mask and the ground truth overlap. For two sets A and B it can be expressed in the following manner:



$$\mathrm{IoU}(A,B) = \frac{|A \cap B|}{|A \cup B|} = \frac{\phantom{x}}{\phantom{x}} = \frac{|A \cap B|}{|A| + |B| - |A \cap B|} \tag{3.7}$$

Here, $A$ could be the ground truth mask and $B$ the network's prediction. The intersection over union $J_c$ for class $c$ can then be defined as follows:

$$J_c(\boldsymbol{y}_{t,c}, \boldsymbol{y}_{p,c}) = \frac{\boldsymbol{y}_{t,c} \cdot \boldsymbol{y}_{p,c}}{|\boldsymbol{y}_{t,c}| + |\boldsymbol{y}_{p,c}| - \boldsymbol{y}_{t,c} \cdot \boldsymbol{y}_{p,c}} \tag{3.8}$$

$\boldsymbol{y}$: Vector containing the ground truth or the prediction for class c.
$|\ldots|$: Sum of all vector elements.

By averaging over all classes one obtains the mean IoU $\bar{J}$. The different classes can be weighted using a weight parameter $\mu_i$, which we will set to 1, if not mentioned otherwise.

$$\bar{J}(\boldsymbol{y}_t, \boldsymbol{y}_p) = \frac{1}{C} \sum_{i=1}^{C} \mu_i \, J_i(\boldsymbol{y}_{t,i}, \boldsymbol{y}_{p,i}) \tag{3.9}$$

In cases where we have to set $\mu_c$, we will choose it according to the following equation:

$$\mu_i = \left( n_i \sum_{k}^{C} \frac{1}{n_k} \right)^{-1} \tag{3.10}$$

$n_i$: Number of instances (or pixels) for class $i$.

**Accuracy, Precision and Recall**
The accuracy, or more specific the Rand Index, measures the agreement of prediction and ground truth. One can apply it to predicted masks or to sets of objects. It is zero, if the prediction misses every ground truth element and one, if both match perfectly.

$$\text{Accuracy} = \frac{t_p + t_n}{t_p + t_n + f_p + f_n} \tag{3.11}$$

$t_p$: True positives, $t_n$: True negatives, $f_p$: False positives, $f_n$: False negatives

Several other metrics can be evaluated in order to characterize a prediction: Considering the example of identifying tree instances in an image, the *precision* tells how many non-trees were falsely marked as trees. High precision means, that almost every identified object is a tree. In the context of remote sensing, the precision metric is also called *user's accuracy*.

$$\text{Precision} = \frac{t_p}{t_p + f_p} \tag{3.12}$$

In contrast, the *recall* measures how many instances were missed during prediction. A high recall means, that every instance was found, regardless of how many objects were falsely marked positive. The recall is also known as *producer's accuracy*.

$$\text{Recall} = \frac{t_p}{t_p + f_n} \tag{3.13}$$

It is often the case that changing certain parameters (such as detection thresholds) during the prediction phase influences precision and recall. Higher precision comes with lower recall and vice versa. Depending on the task at hand, one then has to decide which metric is more important and to adjust the parameters accordingly. The $F_1$-Score is a “tradeoff” between the two and defined as their harmonic mean:

$$F_1 = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{3.14}$$

### 3.1.6. Loss Functions

The loss function is a measure for the error of a prediction with respect to the ground truth. The choice of the loss function is crucial for the success of the network training and depends on the task to solve. However, there is not only one possible loss per task and the decision which to use has to be made on empirical grounds and based on the task's properties. If the neural network is for example used to approximate a function, which is essentially a regression, the mean squared error (MSE) is an appropriate choice. When it comes to the task of classification, other loss functions are suited better because the dependent variables in classification tasks are discrete (0 or 1) and not continuous as in regression. Classifications are normally represented as vector; when there are $C$ classes, the vector is $C$-dimensional and each entry corresponds to a class. If a ground truth vector should represent class $i$, the $i$-th entry is set to one and all others are zero. In contrast, the network output is the output of e.g. the softmax function (eq. (3.6)), with real values that are normalized to one, as vector entries - this is best understood as a probability distribution. The difference between the true and predicted probability distribution is commonly measured using the *categorical cross-entropy* [27, p. 73ff].

$$H(\boldsymbol{y}_t, \boldsymbol{y}_p) = -\sum_{i=1}^{C} \mu_i \; y_{t,i} \log(y_{p,i}) \tag{3.15}$$

$\boldsymbol{y}$: Vector containing the ground truth or the prediction.
$\mu_i$: Weight parameter for class i.

In a classification task with mutually exclusive classes, only one entry of the true probability distribution can be one. In this case, the categorical cross-entropy (CCE) ignores the predictions for all other classes and the prediction for the true class is logarithmically weighted. Therefore, the resulting loss can be between infinity for a completely wrong, and zero for a perfect prediction. In segmentation tasks the CCE is calculated per pixel of the output probability map and then averaged over all pixels.

Of course there are many other loss functions and it is possible to combine them or to construct own loss functions. It is advisable to incorporate the quantities that should be optimized into the loss function in order to get good results. For segmentation tasks a high IoU is desired and at the same time, the task can be seen as pixel classification problem. Consequently, it is useful to combine the normal CCE with an IoU-derived loss [30]:

$$L = H - \ln(J) \tag{3.16}$$

Here we used the negative logarithm, because we want to penalize a low IoU with a high loss. In a setup with only one class, we use $J = J_c$ and for more classes $J = \overline{J}$ (mean). It can easily be explained, why the CCE alone is no sufficient loss function. Imagine a large image with a comparatively small labeled object on it. If the network now predicts 100% background, the CCE loss will be very small. As a consequence, the small object will be neglected during training. With the combined loss, the IoU for the object will be very low. Taking the negative logarithm, the loss becomes high. Therefore, the network will be punished stronger for not segmenting the small object.

### 3.1.7. Backpropagation

Backpropagation is the algorithm, that enables neural networks to "learn". It is an optimization procedure for the internal network state; the weights and entries of the convolutional kernels. Figure 3.12 gives an overview of the general training procedure. In this section, we deal with the highlighted step, which is calculating the derivatives of the loss function.

Consider the following sequence: we have a network which has produced an output from a given input. We can then calculate the error the network made, the loss, according to Section 3.1.6. The loss depends on the the desired network output, the actual output and by that also on all the weights inside the network. We now want to update the set of weights $\omega$ by some $\delta$ in a way that the loss is minimized.

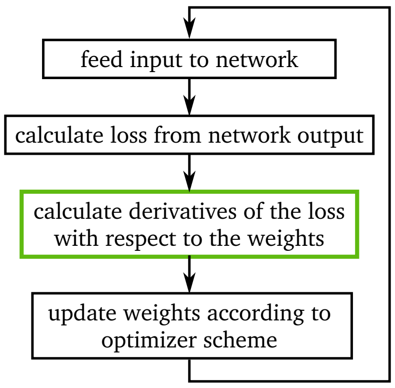

feed input to network
calculate loss from network output
calculate derivatives of the loss with respect to the weights
update weights according to optimizer scheme

**Figure 3.12.:** General neural network training procedure.

$$\omega_{new} = \omega_{old} + \delta \tag{3.17}$$
$$L(y_t, y_p; \omega_{new}) < L(y_t, y_p; \omega_{old}) \tag{3.18}$$

The question is: How do we calculate the $\delta$ in a way that minimizes the loss? As the loss and all internal functions of the network are differentiable, we can calculate the derivative of the loss function with respect to every weight. This *gradient* represents the amount and direction of influence each weight has on the loss. The gradient points uphill in the direction of steepest slope in the "loss-landscape". Knowing this, we can write down an equation that optimizes the weights:

$$\omega_{new} = \omega_{old} - \alpha \nabla_{\omega_{old}} L \tag{3.19}$$

Where $\nabla_\omega$ is the nabla-operator, which takes the derivative of $L$ with respect to every weight and $\alpha$ is the so-called learning rate, which is essentially a step-width that defines how far each update step goes against the direction of the gradient. The minus sign is chosen, because we want to *descend* in the loss-landscape. Equation (3.19) is the most simple form of an update rule: The gradient descent.

The next question is how the gradient is calculated. The answer is trivial to give analytically: By using the chain rule of differential calculus. But computationally, it can be challenging to find the gradients in an efficient manner and much of the processing time is spent in this step.

The chain rule defines how composite functions are derived: Let $z$ be a variable, that depends on $y$, where $y$ itself depends on $x$. The derivative of $z$ after $x$ can than be written as the product of two simpler derivatives.

$$\frac{dz}{dx} = \frac{dz}{dy}\frac{dy}{dx} \tag{3.20}$$

Neural networks can be seen as computational graphs, as each node (operation) is connected to others. Figure 3.13 depicts a very simple graph: The repeated application of the same function $f$ to some input $w$. The intermediate results are then given by $x = f(w)$, $y = f(x)$ and finally $z = f(y)$. The function $f$ here symbolically stands for some function that appears in the network, such as the convolution or ReLU. The output $z$ could be the loss function and $w$, which is now the input, could as well be some weight parameter deep inside the network. Using the chain rule, we can then calculate the derivative of $z$ after $w$:

$$\frac{dz}{dw} = \frac{dz}{dy}\frac{dy}{dx}\frac{dx}{dw} = f'(y)f'(x)f'(w) \tag{3.21}$$

We immediately see that we have to calculate the intermediate results x and y before we can calculate the derivatives. This is called the *forward pass*, which is followed by the *backward pass*, where we actually calculate the derivatives. The backward pass is only needed during the training of the network, which is why the pure application of the network is faster.

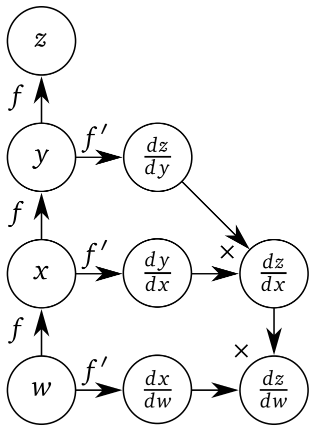


**Figure 3.13.:** Scheme of the backpropagation process. Once the forward pass is complete, one can start to calculate the derivatives, going from top to bottom. The “×” indicates multiplication. The desired final derivative is shown in the lower right. Recreated from [27, p. 214].

The last question is *when* to apply the gradient update, which we will answer in the next section. As we now know how to calculate the gradient of the loss function, we can move on and take a closer look at how it is utilized in different optimizers.

### 3.1.8. Optimizers

**Stochastic Gradient Descent**

The Stochastic Gradient Descent (SGD) algorithm [36] is an expansion of the simple gradient descent from equation (3.19). In theory it is possible to apply one gradient update step after each single network input. But this raises the problem that the gradients fluctuate a lot, resulting in poor convergence. Another option would be to calculate the loss for each item in the training set and to perform the update step with an averaged gradient. In this case, the convergence would be monotonous, but very slow, as the network has to iterate through the entire dataset for each update.

The best compromise is to divide the dataset into *batches*. One batch usually contains between 16 and 256 samples, depending on the task and the used hardware. During the training process, the network receives a batch of samples at a time and calculates an output for each of the samples in the batch. From the output, the loss function and the gradients are calculated. Thereafter, a regular update step takes place with the *batch-averaged* gradients. This procedure is called *Stochastic Gradient Descent* and the term *stochastic* comes from the fact, that the dataset is randomly sampled to obtain the batches. Actually, the use of batches is advisable for most deep learning problems and will also be used with other optimizers.

**Adam Optimizer**

The SGD algorithm has the advantage of being simple, but also has several drawbacks. One major drawback is, that it is slow in some cases (e.g. at saddle points of the "loss landscape") and gets stuck in sub-optimal minima. The Adam Optimizer [38] was developed to address these and other issues of SGD. It converges faster than most other optimizers in a wide variety of scenarios, while staying relatively simple in implementation and execution. The optimizer uses exponential moving averages of the first and second moment (mean and uncentered variance) of the gradients in order to derive update steps specific for each single weight. The name is derived from adaptive moment estimation. For the sake of completeness, the algorithm listing from [38] is presented below.

**Algorithm 1** Adam optimization [38]. In the process, the weights $\omega$ in the network are changed in a way that minimises the loss $L$. Each weight receives its own update, depending on the mean $m$ and uncentered variance $v$ of the gradient of $L$ w.r.t. the weight. The decay rates are chosen as $\beta_1 = 0.9$ and $\beta_2 = 0.999$.

```
Require: $\alpha$ (Step size or learning rate)
Require: $\beta_1, \beta_2 \in [0,1)$ (Exponential decay rates for moment estimates)
Require: $L(\omega)$ (Loss function to be minimised, depending on the weights $\omega$)
Require: $\omega_0$ (Initial weights of the network)
1: Initialisation:
2: $t \leftarrow 0$
3: $m_0 \leftarrow 0$ (Initialise first moment vector with zeros)
4: $v_0 \leftarrow 0$ (Initialise second moment vector with zeros)
5: while network is training do
6:     $t \leftarrow t + 1$
7:     $g_t \leftarrow \nabla_\omega L_t(\omega_{t-1})$ (Calculate the gradients of $L$ w.r.t. the weights $\omega$ at time step t)
8:     $m_t \leftarrow \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t$ (Update biased first moment estimate)
9:     $v_t \leftarrow \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot g_t^2$ (Update biased second moment estimate)
10:    $\hat{m}_t \leftarrow m_t / (1 - \beta_1^t)$ (Perform bias correction for the first moment estimate)
11:    $\hat{v}_t \leftarrow v_t / (1 - \beta_2^t)$ (Perform bias correction for the second moment estimate)
12:    $\omega_t \leftarrow \omega_{t-1} - \alpha \cdot \hat{m}_t / (\sqrt{\hat{v}_t} + \epsilon)$ (Update weights)
13: end while
```

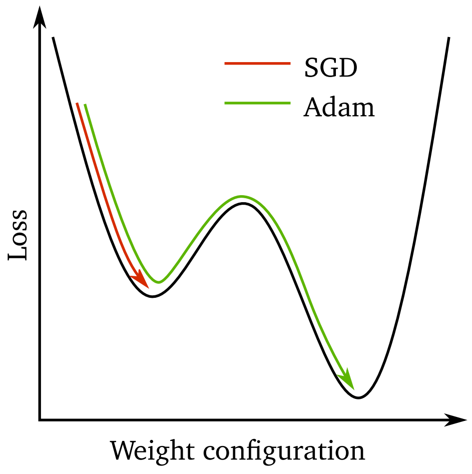


**Figure 3.14.:** The SGD optimizer continues until it reaches the first minimum and possibly oscillates around it. In contrast to that, the Adam optimizer has a momentum that may carry it over hurdles, allowing it to reach minima further away.

In the beginning, the mean $m$ (first moment) and the uncentered variance $v$ (second moment) of the gradients as well as the time are initialized as zero. In each training step the network generates an output from the input it received and the time step is incremented by one. With the output, the gradient of the loss function is calculated. In line eight and nine, the gradients and their element-wise squares are multiplied with a factor and accumulated in the estimates for the mean and uncentered variance. In addition to that, the mean and variance from earlier time steps are multiplied with a factor $\beta_i < 1$, resulting in an exponential decay of older gradient values. Thereby, the mean and variance represent moving averages that value the newest value highest; older values decay exponentially. The choice of initial conditions as zero introduces a bias to the first and second moment. In lines ten and eleven this effect is mitigated by dividing by an appropriate value. This can be understood by calculating the results for the first time step. In line eight, $m_0$ is zero for the first time step and $g_1$ is multiplied by 0.1 (as $\beta_1 = 0.9$). Consequently, $m_1$ now equals $0.1 \cdot g_1$. In line ten, $m_1$ is divided by $(1 - \beta_1^{t=1}) = 0.1$ and the bias-corrected mean $\hat{m}_1$ now correctly holds $g_1$. In line 12 the final update takes place and each weight is corrected with an effective step width $\Delta\omega = \alpha \cdot \hat{m}_t / (\sqrt{\hat{v}_t} + \epsilon)$. This effective step width is composed of the learning rate and the quotient of the mean and the square root of the variance. One can see, that weights with a high variance compared to the mean receive a smaller update step. This is why the Adam optimizer still performs well under noisy conditions, which makes it favorable for deep learning, where a wide variety of inputs is present. See [38] for a more detailed explanation of the algorithm and a derivation of the bias correction term.

In this thesis a variant of the Adam optimizer is used that incorporates Nesterov's momentum [58, 17]. See [68] for an overview of existing optimization methods and a visualization of the different convergence behaviors.

### 3.1.9. Batch-Normalization

The batch normalization (BN) layer [32], as the name already suggests, serves the purpose of normalizing the input to activation functions. Normalizing their input is beneficial because it keeps the activation functions in a regime where they are not constant (see Eq. (3.5)). In this region, the derivative of the activation functions with respect to their inputs is non-zero. If these derivatives *were* zero, the weights in layers before the activation layer would not get updated anymore. Hence, the network learns faster when batch normalization is used. Given a batch of $m$ training samples $x_i$, the BN-layer computes its average $\mu$ and standard deviation $\sigma$, then subtracts $\mu$ from the input and divides by $\sigma$. In some cases such a transformation might not be optimal, so the result is rescaled and shifted by two learnable parameters $\gamma$ and $\beta$. In this manner, the BN-layer can learn the identity transform and "act" as if it was not present. Equation (3.22) defines the batch normalization layer:

$$\begin{aligned} \mu &= \frac{1}{m}\sum_{i=1}^{m} x_i \\ \sigma^2 &= \frac{1}{m}\sum_{i=1}^{m}(x_i-\mu)^2 \\ \hat{x}_i &= \frac{x_i-\mu}{\sqrt{\sigma^2+\epsilon}} \\ \mathrm{BN}(x_i) := y_i &= \gamma\hat{x}_i+\beta \end{aligned} \tag{3.22}$$

In convolutional neural networks the batch normalization is applied to the output of convolutional layers, the feature maps, which have several channels in depth. The batch normalization acts on each channel independently.

Batch-normalization allows to remove *dropout* [74] from the network, as it provides a similar effect with respect to preventing a network from overfitting [32]. Therefore, we will not use dropout. Dropout is the process of omitting a random subset of a layer's weights during back-propagation.

### 3.1.10. Augmentation

Augmentation is the process of artificially increasing a given dataset by applying a set of transformations. Common operations for images are rotation, random cropping, flipping and color shifts. These operations can be applied during the training in order to reduce overfitting. Furthermore it is possible to do test-time augmentation during the inference process, which means presenting the network the same image with different transformations applied. The network outputs for the differently augmented images are then averaged to improve the segmentation or classification result.

## 3.2. Remote Sensing

In this thesis we will work with images coming from two different sources: satellites and airplanes. Both differ in several aspects, such as lateral and spectral resolution, view angle and more. In this section we will describe the data sources used and elaborate on the challenges each one brings.

### 3.2.1. Satellite Imagery

There is a wide variety of remote sensing satellites with different fields of application in orbit today. We will focus on optical sensors only, as they are most interesting for deep learning image processing tasks and most widely available. Some of these satellites, such as those from the *Landsat* program, are in orbit for several decades already, which gives the opportunity to study the temporal evolution of land cover. Other important satellites are for example *Pléiades 1A/B*, and the *WorldView* satellites. Most optical remote sensing satellites orbit the earth in lower earth orbit with mean altitudes between 450 km and 800 km in a poloidal, sun-synchronous orbit.

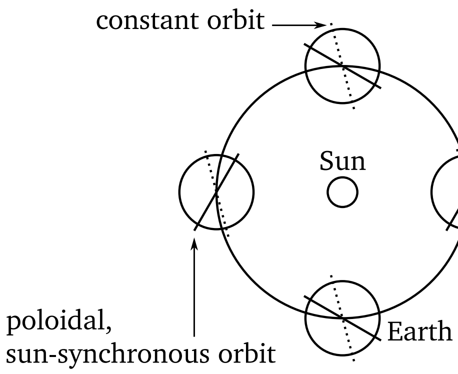


**Figure 3.15.:** The rotation plane of the sun-synchronous orbit (solid) rotates around the earth once per year. This rotation is driven by an angular momentum coming from earth's non-spherical shape. In contrast to that, the rotation plane of an unperturbed orbit (dashed) retains its orientation.

The sun-synchronous orbit has two main advantages: First, it is possible to place the satellite in constant sunlight and second, a certain point on the earth is always photographed under the same lighting conditions, apart from seasonal changes. This increases the comparability of acquired images. As the sun-synchronous orbital plane slowly rotates around the earth and the earth itself rotates, the satellites are able to revisit each area of interest within a couple of days or even once per day.

Many satellites scan the earth using line sensors ("push broom sensors"), as one imaging dimension is already covered by the satellite's motion, but two-dimensional sensors are employed as well [15, 1]. Depending on altitude, focal length and sensor, the satellite captures a region of certain width (the swath width) at ground level. The swath width ranges for example from 8 km (SkySat) to 185 km (Landsat 8) [1, 2], resulting in a theoretical coverage of up to one million km$^2$ per day.

Each satellite captures images at a certain set of wavelengths. Figure 3.16 gives an overview of those imaging *bands* provided by the WorldView-3 satellite, which we will use later on.

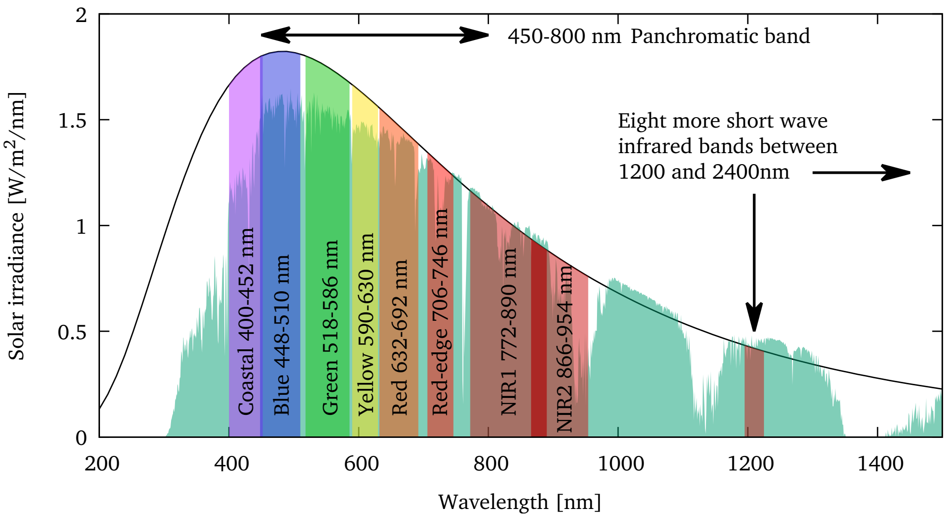


**Figure 3.16.:** WorldView-3 imaging bands and their wavelengths [3, 4], along with the solar irradiance spectrum and the spectrum of a black body at 5500 K.

The most important bands for land cover classification are red, green, blue and the near infrared bands. Longer wavelengths are mostly used for fire-detection, military applications or geological surveys [3]. Especially the difference between the red and the red-edge or NIR band provides information about the amount of vegetation covering the ground. In section 3.2.3 we will discuss this further.

Many satellites carry multiple imaging sensors for different sections of the spectrum. These sensors often differ in lateral and spectral resolution. For example, the resolution of the multi-spectral bands of WorldView-3 is 1.24 m and 0.3 m for the panchromatic band [3]. In order to obtain a colored image with high resolution, the panchromatic and multispectral bands are combined in a process called *pansharpening*. This process usually involves upsampling, alignment and intensity-shifting of the multispectral bands. An overview of the different techniques can be found in [50, 60].

## 3.2.2. Aerial Imagery

Aerial imagery comes with advantages and drawbacks compared to satellite images. The most prominent advantage is the higher resolution, which can easily be adapted by flight altitude. In contrast to satellites, aerial photography uses regular CMOS-chip cameras and not line scanners. Due to economical constraints the resolution is mostly limited to 5-10 cm per pixel. The low flight altitude necessary for this image quality comes with the drawback, that the imaging system needs a low focal length and hence a wide opening angle. Consequently, objects at the image border are captured under a different angle than center objects. During a flight, the plane captures a set of overlapping images along with its position and rotation angle. These tiles have to be mosaicked, in order to form a single, contiguous image. During this tiling process, an algorithm tries to detect key features along the image borders, which can be used to match neighboring tiles [86, 49]. The combination of different view angles on the same object and image tiling can lead to problems, which we will discuss in Section 6.3.4.

### 3.2.3. Spectral Indices

Having more than the regular red, green and blue imaging bands allows to make use of the spectral characteristics of certain materials. A very common substance in nature is chlorophyll, which comes in different forms. Figure 3.17 depicts the absorption spectra of two chlorophyll variants, which are present in bacteria, algae and plants.

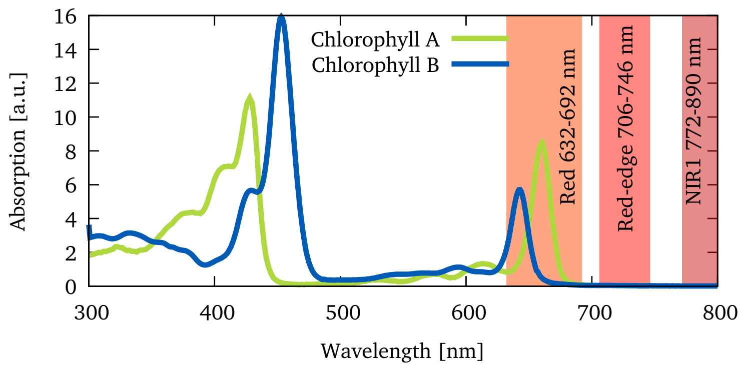


**Figure 3.17.:** Absorption spectra of Chlorophyll A and B in diethyl-ether[33], together with the relevant WorldView-3 imaging bands [3].

It shows, that the chlorophyll variants have absorption maxima in the blue (430 nm, 454 nm) and red (643 nm, 662 nm) region of the spectrum. For longer wavelengths the absorption is low compared to these maxima. As most blue and red light is absorbed, chlorophyll-rich materials look green to the human eye. Furthermore, most of the infrared light is reflected. One can make use of these spectral properties and derive certain *indices* with the aim of differentiating between different materials. When it comes to detecting plants, the most important index is the normalized difference vegetation index (NDVI), which is defined as follows [67]:

$$\mathrm{NDVI}(\mathrm{red}, \mathrm{nir}) := \frac{\mathrm{nir} - \mathrm{red}}{\mathrm{nir} + \mathrm{red}} \tag{3.23}$$

red: value of the red band (~650 nm)
nir: value of a near infrared band (~720-900 nm)

The NDVI can take values between -1 and 1, being high for areas with green vegetation and low for impervious surfaces or bare soil. Negative values normally distinguish water bodies, but especially if they are contaminated by algae, this is no clear indicator. Therefore, another index can be used to clearly identify water: the NDWI (normalized difference water index). It is based on the fact, that water absorbs longer wavelengths very well and obtained by replacing the red band in equation (3.23) by values from a near infrared band (~860 nm) and the near IR values by a further IR band (~1240 nm) [24]. Figure 3.18 gives an example of the NDVI, derived from the red and the red-edge band of a 30 cm WorldView-3 image.

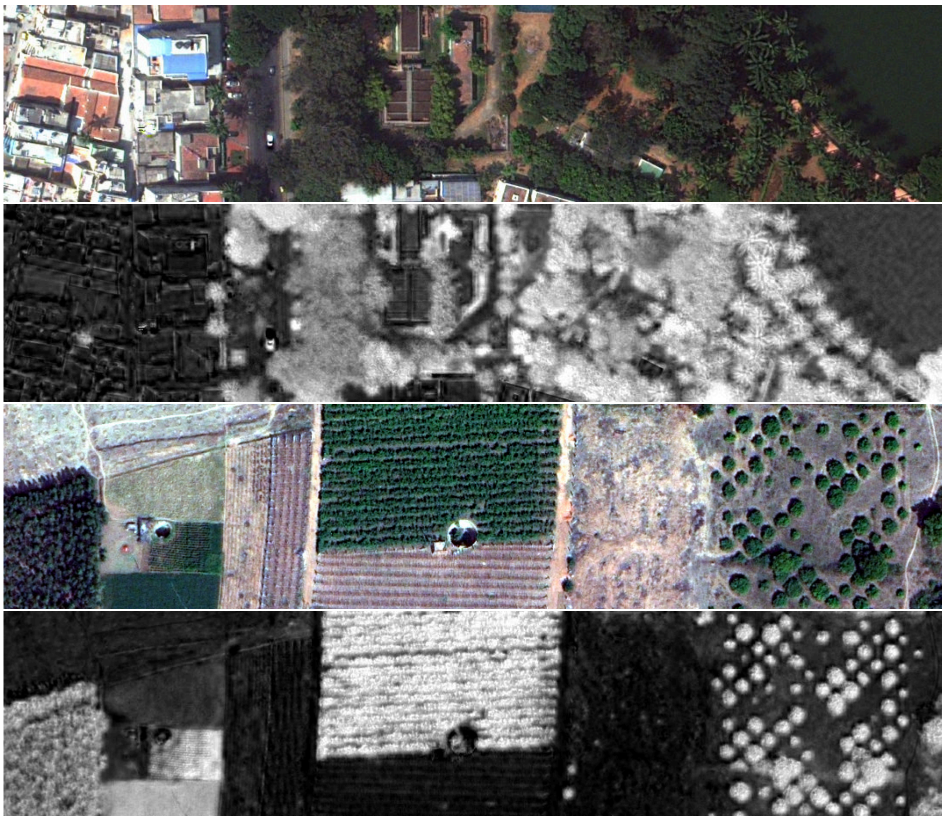


**Figure 3.18.:** Samples of the NDVI at two different locations. Vegetation has a high (bright) NDVI and bare ground or impervious surfaces have a low (dark) NDVI.

As the NDVI value is correlated with the chlorophyll content of the leaves, it is possible to draw limited conclusions regarding plant health [8]. This of course requires prior knowledge of the plant position, to which we will come in Section 6.

There are far more indices which are traditionally used for land cover classification, but as we do not want to take the traditional approach, but a deep learning based instead, we will not elaborate further on those indices. Nevertheless it is remarkable how well these simple, wavelength-derived indices work on a global scale [14].

### 3.2.4. Lidar

The term *Lidar* is an abbreviation for “light detection and ranging”; a technology for measuring three dimensional surfaces. Lidar devices emit laser pulses and measures their return time. Knowing the speed of light, the distance to the reflecting object can be calculated. Together with the sensor position and orientation, a three dimensional point cloud can be derived. In the field of remote sensing this is usually done to generate digital surface models of certain areas, using airplanes or drones. The devices used in airborne laser scanning employ a rotating laser in order to capture the height profile of a line. Similar to satellite sensors, the second imaging direction is covered by the direction of movement.

# 4. Models

In this section we will take a closer look at the models employed in this work. There are two main tasks we have to solve in the course of the computer experiments: *classification* and *semantic segmentation*. In the classification task, a class is assigned to an entire *image*, whereas in the segmentation task, a class is assigned to each *pixel*. For both of these tasks, we will use convolutional neural networks, each composed of the building blocks described in the theory section. To solve the segmentation task, we use the U-Net [65] architecture and different versions of it. For classification, the ResNet [28] network is used.

## 4.1. U-Net

The U-Net is a deep neural network that generates semantic segmentations. It was invented by Ronneberger et al. [65] in order to segment biomedical images of bacteria and is based on the SegNet architecture [6]. There are many other network architectures that are able to perform semantic segmentations, but there are several reasons why to choose the U-Net. First, it is a very simple architecture, which means it is easy to implement and to debug. Second, this simplicity makes the network fast. This is advantageous, as we want to apply the model to large areas within a reasonable amount of time. Furthermore, it was shown in the initial publication [65] that the U-Net is trainable with a small amount of training data. This is exactly what we need in this thesis, as most of the training data has to be created by hand, because no public training datasets exist for our case studies. Lastly, it has been shown in various cases, that the U-Net is a very capable network with high performance [30, 31, 46, 84].

As mentioned above, the U-Net is based on the SegNet [6] architecture. SegNet consists of an “encoding” and a “decoding” part. The encoder is an alternating series of convolution- and pooling-layers, which extracts features from the input, very much like an ordinary classifier. The decoder produces a segmentation map, based on the features derived in the encoder, by alternating transposed convolution layers (or upsampling) and convolution layers. SegNet is able to produce segmentations, but they are often lacking the needed detail and precision. The reason for this is, that the decoder cannot access information from the encoder, such as edges in the original images. The idea behind U-Net is to introduce *skip-connections* between encoder and decoder, at levels where the feature maps have the same lateral extent (see Fig. 4.1).

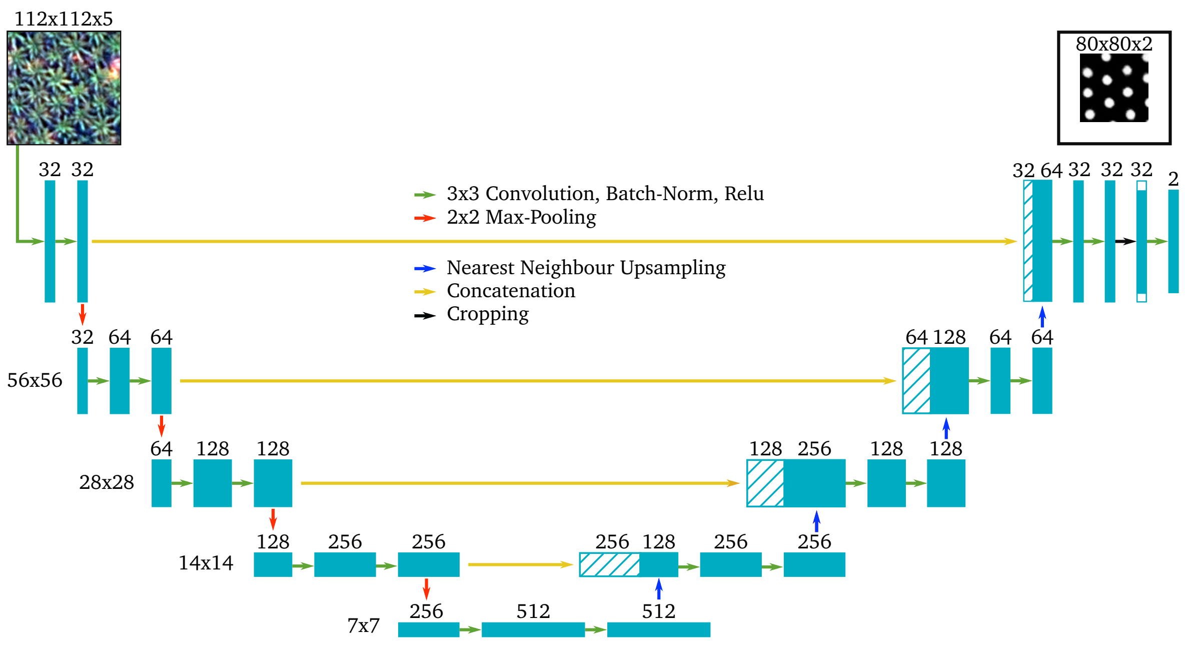


**Figure 4.1.:** U-Net architecture “A” [30, 22]: The U-Net has an “encoding” (left) and a “decoding” part (right). Information from earlier layers is fused with the output of later layers by concatenation. The last convolution has a kernel size of one.

As shown in Figure 4.1, the U-Net receives an image patch as input and produces a probability map (the segmentation) for predefined classes as output. Initially, the produced segmentation map has the same lateral dimension as the input (e.g. 112x112 pixels). But as the prediction quality in the segmentation map deteriorates close to the border [30], the output is here cropped by 16 pixels from each edge.

The network depicted in Figure 4.1 will be used later in this thesis - we call it “U-Net A”. It is an adaption of the network presented by Iglovikov et al. [30] for the task of land cover classification. We ported their implementation to Keras [12] with TensorFlow [54] as backend, and made slight modifications to it. “U-Net A comprises five stages. At each stage two 3x3 convolution operations are applied, each followed by batch normalization and the ReLU activation. The downsampling between the stages is performed by 2x2 pooling operations and - in contrast to the original implementation - we use nearest neighbor upsampling in the expanding part of the network, instead of transposed convolutions. The second adaptation we made is that batch normalization is only used in the contracting part of the network. These changes result from preliminary experiments and increase the speed of the network without decreasing the accuracy. The last convolution has a kernel size of one and is followed by a softmax activation.” [22] The softmax activation normalizes the network output, therby generating the final probability maps. In principle, the U-Net is capable of handling an arbitrary number of classes, but we will always consider just one class of interest (e.g. tree cover) and a background class.

### Training

In general, neural networks have to be trained with data that matches their output data format. The U-Net outputs a probability map; thus it has to be trained with one, or a mask respectively. The mask ideally labels all objects of interest. Figure 4.2 gives an example for the network input, the training mask and the output.

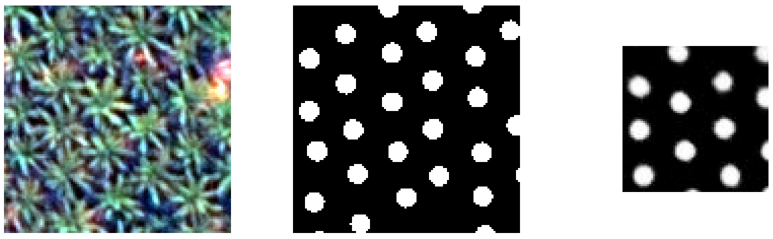


**Figure 4.2.:** The network receives an input (a) and generates an output (c). The output is optimized via back-propagation in order to match the ground truth mask (b) best. Here the network detects palms. Figure taken from [22].

### Inference

We want to apply the U-Net to images, which are much larger than the network input size, so the question is how we can achieve this. There are two options: first, we can instantiate a new U-Net model with a larger input size and copy the weights from our trained model to the new one. This is possible because the number of parameters in the U-Net does not depend on the input size. But still then, the maximum input size is limited by the available memory. Second, we can apply the network in a moving window fashion, as presented in Figure 4.3. In this case, the image is iteratively processed by moving the network input window over it in steps of the network output size or smaller. The resulting probability maps are concatenated (much like ordinary tiles) to create the final result. If one uses a step size smaller than the network output width, the resulting tiles overlap. In these overlap-areas the results have to be averaged. Both techniques can be combined with each other and also with test-time augmentation (see Sec. 3.1.10). By using test-time augmentation and an overlapping tiling scheme it is possible to increase the continuity and general quality of the segmentation.

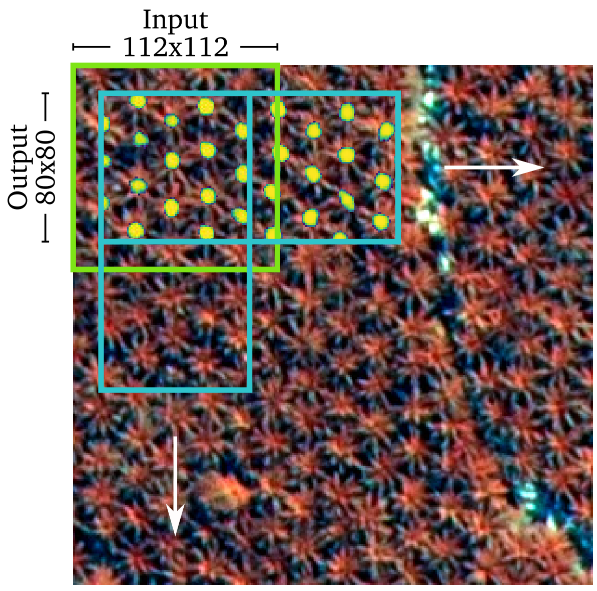


**Figure 4.3.:** The input window (green) is moved across the image in steps of the output window size (blue). At each position, the network returns a probability map, which are then fused to form the final result. Image taken from [22].

## 4.2. ResNet

The ResNet architecture was first presented in 2015 by He et al. [28] from Microsoft Research. By the time, this architecture won several competitions and set the new record for the ImageNet dataset classification task. The main idea behind ResNet is that its building blocks are designed to “learn residual functions with reference to the layer inputs, instead of learning unreferenced functions” [28]. Imagine an arbitrary input $x$ to a layer of a neural network and the optimal output $y$. In the case of an unreferenced function, the network directly approximates a function $f$ that generates $y$: $y = f(x)$. What is meant by “residual function” in this context, is that the new network now approximates $y = x + f(x)$. So instead of learning how the input has to be transformed to generate the desired output, the network learns the difference between input and optimal output. Figure 4.4 shows how this concept is applied to the layers of a convolutional network.

An observation from earlier models was, that adding more layers to an existing network not necessarily increased the performance. The key contribution of He et al. was to develop a layer, or *block*, that was able to easily learn the identity transformation (e.g. by setting $f(x) = 0$). Stacking these blocks on top of an existing network should ideally not deteriorate its performance, since the new blocks can learn the identity transformation, thereby retaining the base model’s performance. But the newly added blocks increase the number of degrees of freedom and the network depth, therefore enhancing the capacity to learn complex features. So in case there is a better representation of the data, the new network should be able to learn it.

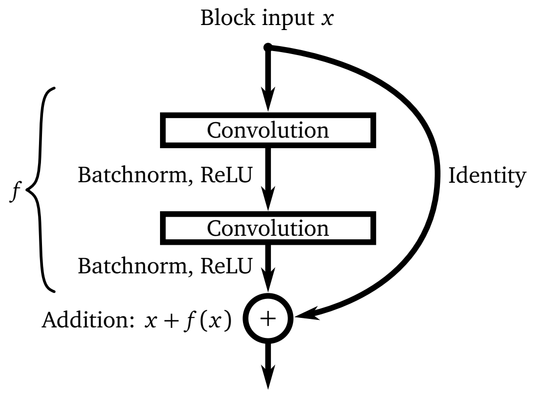


**Figure 4.4.:** The ResNet building block [28].

The core building block of the ResNet architecture is depicted in Figure 4.4. The block receives an input to which it applies a convolution, followed by batch normalization and the ReLU activation. This is repeated a second time and the input is added to the resulting feature map. With this architecture it is easier to learn the identity transformation, because the optimizer just has to drive the weights in the convolutional layers to zero. Further information about the role of skip connections and identity mappings can be found in the original publication [28] and in [29].

The two convolution layers in the block usually share the same set of hyper-parameters, such as the number of filters, kernel size and so on. We will call a set of blocks, that share the same hyper-parameters a *unit*. In order to obtain the full ResNet, several of these units are stacked on top of each other, followed by a fully connected layer. Therefore, the ResNet can be characterized by: 1) the number of units, 2) the number of blocks in each unit, 3) the hyper-parameters of the blocks in the units and 4) by the number of nodes in the fully connected output layer. The ResNets are named according to their total layer count; e.g. ResNet18 has 18 layers, and ResNet34 has 34. Figure 4.5 shows the detailed network architecture of ResNet18, including an initial convolution and the pooling layers. First, a convolution with a stride of two acts on the input image, followed by max pooling, which results in strong downsampling of the input. Then a unit with two of the base building blocks follows. The second unit also contains two blocks. But here, the first convolution has a stride of two, therefore it is downsampling the feature maps (see Sec. 3.1.2). As a consequence of the downsampling and the differing number of filters per convolution, input and output of the first block in the second unit differ in their shape. Therefore, the input's shape is matched via a strided 1x1 convolution, which is indicated by the dashed line in Figure 4.5.

### Training and Inference

The ResNet is trained in the same way as every other classifier: the network receives an image and the information, to which class the image belongs. This information is encoded as "one-hot vector" where all entries are zero, except the entry for the true class - which is set to one. The inference is done in the same way; the network receives an image and outputs the classification vector with the probabilities for each class. Test-time augmentation can be applied here as well.

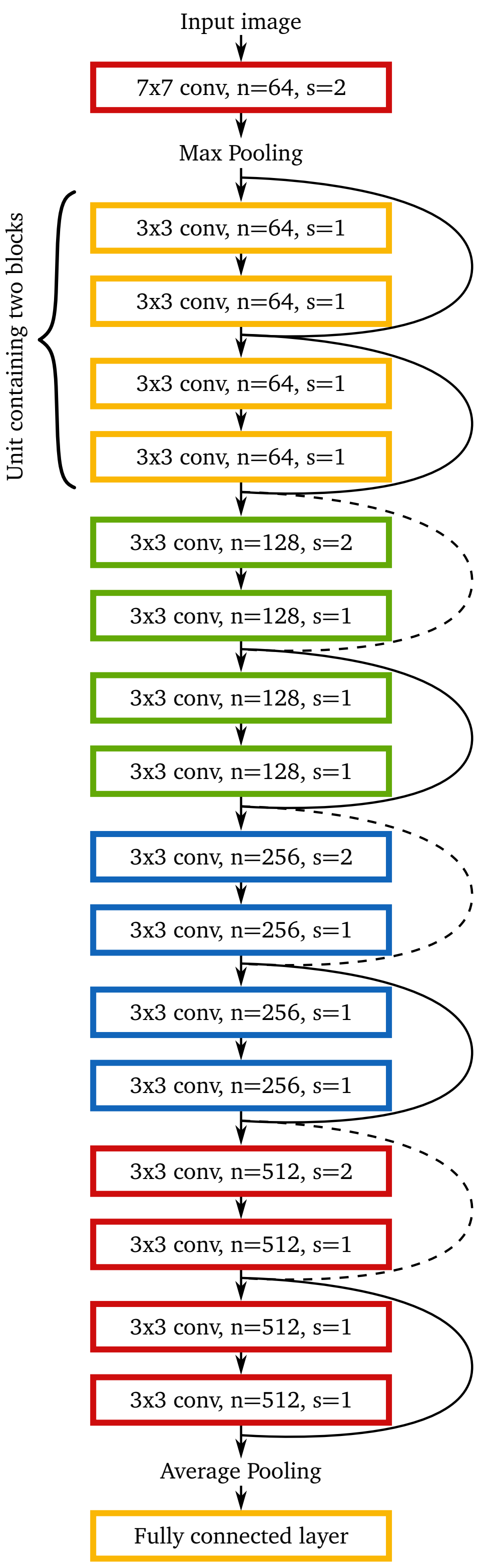


**Figure 4.5.:** Architecture of ResNet18, which is used in Section 6.3. Activations and additions are omitted for clarity.
s: stride, n: number of filters in convolutional layer

# 5. Study Areas

In this thesis we will work with three datasets coming from different regions of the world. Two of them are captured by satellites, and one by airplane. As a result, they differ in lateral and spectral resolution. Each dataset comes with specific questions we want to answer. This section describes each study area and lists the corresponding image properties. The training data will be described later in the introduction of each computer experiment.

## 5.1. Bengaluru, India

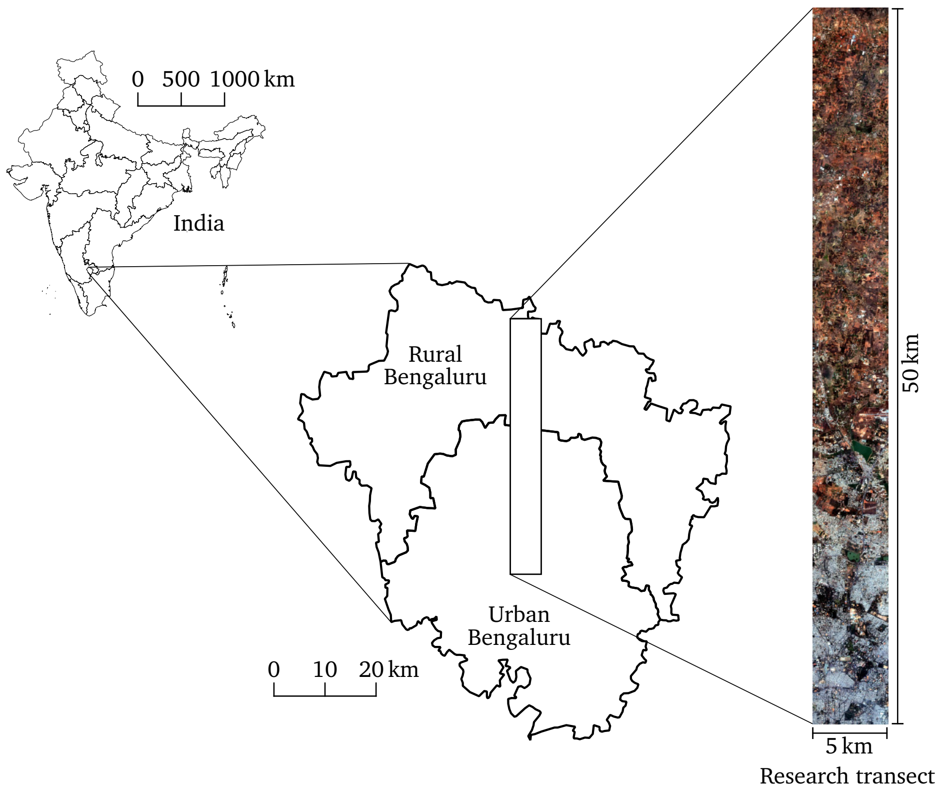


**Figure 5.1.:** The study transect is located in the northern part of Bengaluru and lies partly in the administrative regions “Urban Bengaluru” and partly in “Rural Bengaluru”.

Figure 5.1 depicts the first study site, which is located at Bengaluru, India (formerly Bangalore). Bengaluru is located in central, southern India at 12°59’N 77°35’E and is the capital of the state of Karnataka. It lies on the Deccan Plateau at a height of 900 m above mean sea level and has a moderate tropical climate with distinct wet and dry seasons [75]. Temperature ranges from 15°C in winter to 35°C in summer. Bengaluru is a rapidly growing city with an estimated population of 11.4 million in 2018 (2000: 5.6 million) [5]. This population growth is driven by the city’s fast economic development, which also triggers high construction activity. In turn, the city expands into the surrounding countryside, causing a significant loss in tree and vegetation cover [56, 57].

The city center is characterized by tall office buildings, interspersed with large housing complexes. Vegetation is found in parks, small gardens or alongside main roads, often in the form of large trees and coconut palms. In the outskirts of the city, large industrial complexes can be found, but also upper class housing and areas that are prepared for being built up in the future. Further north, settlements are sparse and with increasing distance from the city the villages become smaller. They are often located in the midst of agricultural fields, eucalyptus- or palm-plantations.

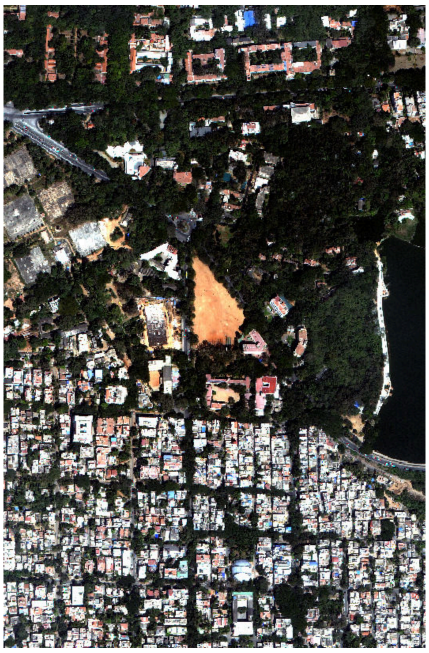


**Figure 5.2.:** Sample from the Bengaluru study site.

**Remote Sensing Data**

In the course of the research project FOR2432[1)], several satellite images of the Bengaluru region were acquired, taken at different points in time. The main dataset is a 5 km wide transect, ranging 50 km north from the city center. It was acquired in November 2016, at the beginning of the dry season. The image has a resolution of 31 cm per pixel and was captured by the WorldView-3 Satellite, therefore it comes with eight bands (see Fig. 3.16). The first (ultraviolet) of these bands was discarded, as it is strongly influenced by air humidity. Thus, all experiments work with the remaining seven bands or less.

## 5.2. Jambi, Indonesia

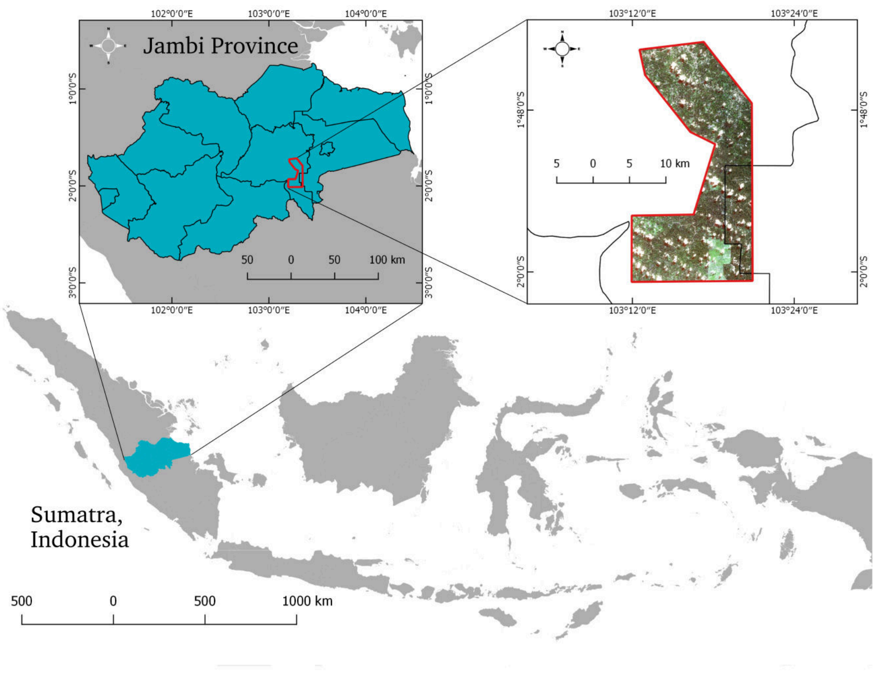


**Figure 5.3.:** The second study site is located 44 km south-west of Jambi City, Indonesia [22].

The following paragraph is taken from the publication related to this thesis [22]:

1) www.uni-kassel.de/go/for-2432

"The [second] study area is located 44 km south-west of Jambi City, Indonesia, and covers an area of about 348 km$^2$. It is part of a larger study area of a collaborative research project in the area (CRC990 - EEForTS) [18].

In particular, the flat lowland regions of Jambi have seen a dramatic increase in oil palm plantation area over the past decades, which mainly consists of very intensively managed large plantations but also smallholder plantations [42]. In the northern part of the study area, there are mainly smallholder oil palm plantations, with relatively small and loosely grouped patches of remnant forests and villages in between. In the South, large commercial oil palm plantations prevail with several hundreds of thousand of oil palms. There is a big gap in the vegetation cover in the south of the area depicted in the image at the top left of Figure 5.3, where all palms have been cleared and, in some parts of this gap area, young palms have been re-planted. This is a normal cycle in oil palm management: Old plantations that have passed their stage of high productivity are being removed and replaced by young plants. Accordingly, in larger areas - and also in our dataset - we find palm trees of different age classes and development stages. Younger palms are clearly separated as solitary plants and their crowns are still small. Young palm trees do not yet have the characteristic 'star'-shaped arrangement of the leaves when seen from above. As they get older, the palm trees close the gaps between the plants, until the crowns touch and start overlapping." [22]

Figure 5.4 gives two examples from the Jambi study site:

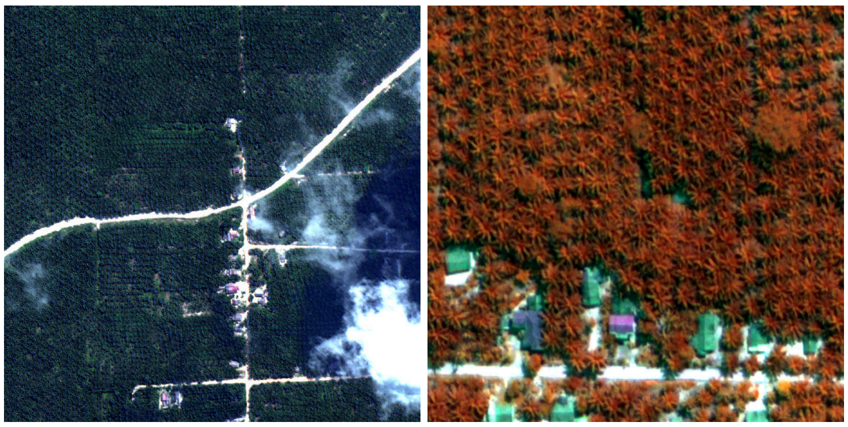


**Figure 5.4.:** Two samples of the Jambi study site at different magnifications and color compositions.

**Remote Sensing Data**
The imagery for this study site was captured on 2$^{nd}$ of July 2017 by the WorldView-2 satellite. It has a resolution of 40 cm and eight bands, of which we again discard the ultra-violet one.

## 5.3. Gartow, Germany

Our last study site (see Fig. 5.5) is a forest near Gartow, Lower Saxony, located at 53°01'N 11°27'E. It covers an area of approximately 140 km$^2$ and is situated around the nuclear waste interim storage facility of Gorleben. The forest is privately owned and therefore managed, with pines being the predominant tree species. Apart from pines, other species from temperate mixed forests such as oaks, beeches and firs can be found. The deciduous species are mostly grouped in the north-eastern part of the study transect and a large share of the oaks suffer from the oak procession moth.

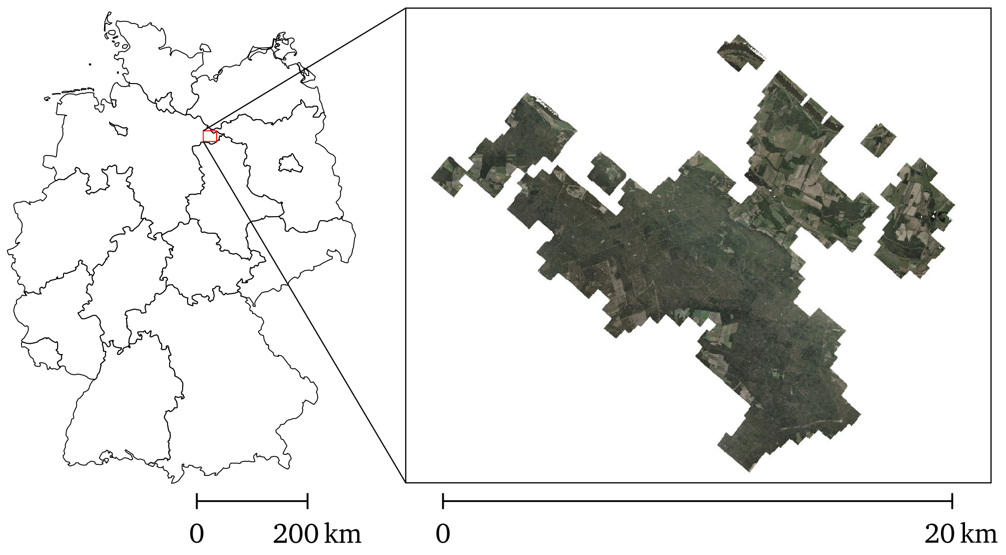


**Figure 5.5.:** The Gartow study site and its location in Germany.

**Remote Sensing Data**

The aerial imagery covers an area of approximately 140 km$^2$ and has a resolution of 5 cm per pixel with four bands (RGB, NIR). In addition to the optical imagery, a complete digital surface model was captured using Lidar. The point cloud resulting from the Lidar scan was rasterized and processed further to obtain a canopy height model (CHM) with a lateral resolution of 25cm and a height resolution of 10 cm.

Within the transect, an extensive forest inventory took place, during which about 7800 trees were surveyed. For each tree its species, position, diameter and other attributes were recorded. Therefore we are able to use this database for classification tasks.

Furthermore, we used the height information from the Lidar image to extract the position of every tree in the study area by local maximum detection. Around these positions, a watershed segmentation of the tree crowns was performed. This results in a set of polygons, where in the ideal case each corresponds to a tree. By this procedure approximately 1.8 million tree positions were derived. For the surveyed trees, the species, as well as the position and crown outline are known, which is depicted in Figure 5.6:

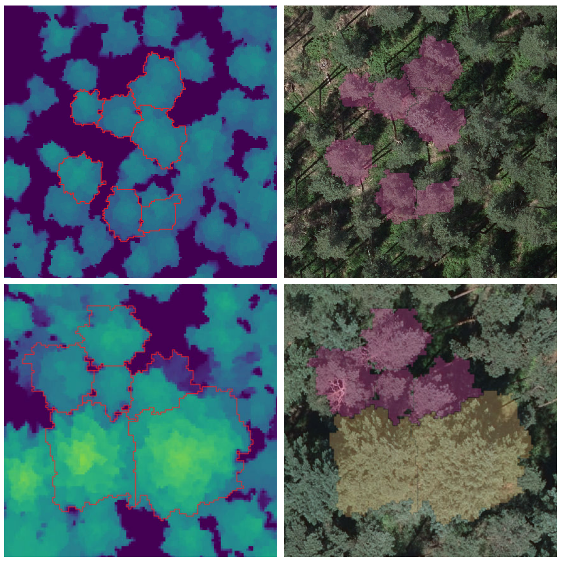


**Figure 5.6.:** Lidar images with watershed segmentation on the left. The same polygons, including class labels, on the right. As one can see, the Lidar and visual images do not match exactly. Pines are shown in purple, beeches in yellow.

# 6. Experiments

This section addresses several research questions, belonging to different topics. This subsection, and the experiments in them, are structured along these questions, ordered with increasing difficulty. The three greater topics and their related questions are:

1. Treecover segmentation
   - Which imaging bands are required for this task?
   - Which neural network works best?
2. Palm detection
   - Which accuracy and speed can be reached compared to existing methods?
   - Can the method used here be transferred from one region on earth to another?
3. Species classification
   - Which accuracy can be reached, when classifying trees in aerial images?
   - Which tree species can be separated?

For each experiment, the setup, including training data and network training is described. The results will be listed for each topic separately and each experiment receives its own discussion. A subsuming review and conclusion will follow at the end.

## 6.1. Tree Cover Mapping

In this section we used the U-Net to detect tree cover in satellite images of the Bengaluru Metropolitan Region in India. This task has a practical relevance, as it helps to assess the ongoing loss in tree cover in the region around the quickly growing city. As our WorldView-3 satellite images comprise seven imaging bands, we first assessed which of those bands are necessary to achieve a satisfying accuracy. We tried different band combinations, with and without the normalized difference vegetation index (NDVI) appended. The motivation behind this was to reduce the enormous file sizes of remote sensing images, thereby saving disk space and valuable time when reading images into memory.

Under the assumption that the band combination found in this so-called ablation study is best for various neural networks, we then assessed different networks. This assessment, however, had to be restricted to the U-Net class of neural networks, due to the virtually infinite number of possible architectures for solving this task. Three different networks with varying complexity were tested.

### 6.1.1. Labeled Data

This experiment was done using the Bengaluru dataset, described in Section 5.1. Within this dataset, 330 quadratic one hectare plots were sampled in a regular grid. In each of these plots, the tree crowns were delineated manually. No on-ground validation has been performed for this data, but it can be expected that the accuracy of the labels is sufficiently high, as the image resolution is high. This was checked by random visual inspection. Trees cover 24% of the dataset, the remainder is considered as background. Figure 6.1 gives an example of two labeled images.

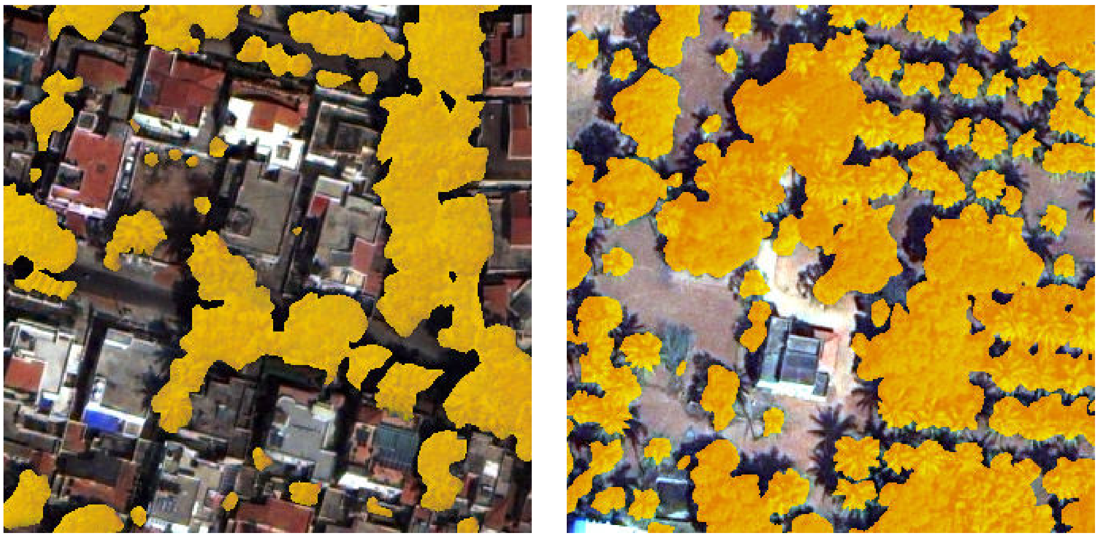


**Figure 6.1.:** Two of the 330 one hectare tiles with masked tree crowns.

### 6.1.2. Experimental Setup

**Ablation Study**

In this ablation study we tested which bands or indices are needed in order to detect tree cover. For this test we used U-Net A, described in Section 4.1. The annotated images were split into training and test set with a 70%-30% ratio. As the final result depends on the selected training and test data, we used five-fold cross-validation. The network was trained and tested with different numbers of image bands and indices. With each band combination the network was trained for 140 epochs (9100 gradient updates) using a batch size of 32. In one epoch, the network was fed with images with a total area equaling the area of the training dataset. The image patches were randomly cropped from the training images and had a size of 112x112 pixels. We used transformations from the so-called $\mathrm{Dih}_4$ group (90 degree rotations, reflections) in order to augment the training data. In addition to that, we used uniform random, channel-wise color shifts with an amplitude of half the standard deviation of that channel. This was the standard augmentation scheme, if not mentioned otherwise. No mean subtraction or other normalization was applied, as we were not interested in transferring the models obtained here to another dataset. For the training process the joint loss function of categorical cross entropy and intersection over union given by equation (3.16) was employed. All experiments related to semantic segmentation in this thesis are carried out using this loss function, as it proved to be best in preceding studies. The Adam optimizer with Nesterov momentum was used with a constant learning rate of $5 \cdot 10^{-5}$ (see Sec. 3.1.8). In order to measure the segmentation quality, we applied the network to the entire test set in form of a moving window after each epoch, as described in Section 4.1. The quality is quantified using the accuracy measure and the IoU from Section 3.1.5. All following experiments used the optimal band combination found here.

**Architecture Assessment**

In this part of the study we assessed which network architecture works best for tree cover segmentation. We tested three different architectures of the U-Net type with varying complexity. First, we identified the optimal learning rate, then we assessed how the networks behave during longer training. Finally, each network was trained according to the optimal scheme to inspect the results.

The first tested network was U-Net A, which was already presented in Section 4.1, so it will not be further explained at this point. U-Net A comes with approximately 7.8 million parameters. As tree covered areas are mostly green and have little variety in texture, we assume that a simpler network is able to deliver a performance comparable to a more complex network. Therefore we simplified U-Net A on empirical grounds, arriving at our second model, U-Net B. U-Net B comprises four stages with only one convolution per stage. These convolutions also feature less filters than in U-Net A, which results in a number of 260 000 trainable weights. The precise architecture is presented in the appendix (Fig. A.1).

An interesting alternative to the above U-Nets is the so-called U-ResNet [46, 84, 31]. In this network, the encoding part consists of a ResNet and also the decoder comprises residual blocks. As in the original U-Net, the output of each encoder stage is concatenated with the appropriate decoder stage. We used the implementation of the network with a ResNet-18 encoder from [82]. The corresponding architecture is depicted in Figure A.2. With 14 million parameters, this network is the largest of the three.

We first evaluate which learning rate works best for each network, then all networks are trained using their optimal learning rate. For all these experiments we use a batch size of 32 and an input window size of 224x224 pixels. This large window size has to be chosen, because the U-ResNet demands a minimum input size.

For the optimization of the learning rate, each model was trained on a constant set containing 70% of the labeled data. Each test is carried out over a period of 10 000 gradient update steps or 620 epochs, respectively. As before, we used the Adam optimizer. The learning rate was swept between $10^{-2}$ and $10^{-6}$, starting from the highest value and halving the learning rate after each training run. This results in nine training runs per network. To obtain the final measurement, the test metrics of each run were averaged over the last 100 epochs.

The next step was to assess the three models using the optimal parameters obtained before. We used the same training setup as for the learning rate test, now using five-fold cross-validation with a 70%-30% training-test split. The network was also trained longer, for 20 000 gradient updates, in order to be able to observe eventually occurring overfitting.

From the resulting training curves, we deduced the optimal training duration. With this setup, we finally trained all three networks and inspected the resulting imagery. For the inference, we performed test-time augmentation by rotating and reflecting the images, averaging the results.

### 6.1.3. Results

**Ablation Study**

The ablation study shows that the quality of the segmentation depends on the number of imaging bands used and also on whether the NDVI is included or not. Table 6.1 gives the resulting accuracy and intersection over union after 140 epochs of training, averaged over the last five epochs.

| Band combination | Accuracy | IoU |
|---|---|---|
| NIR1 | 91.7 | 66.4 |
| B, G, NIR1 | 93.5 | 73.3 |
| B, G, R | 93.7 | 73.6 |
| B, G, R, NIR1 | 94.7 | 77.7 |
| B, G, R, NIR1, NDVI | 94.4 | 87.9 |
| all seven bands | **94.8** | 78.0 |
| all seven, NDVI | 94.4 | **88.1** |

**Table 6.1.:** Results of the ablation study.

The network reaches accuracies between 91.7% for the near infrared band only and 94.8% when using all seven bands. Looking at the intersection over union, differences in performance become apparent. With only one band, the IoU is relatively low with only 66.4%. Using the blue, green and either the red or the near infrared channel results in an IoU of approximately 73%. When all seven bands are used, the network reaches an IoU of 78%. This is comparable to the performance when using (B, G, R, NIR1), which scores at 77.7%, respectively. An improvement in IoU is achieved, when the NDVI is added to the four channels used before. With 87.9% the performance is ten percentage points better compared to using all seven bands. Using all seven bands in conjunction with the NDVI yields an intersection over union of 88.1%, which is essentially at the same level with using four bands plus NDVI. Therefore, we considered the combination of RGB, NIR1 and NDVI as best compromise and trained all following networks with this combination.

### Architecture Assessment

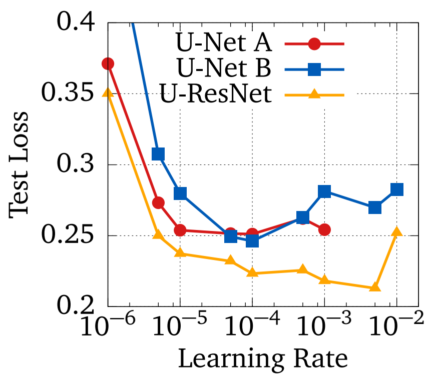


**Figure 6.2.:** Loss on the test set for different learning rates.

We first evaluated which learning rate is best for which model. Figure 6.2 shows that the loss increases for very low as well as for very high learning rates. In the case of U-Net A, the training process did not converge for learning rates higher than $10^{-3}$. For U-Nets A and B a learning rate of $10^{-4}$ resulted in the lowest test loss, for U-ResNet $5 \cdot 10^{-3}$. However, from empirical trials and from literature [31] we know, that such high learning rates induce strong fluctuations in the observables and therefore, a learning rate of $10^{-4}$ was chosen for U-ResNet as well. Consequently, we trained with a learning rate of $10^{-4}$ or lower from now on.

Following the training procedure described in Section 6.1.2, we trained the three networks using five-fold cross-validation. Figure 6.3 depicts the loss on the training and test set, averaged over all cross-validation runs:

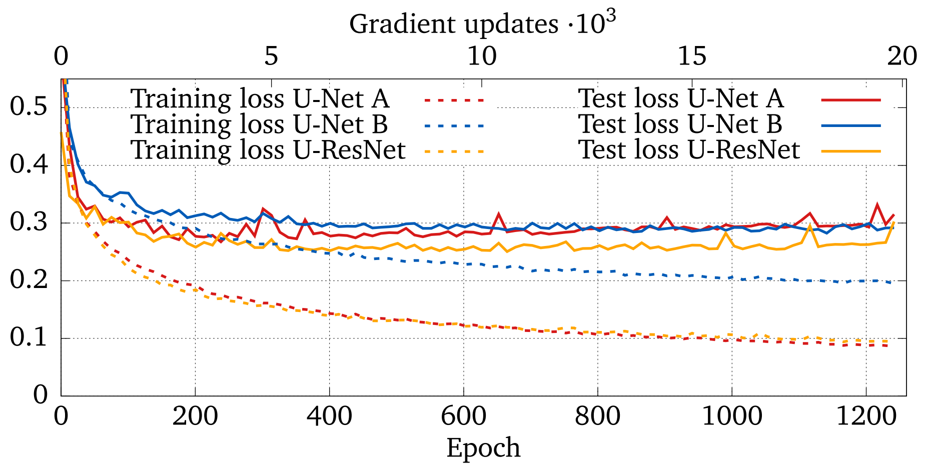


**Figure 6.3.:** Training (dashed) and test loss (solid) for the three tested networks. The presented curves are the mean of the five cross-validation runs.

Figure 6.3 shows that in general, the training loss is lower than the respective test loss - which obviously should be the case, as the network is optimized for the training set. The training loss of U-Net A and U-ResNet behaves similarly, reaching a final value of 0.1, and the test loss of U-Net A is higher than the one of U-ResNet. U-Net B has a higher training loss than the two other networks, but its test loss saturates at approximately the same level (around 0.3). One can observe a slight increase in the test loss of U-Net A and U-ResNet from 600 epochs on, which indicates that the models start to overfit. This is backed by the fact, that the spread between training and test loss increases, reaching 0.2 at the end of the training. With a value of approximately 0.1, this spread is about half for U-Net B. Furthermore, the test loss of U-Net B saturates, but it does not increase again, which indicates that it does not overfit.

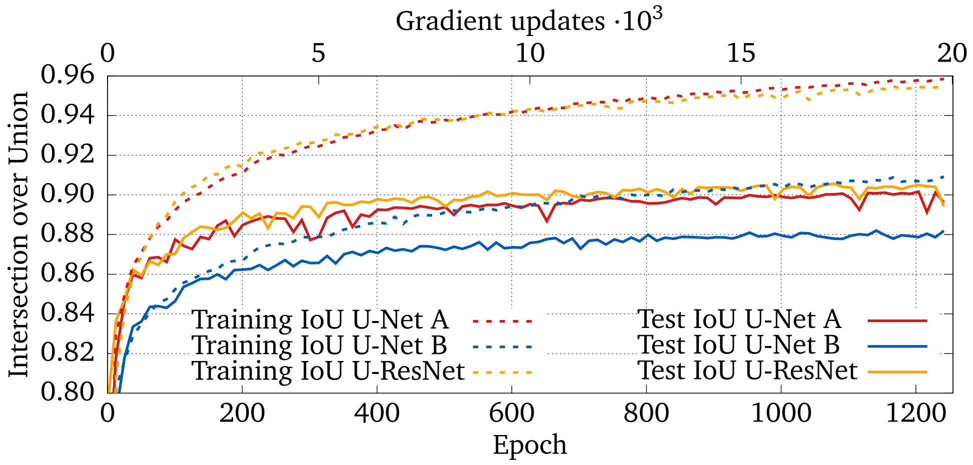


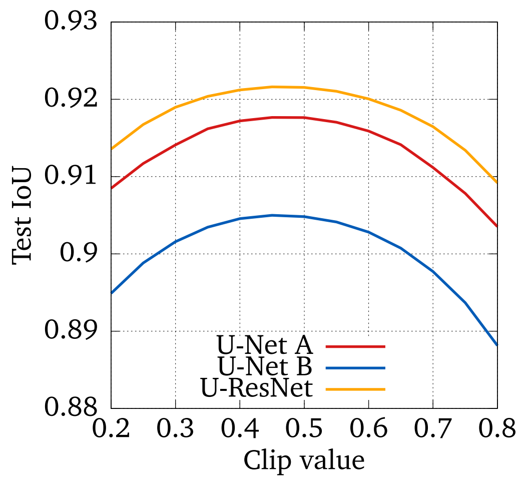
**Figure 6.4.:** Training (dashed) and test (solid) intersection over union for the three tested networks. The presented curves are the mean of the five cross-validation runs.

Figure 6.4 depicts the IoU of the tree cover class for the different networks, averaged over the five cross-validation runs. The spread of the metrics between training and test set can be observed here as well. Longer training does not increase the test metrics significantly, but brings the risk of overfitting, we decided to train the final runs for only 620 epochs, as from there on the test loss increases. To obtain the IoU values that can be expected from an average training run, we averaged the metric curves of all five cross-validation runs separately from epoch 520 to 620. Table 6.2 gives the lowest, the mean, and the highest of these five averages for each network. Calculating standard deviations from such a low number of runs is not reasonable and a higher number of runs was not possible due to computational constraints.

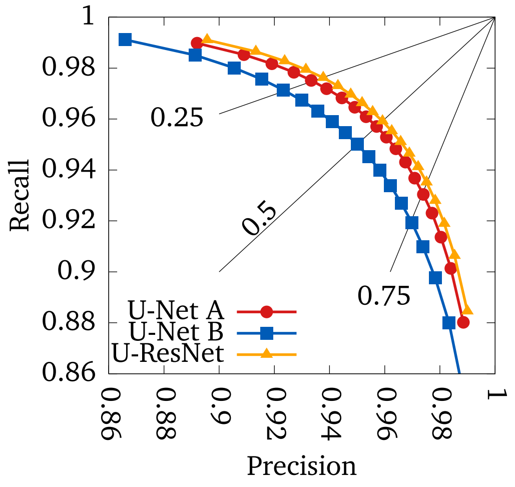
| | Test IoU [%] | | |
|---|---|---|---|
| | Lowest avg. | Mean avg. | Highest avg. |
| U-Net A | 88.6 | 89.9 | 91.0 |
| U-Net B | 86.8 | 87.9 | 89.7 |
| U-ResNet | 89.5 | 90.3 | 91.5 |

**Table 6.2.:** Average IoU calculated from the area marked in Figure 6.4.

The final training runs were carried out over 10 000 gradient updates or 620 epochs. For the first 7500 updates, the learning rate was again set to $10^{-4}$ and then lowered to $5 \cdot 10^{-5}$, to improve the optimizer's convergence. The resulting training curves can be viewed in the appendix (Figures A.4 and A.5). Before the results can be inspected, the optimal clip value for transforming the probability maps into binary masks has to be determined. Probabilities above the clip value will be counted as tree cover.



**(a)**



**(b)**

**Figure 6.5.:** Test IoU as a function of the clip value (a) and the trade-off between precision and recall for different clip values (b).

Figure 6.5a shows a broad maximum of the IoU between a clip value of 0.45 and 0.5. According to 6.5b, precision and recall can be balanced to some extent and the best compromise is achieved at a clip value of 0.5. Therefore we selected 0.5 as clip value, which is in line with the expected optimal value for a two-class problem. As we used test-time augmentation, the networks reach a higher IoU than in the cross-validation test. After clipping, U-Net A reaches an IoU of 91.8%, U-Net B 90.5% and U-ResNet 92.2%. Figure 6.6 shows a sample of the resulting tree cover masks.

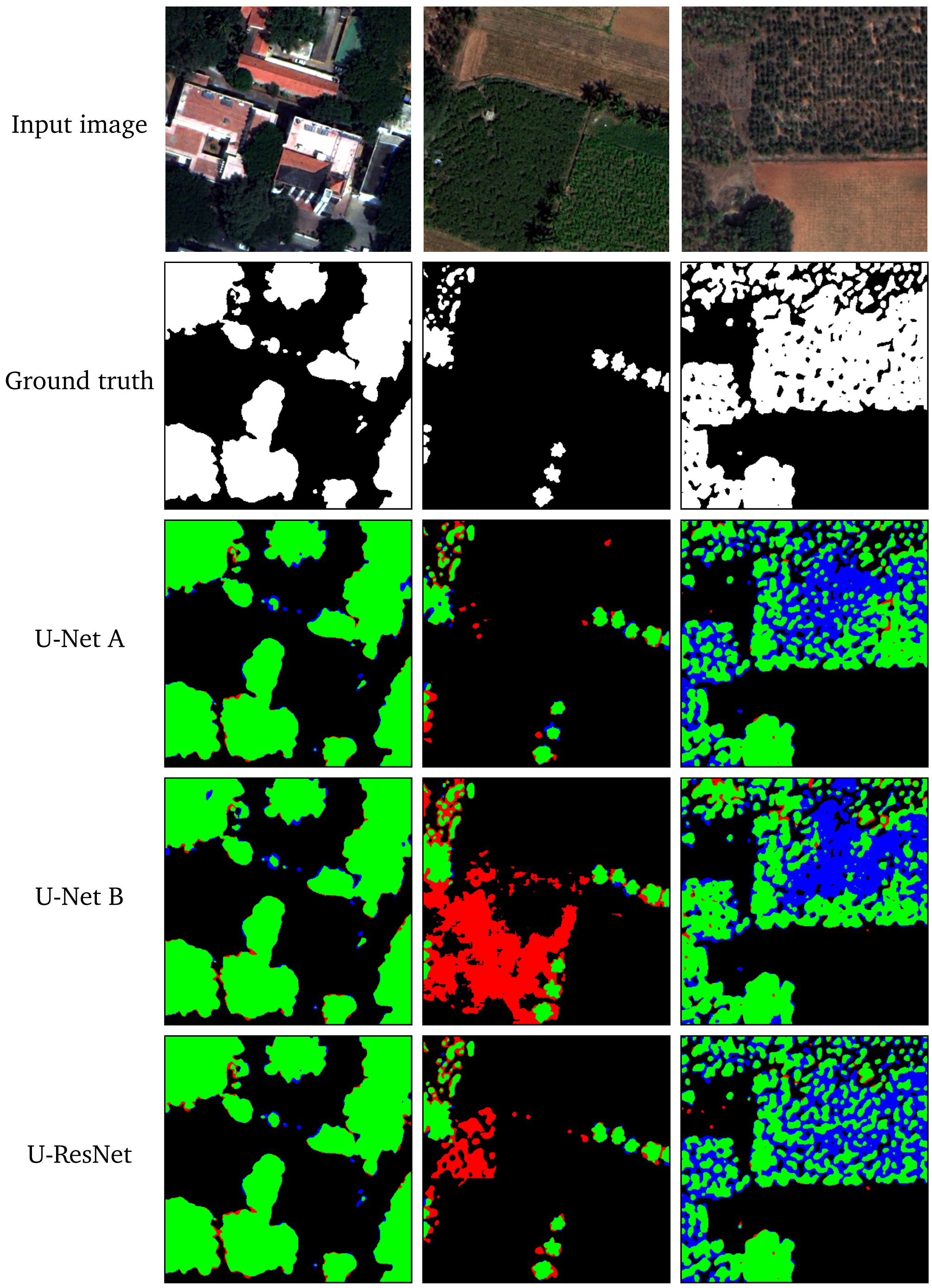
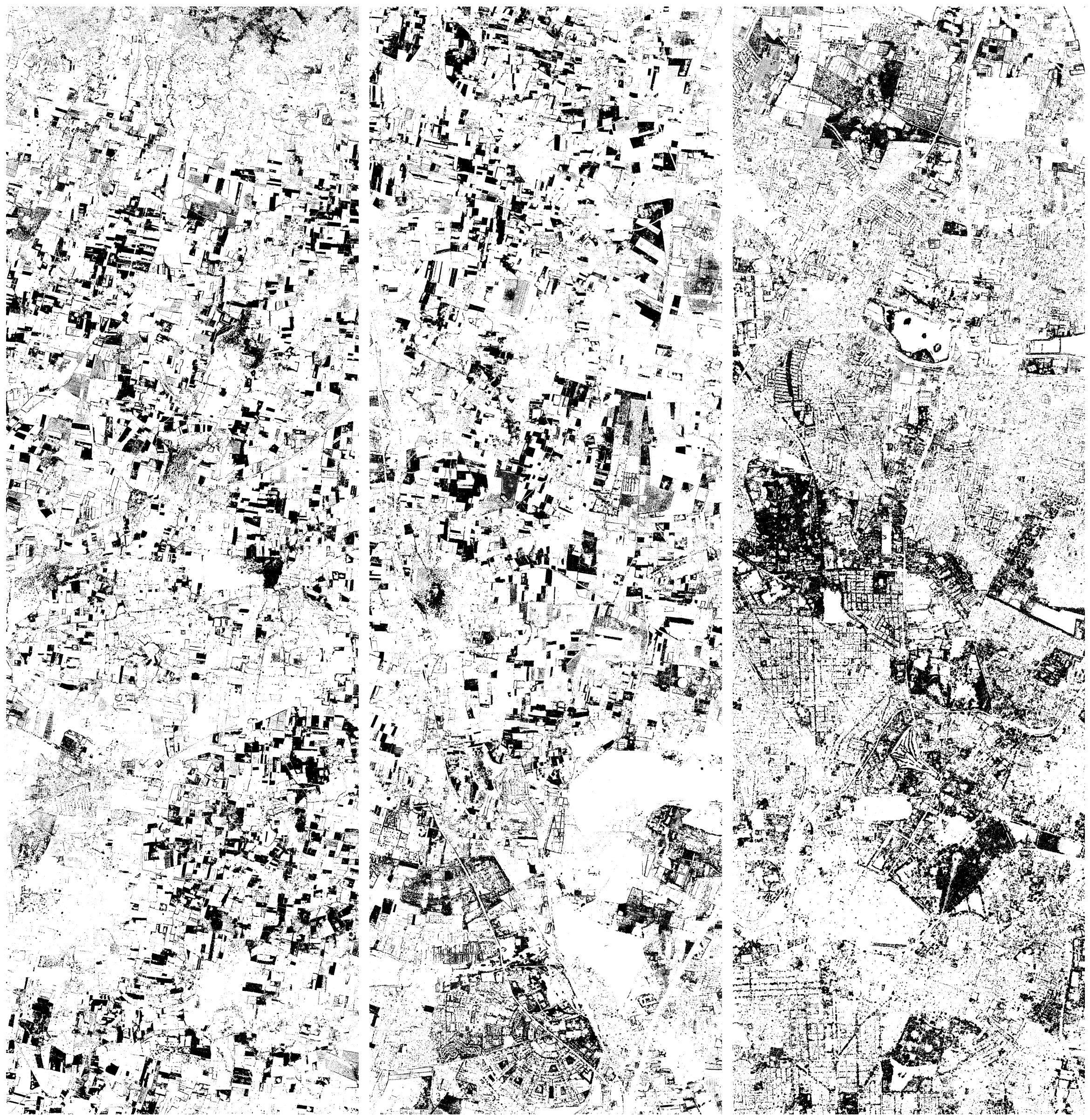


**Figure 6.6.:** Input images, ground truth and the output of the three networks after clipping. Correctly detected pixels are masked green, false positives in red and false negatives in blue.

The selected images demonstrate the commonalities and differences between the networks: While all three networks agree well when it comes to segmenting fully grown trees (first column), they differ on agricultural fields and tree plantations (second and third column). U-Net B falsely marks the agricultural areas as tree cover and to some lesser extent, U-ResNet fails here as well. In contrast, U-Net A discriminates better between crop fields and trees. On the plantation (third column), U-ResNet resolves the finest details and U-Net B produces the coarsest segmentation. U-Net A performs between the two others.

**Application to the entire transect**

As U-Net A and U-ResNet deliver a comparable IoU (see Fig. 6.5a), but U-Net A distinguishes better between crop fields and tree cover, we applied it to the entire transect of 5 km x 50 km. Figure 6.7 depicts the resulting tree cover maps. As the transect has a high aspect ratio, it was split into three tiles.



**(a)** North **(b)** Center **(c)** South

**Figure 6.7.:** Tree cover map for the northern, center and southern thirds of the Bengaluru transect. Tree cover is shown in black. Each tile has a size of $1.\overline{6}$ km x $16.\overline{6}$ km; north is up.

Figure 6.7 shows that trees are found in clusters of varying size throughout the entire transect. On average, 20% of the transect are covered by trees. In the North and center, trees are mostly found in an agricultural context, such as plantations and rarely in natural stands. In contrast, the trees in the southern part, where the city center is located, are mostly found in parks, gardens and alongside roads. On the contrary to what one might expect, the largest continuous tree cover is found in the city's parks and not as a forest in the rural area.

Figure 6.8 gives a magnified view of different typical areas within the transect:

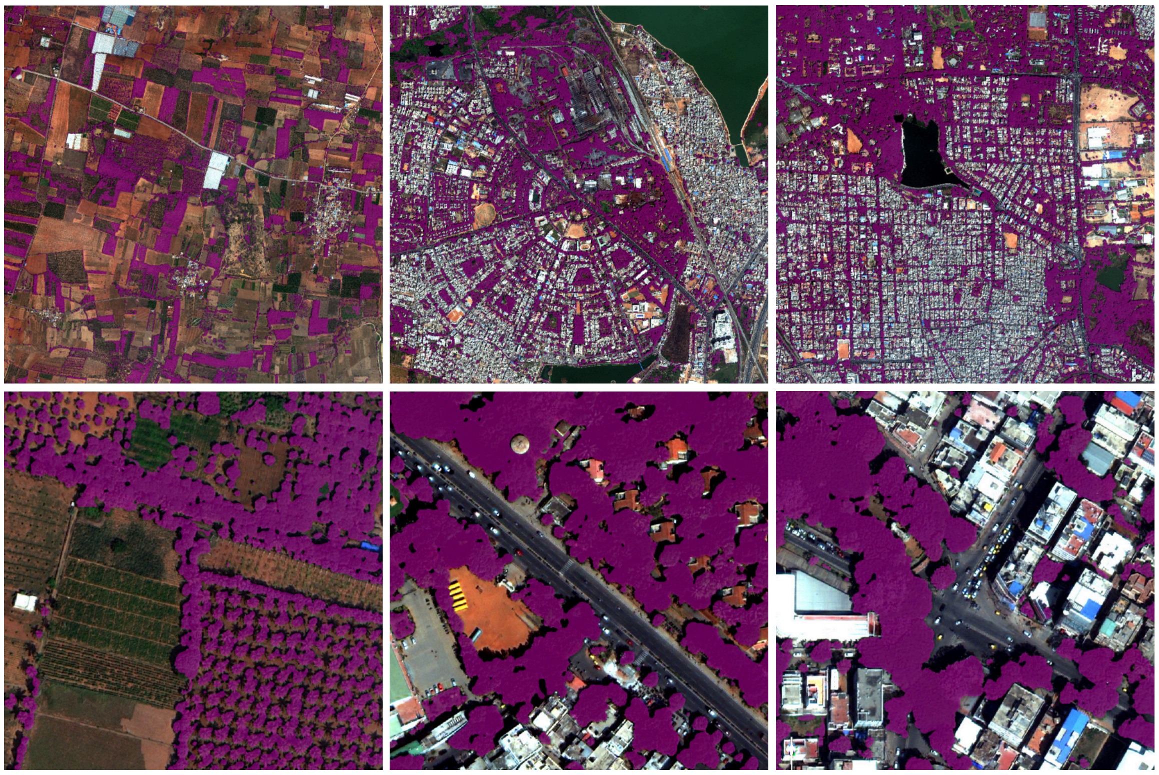


**Figure 6.8.:** Magnifications of different generic locations in the Bengaluru region. Top row: 2.5 km x 2.5 km, bottom row: 200 m x 200 m.

### 6.1.4. Discussion

#### Ablation Study

The ablation study showed that while more imaging bands do improve the performance in terms of IoU, not all of them are necessary for having a good performance in the tree cover segmentation. The yellow, red edge and NIR2 bands do not carry significant additional information. Although we did not test this explicitly, it has to be mentioned that the red edge, NIR1 and NIR2 bands are interchangeable when it comes to the calculation of the normalized difference vegetation index (NDVI). This is because they carry similar information and NIR1 and NIR2 overlap, as can be seen in Figure 3.16. Concatenation of the NDVI increased the network performance by about ten percentage points, which probably comes from the fact that the NDVI on its own already discriminates well between vegetation and non-vegetation. This shows that not only the traditional machine learning methods but also neural networks benefit from handcrafted features, which the NDVI is. We considered the combination of blue, green, red, NIR1 and NDVI as the best band combination and thus worked with it throughout the rest of the thesis.

#### Architecture Assessment

During the network assessment and the parameter optimization we found out that too high or too low learning rates yield sub-optimal results. Values in the order of $10^{-4}$ proved to be best for all U-Nets. As our dataset contained only 330 images, the networks started to overfit after about 620 epochs with the given training setup. We had to stop training at that point, because we wanted to apply the networks to the whole 250 km$^2$ of the transect and our training dataset sampled just 1.3% of that area.

Averaged over five cross-validation runs and 100 epochs, U-Net A reached an IoU of 89.9%, U-Net B 87.9% and U-ResNet 90.3%. Although the number of their parameters varies by more than one order of magnitude (A 7.8M, B 0.26M, Res 14M), the performance of the networks is comparable. This indicates that the problem of tree cover detection can be solved by networks which are simpler than the U-Nets used for other tasks [30, 31, 46, 84]. For U-Net B the spread between training and test metrics was lowest, which indicates that it is overfitting the least. This was expected since this network has the lowest number of parameters.

The network training for the final inference was carried out using the optimal parameters obtained before. A clip value of 0.5 proved to be best for transforming the probability maps into binary masks. U-Net A reached an IoU of 91.8%, U-Net B 90.5% and U-ResNet 92.2% on this specific division into training and test set. These values are about 0.8 percentage points higher than the highest averages given in Table 6.2. A quick test showed that the test-time augmentation improves the results by only 0.5 percentage points, which may indicate that the IoU on the whole transect is lower.

In order to obtain even better segmentation quality, it would be possible to combine the output of the different networks. Furthermore, a small network based on the building blocks of ResNet would be an interesting object of research.

The resulting tree cover map of the Bengaluru Metropolitan Region can now be used to derive different quantities of interest used in remote sensing, such as the tree cover density, or with additional information, also the biomass. Furthermore, the networks could potentially be applied to satellite images captured at other points in time in order to track changes. As the accuracy in terms of IoU is high, the simple difference of the tree cover masks would reveal changes. However, seasonal changes and changing atmospheric conditions pose a challenge to the neural networks and require further research. In order to tackle these problems, one can employ different tools, such as atmospheric correction algorithms, normalization of the input images or re-training and fine-tuning of the networks. In the upcoming second part of the experiments, we revisit some of these methods.

## 6.2. Palm Detection

This section addresses the detection of palm trees in satellite images. The contents presented here are partly identical to the publication [22]. It was published together with Nils Nölke, Alejandro Agostini, Kira Urban, Florentin Wörgötter and Christop Kleinn in the MDPI Remote Sensing Journal under the title "Large Scale Palm Tree Detection in High Resolution Satellite Images Using U-Net". Passages extracted from the article are again marked by quotation marks.

Palm trees can be considered as the "Hydrogen atom of forestry": they are widely abundant in tropical countries, have been studied well because of their high economical importance, and have a simple shape when seen from above. The demand for palm oil has steadily increased over the past decades, since it is used in food products, cosmetics and even as fuel [13]. Planting oil palms offers a significant source of revenue in tropical countries and new plantations are one of the root causes for deforestation. These grounds provide the motivation for the development of an algorithm for the localization and the counting of palm trees. This would allow the close monitoring of even the smallest plantation, including the control of illegal ones. Having such detailed information is relevant for three groups of stakeholders: First, for plantation managers, who can better predict their yield. Second, for government institutions, as they can control tax computation and also the adherence to given concessions. Third, for NGOs and also the oil-processing industry, as they can monitor the impact on the environment and the compliance with local protection laws [77].

In order to locate the palm positions, we employed U-Nets A and B, which were already presented in Section 6.1. The description of the architectures can be found in Sections 4.1 and Appendix A.1. U-ResNet is not included, as it was evaluated after the writing of the publication. The networks generate a "palm-probability map" from which the palm positions can be inferred by local maximum detection. In this computer experiment we worked with the Jambi dataset from Indonesia and the Bengaluru dataset from India. We compared our U-Net based method with the state of the art method [47] in terms of accuracy and speed. Furthermore, we analyzed the transferability of our method from one dataset to another. This proved to be challenging, as the two datasets (Jambi / Bengaluru) differ in atmospheric conditions and therefore their spectral signature. In addition to that, the contexts in which the palms occur differ in both datasets. Moreover, we compared palm positions found by the network with human labeled data in a small survey involving five persons.

### 6.2.1. Labeled Data

"For generating the training data we first manually digitized the palm tree crown centers. Around these center points, all pixels within a radius of 2 m are then marked as 'palm' - this is the ground truth mask. Within the study area in Jambi, we randomly sampled 160 quadratic one hectare plots, wherein a total of 10 679 palms were marked. In addition to that, we marked 4600 non-palm points, which are later used for the training of the classifier (Section 6.2.3). The training data was collected on the entire dataset, as we are interested in evaluating the accuracy of our model on a large scale.

The Bengaluru training dataset is structured differently: here we selected nine different areas of interest of varying sizes (between 1 and 60 ha). These tiles were selected with the aim of including as many different contexts as possible. Within those tiles we marked 1124 coconut palms for training and 1418 for testing. During the entire labeling process, different band combinations with different contrasts were used in order to ensure high precision." [22]

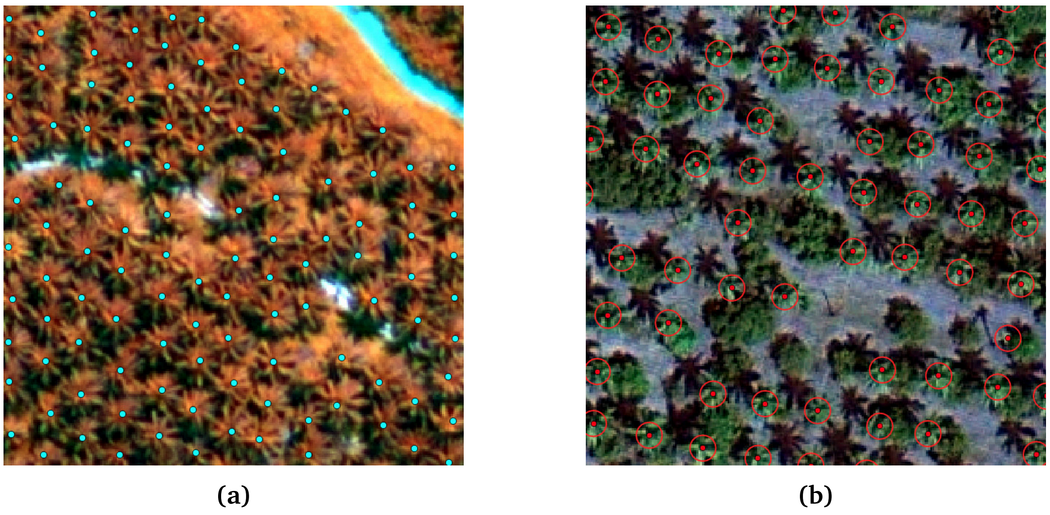


**Figure 6.9.:** "Figure (a) shows a one-hectare tile in the Jambi dataset in false colors. An example from the Bengaluru dataset is depicted in (b), together with the two meter circle, within which all pixels are marked as palm." [22]

### 6.2.2. Method Description

"Our approach is comparable to existing ones, in so far as it slides an input window over the area of interest. The difference is that it uses a much larger window size, as can be seen in Figure 6.10. At each position, an entire segmentation map is produced, instead of a single probability. This allows the detection of several palms at once. Accordingly, the step width can be much larger and far less patches have to be processed." [22]

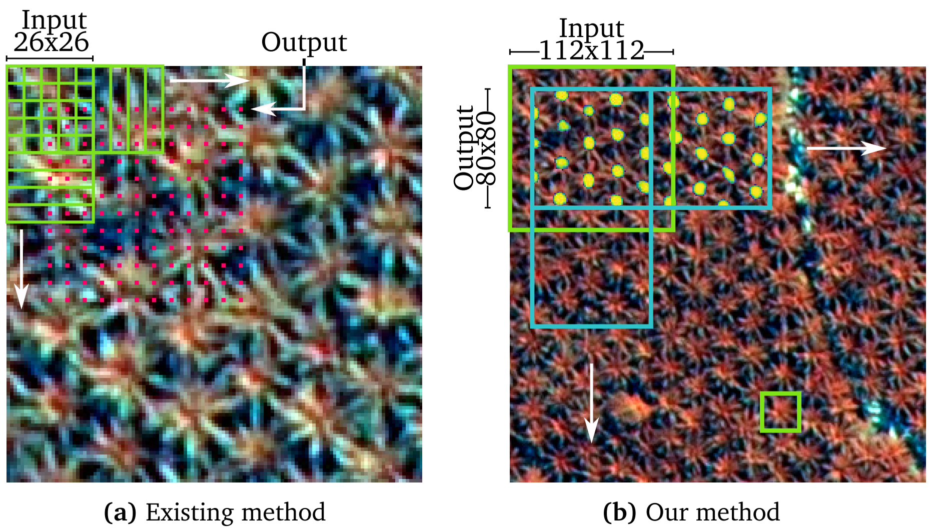


**Figure 6.10.:** "In (a) the existing method is shown [48]. A classifier window (green) is moved across the image in steps of a few pixels. At each position (red), the classifier calculates a "palm probability", from which a coarse probability map is created. In our method (b) the input window (green) is moved over the image in larger steps. At each position, the segmentation is calculated. The output window (blue) is slightly smaller than the input, because it is cropped due to the lower prediction quality at the borders. The palm positions can then be inferred from each probability map. The small rectangle at the bottom of (b) has the size of the classifier input in [48], scaled up to match our image resolution ($26 \times 26$ pixels)." [22]

"The network output shown in Figures 4.2 and 6.10b is smoothed with a Gaussian filter with a standard-deviation of 1.2 m, which equals 3 or 4 pixels, depending on the dataset. Then we perform a local maximum detection. SciPy's [34] peak local max function with a minimum distance of 1.2 m, an absolute threshold of 0.15, and a relative threshold of 0.1 is used. The resulting palm positions are then classified as true positive, false positive, or false negative. Peak positions closer than 3.2 m to a true position are counted as true positives, while ensuring that each true palm position can count for only one true positive. Predictions closer than 3.2 m to the global image border were left out of the accuracy assessment in order to avoid boundary effects. The radius of 3.2 m approximately equals two thirds of the crown radius of a full grown palm.

When evaluating the network performance, the inference is done on all images in the test dataset. The true positives, false positives, and false negatives are summed up across all images, then the performance metrics are calculated." [22]

### 6.2.3. Experimental Setup

#### Network Training

"We trained the two different U-Net architectures A and B on the WorldView-2 imagery in Jambi and evaluated their performance. Apart from that, we benchmarked our method against the existing classifier based on [48, 47] with regard to accuracy and computational performance. Then we re-trained and transferred the networks to the WorldView-3 imagery in Bengaluru and assessed their accuracy under these new conditions. Lastly, we utilized the full potential of the U-Net and applied it on a large area. The computer employed was equipped with an Intel Xeon 6136, 96 Gb of memory and two Nvidia GeForce 1080Ti graphics cards, which were both used.

Based on previous experiments, we reduced the original number of bands in both WorldView images to four bands (R, G, B, NIR) and added an additional band: the normalized difference vegetation index (NDVI), so that the network input has five bands in total (R, G, B, NIR, NDVI). Then the images were normalized by subtracting the dataset mean and dividing by the standard deviation for each band separately. The NDVI band remained untouched. In order to train the U-Net, it was fed with randomly cropped image tiles of size $112 \times 112$ pixels and the corresponding masks in batches of 16 samples. We used random transformations from the $\mathrm{Dih}_4$ symmetry group (the symmetry group of the square; 90 degree rotations and reflections) in order to artificially increase the amount of training data. This process is called data augmentation. A combined loss function of categorical cross entropy and the negative logarithm of the intersection over union of mask and prediction was employed, as described by [30]. We used the Adam optimizer [38] with Nesterov momentum [17].

In order to evaluate the performance of U-Nets A and B on the Jambi dataset, we performed a 10-fold cross-validation with a 70%-30% split into training and test data. This was necessary because the prediction metrics heavily depend on the selected training and test images. In each run, the network was trained for 600 epochs with 35 steps per epoch. One epoch corresponds to feeding images with a total area equaling the total training area to the network. The initial learning rate was set to $5 \cdot 10^{-5}$ and first lowered to $10^{-5}$ after 350 epochs, then to $5 \cdot 10^{-6}$ after 450 epochs. These parameters correspond to the best performance obtained from empirical evaluations." [22]

#### Comparison with Existing Methods

"To compare the AlexNet architecture described in [47] with our method, we trained it on the Jambi dataset, again performing a 10-fold cross-validation with exactly the same split into training and test images. Given the dataset size of 11,600 images, the training split of 70%, and the input image size of 26 x 26 pixels, we have approximately 5.5 million pixels available for training. Together with data augmentation, we believe this amount of data is enough to train the 790 000 parameters of AlexNet. Training was done for 100 epochs using a batch size of 16, with 100 steps per epoch, and the same augmentation as before. The initial learning rate was set to $3 \cdot 10^{-5}$ and first lowered to $10^{-5}$ after 30 epochs, then to $5 \cdot 10^{-6}$ after 50 epochs. This learning rate schedule was optimized by trial and error in order to improve the final network accuracy. We used the categorical cross entropy as loss function and the same optimizer as for the U-Net. The coarse probability maps, resulting from moving the classifier over the test images, were upsampled to match a resolution of 0.4 m. Afterwards, palms were searched with the method described in Section 6.2.2, the only difference being a threshold value of 0.5 instead of 0.15 for the peak detection. This different threshold was the result of an optimization using nested intervals.” [22]

### Speed Benchmark

“For an independent performance validation we benchmark our approach against the the approach by Li et al. who used the AlexNet model [41]. This study is the only one that provided the required information about the exact network architecture in our literature revision. All models were tested on the same hardware with the same environment on an image of 4 $km^2$ or $5000 \times 5000$ pixels. In order to reduce CPU calculation overhead and improve GPU utilization, we transferred the U-Net weights gained from training on $112 \times 112$ pixel tiles to a model with the same architecture but 512 pixels input window size and, therefore, larger output size. In order to test if the increased input window size affects the model accuracy, we performed an evaluation on a subset of 1700 palms. During the benchmark, we neglected the time it took to pre-process the images and took the pure inference time only. The timings were taken after one “warmup” run and averaged over 30 repetitions.” [22]

### Transferability of Pre-Trained Network

“We applied the U-Nets, which had been pre-trained on oil palms in Jambi, to the coconut palm trees in Bengaluru. Both datasets differ slightly in their spatial resolution, as well as in the atmospheric conditions. The environmental contexts in which oil and coconut palms grow, however, are significantly different (Figure 6.12), and this is the major challenge for the transferability of the network.

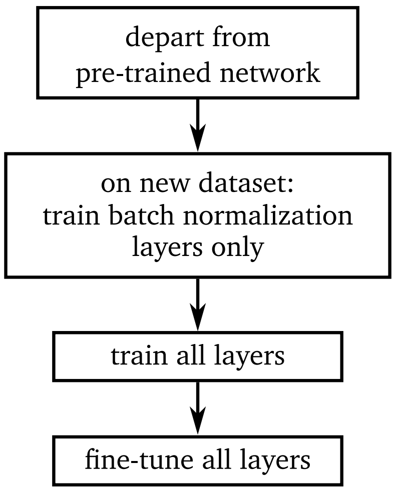
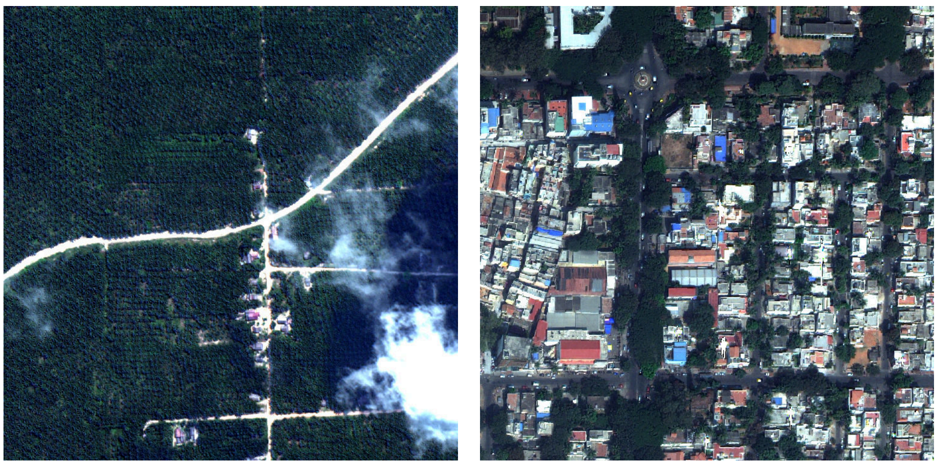

depart from pre-trained network
on new dataset: train batch normalization layers only
train all layers
fine-tune all layers

**Figure 6.11.:** The transfer procedure comprises three steps: exclusive training of the batch normalization layers, full network training, and fine-tuning.

Therefore, directly applying a model pre-trained on one dataset to another may yield bad results. Since collecting massive amounts of new training data was unfeasible, we followed a *transfer learning* strategy by normalizing the training images. The batch normalization layers in our network learn the mean and standard deviation of the activations on the training dataset, thus they adapt to the color spectrum. In contrast to that, the convolutional layers adapt to low level spatial features and their combination into higher level representations of the data. Since both datasets differ only by 10 cm in resolution, we assumed that the kernels learned by the convolutional layers are still valid. However, the color spectrum changed due to the different atmospheric conditions and varying surface materials. To overcome this, we optimized the batch normalization layers for the new color spectrum, which speeds up the training process. Subsequently, we performed minimal re-training of the entire network. The training procedure comprises three steps (see Figure 6.11): We departed from a network, which had been pre-trained on the whole WorldView-2 imagery in Jambi for 600 epochs according to the described scheme. In the first step, the learning rate was set to $10^{-2}$ and only the batch normalization layers were trained for 700 gradient updates (which equates to showing 700 image batches to the network, not to confuse with epochs). Second, we trained all layers for 700 gradient updates with a learning rate of $10^{-3}$. The third step is fine-tuning, which we did for another 700 updates with a learning rate of $10^{-5}$. The training set contained 1124 palms and the test set 1418. Labeling the 1124 palms in the training dataset took about one to two hours and was therefore considered as an acceptable amount of work for transferring the network." [22]



**(a)** Jambi dataset **(b)** Bengaluru dataset

**Figure 6.12.:** "Sample images from the two datasets." [22]

### Comparison of Neural Network and Humans

In order to evaluate the performance of U-Net A in comparison with humans, we conducted a small survey with five test persons. All test persons are doctoral candidates working in the field of remote sensing and acquainted with multispectral imagery. The survey comprised six images from the Jambi dataset, showing different scenarios: 1) young palms, 2) adult palms, 3) palms of mixed age, 4) palms in shadows, 5) palms in shadows and under clouds and 6) a setting considered difficult because of mixed vegetation. In total, these images contain 638 palms. The test persons were told to mark the tree crown centers of all palms they can find. The network was trained according to the scheme presented in the paragraph 'network training' and of course not on the test images. The ground truth labels were created by the author of this thesis, who has by hand marked more than 15 000 palms in total.

## 6.2.4. Results

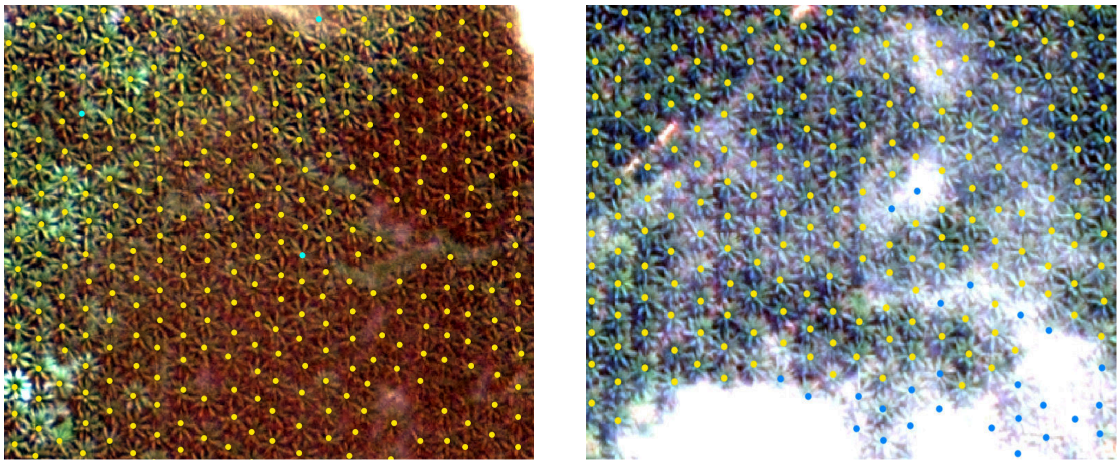
### Classification Accuracy on the Jambi Dataset

"Table 6.3 gives the results for U-Nets A and B, and the classifier approach [47], trained on the Jambi dataset.

| | Accuracy | Precision | Recall | $F_1$-Score |
|---|---|---|---|---|
| U-Net A | **88.6%** | **94.4%** | 93.5% | **93.9%** |
| U-Net B | 87.9% | 93.2% | **94.0%** | 93.6% |
| AlexNet [47] | 75.0% | 88.8% | 83.0% | 85.8% |

**Table 6.3.:** Performance metrics of our model in comparison with state of the art method. The numbers have been obtained by first averaging over the k-fold runs and then averaging over the last 50 epochs, after the metrics have converged. Highest numbers are highlighted. The training history is given in the Appendix A.

The results show that our model outperforms AlexNet. U-Net A scores highest with an accuracy of 88.6%, closely followed by U-Net B, which scores 87.9%. The AlexNet model used in [47] reaches 75%. Detailed curves for the losses and metrics during the training can be found in the Appendix A. Figure 6.13 sheds some light on the performance of U-Net A under difficult conditions:" [22]



**(a)** Palms in the shadow of a cloud. **(b)** Palms partly under clouds.

**Figure 6.13.:** "The network still detects palms under difficult conditions. True positives are marked yellow and false negatives blue." [22]

### Benchmark and Large Scale Performance

"Table 6.4 lists the results of the speed benchmark, which was done following the procedure described in the paragraph 'Speed Benchmark'.

| Network | Input Size [px] | Step [px] | Time [s] | Throughput [ha/s] |
|---|---|---|---|---|
| AlexNet [47] | 26 × 26 | 5 | 17.7 | 22.6 |
| U-Net A | 112 × 112 | 80 | 3.3 | 121.2 |
| U-Net B | 112 × 112 | 80 | 1.8 | 222.2 |

**Table 6.4.:** Results of the speed benchmark. The standard deviations for the timings were 0.08 s for AlexNet, 0.05 s for U-Net A, and 0.03 s for U-Net B.

U-Net architecture B is fastest, reaching a throughput of 222.2 ha/s, followed by U-Net A with 121.2 ha/s. The AlexNet [47] reaches a throughput of 22.6 ha/s. Therefore U-Net B is one order of magnitude faster than AlexNet. The speed of the U-Net architecture can be enhanced even more by feeding it with larger image patches.When feeding image patches with a size of 512 by 512 pixels to the network, U-Net A reaches a throughput of 181.8 ha/s and U-Net B reaches 235.3 ha/s. Increasing the input window size did not affect the accuracy: U-Net A detects 96.1% and U-Net B 92.8% of the 1700 palms the test set created for this task.

As we have shown the performance in terms of quality and speed, we unleashed the full potential of the U-Net and applied it to the entire dataset. In order to do so, we applied U-Net A as a moving window, as shown in Figure 6.10. Inference on the whole Jambi dataset of 348 km$^2$ takes 18 min and yields a number of approximately 2.1 million palms. This is slower than one would expect from the numbers in Table 6.4 due to the non-rectangular shape of our dataset and input/output operations. Figures 6.14 and 6.15 show the results. We can observe that the entire southern part of the study area (at the bottom of Figure 6.14) is covered with large monoculture plantations. In combination with the structured plantation pattern, this indicates a corporate land use. On the other hand, plantations in the northern part are smaller and scattered, therefore most likely owned by smallholders." [22]

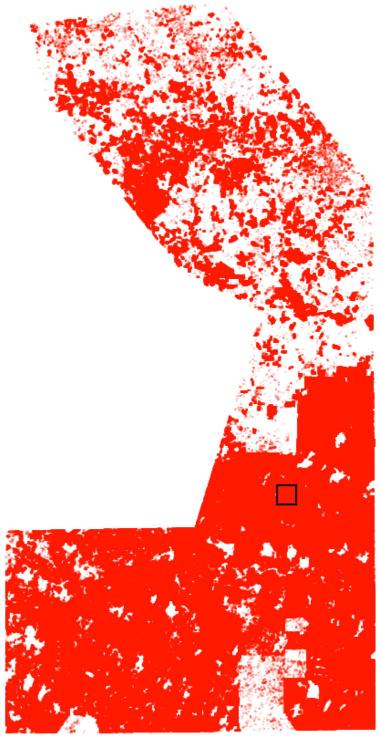


Figure 6.14.: "The palm coverage of the Jambi study site. The image displays 2.1 million palms, each one resolved individually." [22]

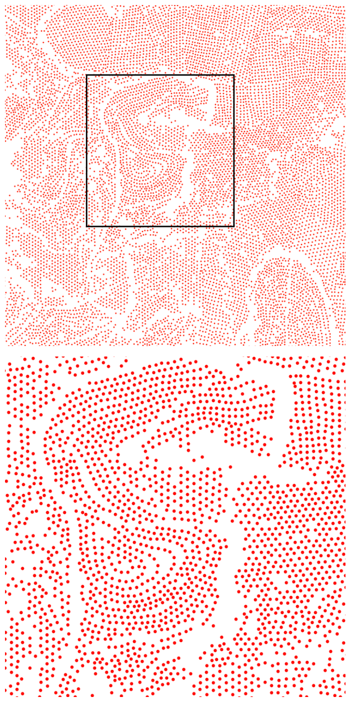


Figure 6.15.: "Zoom into the study site. The magnification increases from top to bottom. With higher magnification, the planting patterns become clear." [22]

### Transfer to the Bengaluru Dataset

"We transferred both pre-trained U-Net models to the Bengaluru dataset, as described in the paragraph "Transferability of Pre-Trained Network". First, we re-trained the batch normalization layers, followed by a fine tuning of the whole network. Training only the batch normalization layers boosted the accuracy from 12% to 83% for architecture A and from 20% to 78% for architecture B. After the subsequent fine-tuning of all layers, architecture A reaches an accuracy of 84.4% and architecture B reaches 91.8%. The entire re-training procedure took only about eight minutes. The final results are summarized in Table 6.5.

| | Accuracy | Precision | Recall | $F_1$-Score |
|---|---|---|---|---|
| U-Net A BN only | 83.3% | 86.9% | 95.2% | 90.9% |
| U-Net B BN only | 77.8% | 83.1% | 92.4% | 87.5% |
| U-Net A | 84.4% | 88.6% | 94.7% | 91.6% |
| U-Net B | **91.8%** | **95.0%** | **96.4%** | **95.7%** |

Table 6.5.: Performance metrics for the two architectures after training the batch normalization layers only, and for the full transfer. Here the metrics have been derived from a set of test images and not from k-fold validation. Highest numbers are highlighted.

The result of the fine-tuning of U-Net architectures A and B showed that U-Net B performed best on the WorldView-3 Bengaluru dataset. To assess the validity of the approach, we applied it to the whole transect; the resulting map shows approximately 106 000 palm trees. In the urban area, coconut palms are found scattered alongside roads, in parks or gardens, as already found by visual inspection. Further north, in the transition and rural region, palms mainly grow in plantations with only few solitary palm trees. The plantations are much smaller than those in

Jambi and the planting distance is larger. Figure 6.16 shows two examples from the Bengaluru study site:" [22]

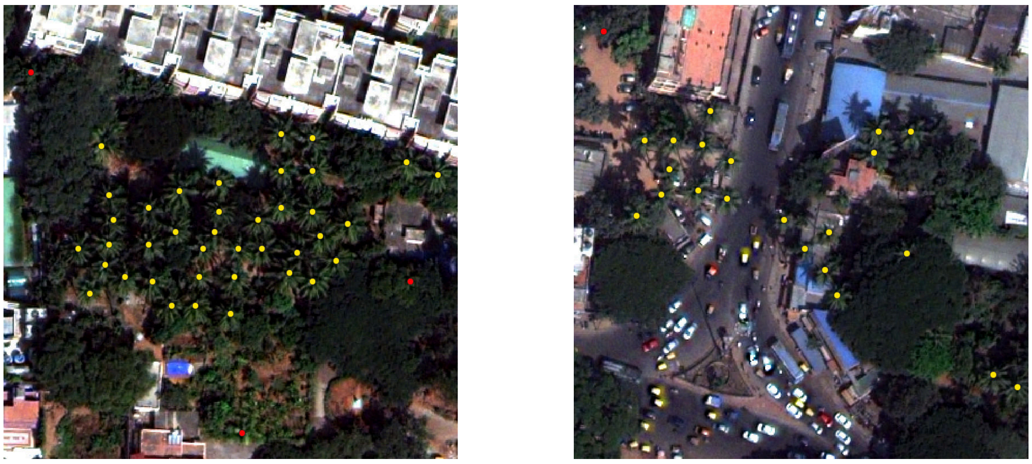


**Figure 6.16.:** "Samples from the Bengaluru study area after transferring U-Net B. True positives are marked yellow and false positives red." [22]

### Comparison of Neural Network and Humans

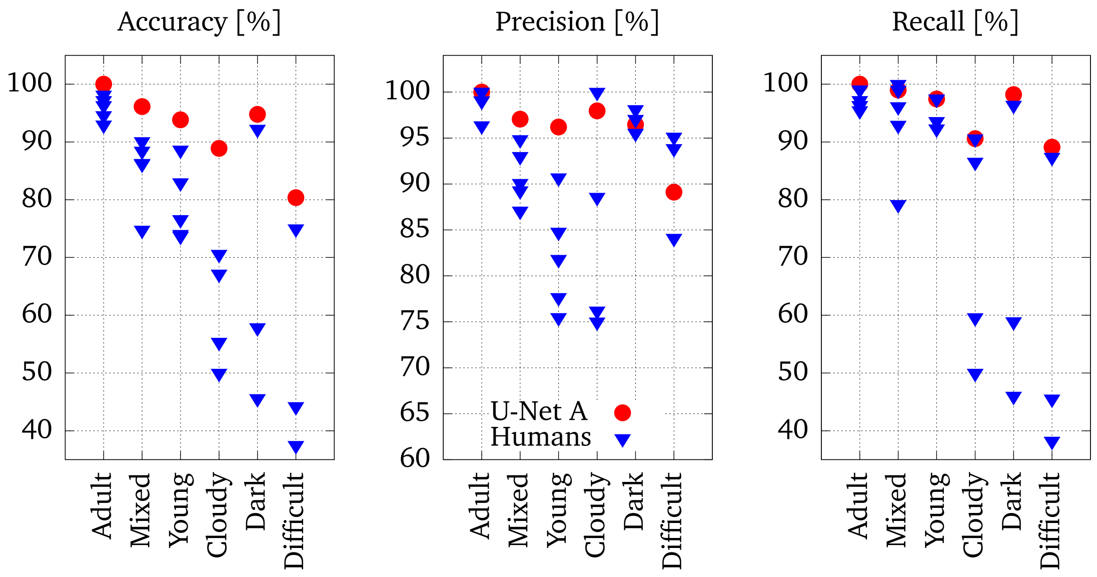
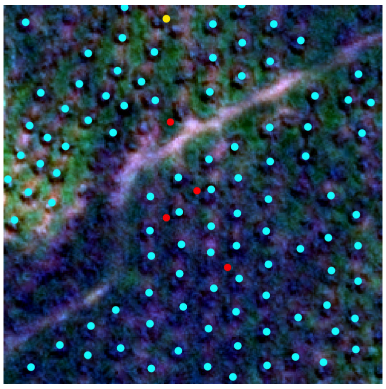


**Figure 6.17.:** Comparison of the human performance (blue) versus U-Net A (red). Please note the different y-axis scaling for the middle graph.

Figure 6.17 depicts the results of the comparison human - neural network. Not every test person completed every annotation and with such a low number of data points we have to resort to a qualitative description of the results. It shows that with 80-100% the U-Net achieves higher accuracy than all humans on all datasets. One can notice strong variations in the human results, except for the image depicting adult palms (which was easiest to annotate). In general, the performance on the images tagged "cloudy", "dark" and "difficult" is worse. On those images, the recall is low for the human participants, which indicates that they did not find all palms. For the images tagged "mixed", "young" and "cloudy" the precision of the human labelers is lower, compared to the other images; they produced false positives. Especially on the image depicting young palms this can happen, as there are spots on the ground, where a palm *will* be planted, but is not visible yet. The test images can be viewed in the appendix (Fig. A.10).

### Failure Cases

“The visual inspection reveals different cases in which the network fails, equally applicable to both architectures. Figure 6.18 presents examples for the study area of Jambi and Figure 6.19 refers to the Bengaluru study site.” [22]


**(a)** Undetected young palms in shadows

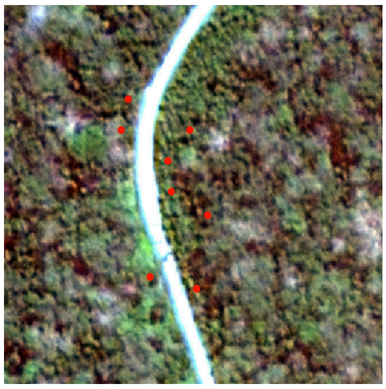
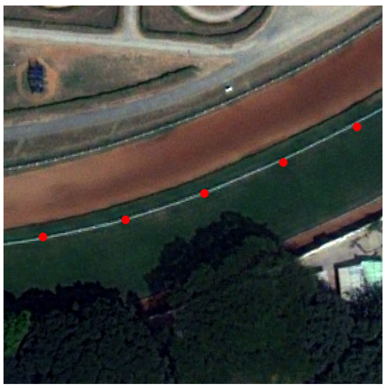

**(b)** Falsely positive marked shrubs

**Figure 6.18.:** “Failure cases in the Jambi study area: True positives are marked in yellow, false positives in red, and false negatives in blue. Young palms in shadows are rare in the dataset, and therefore hardly detected. Other failure cases, such as in (b), are challenging to find, as they rarely occur.” [22]

“Figure 6.18a reveals that the network has problems finding young palms in shadows, which is the most common failure case in the Jambi dataset. Shadow is a common factor that also deteriorates the performance in other experiments. Under low light conditions, the network generates false positives in areas with forest and omits some of the adult trees. Nevertheless, visual inspection shows that the predictions are still quite robust under the influence of shadows (see Figure 6.13). False positives, such as shown in Figure 6.18b, are rare. They mostly occur in forest areas, where the vegetation randomly resembles a palm, or near bright-dark transitions involving green color. Clouds are tolerable to a certain degree, as long as the ground is still visible.” [22]


**(a)** Falsely positive marked bright line

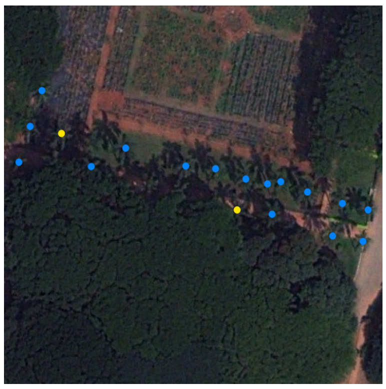

**(b)** False negatives

**Figure 6.19.:** “Failure cases in the study area of Bengaluru. True positives are marked yellow, false positives red, and false negatives blue. In (a) the convolutional filters respond to a bright line. In (b) the network fails to detect some palm trees probably because of dark green grass in the background.” [22]

“Figure 6.19a depicts false positives next to a bright-dark transition in the Bengaluru area, which shows that this error is not restricted to one dataset. In certain areas with low contrast, such as in Figure 6.19b, the network also fails to detect some palms correctly. Another failure case in the Bengaluru dataset is that young mango trees are also labeled, as they resemble the round, 'blob-like' shape of young palms. Apart from that, the network performance depends on the location: it is worse for mixed terrain with forest or urban areas and best on plantations, where it labels almost every object correctly." [22]

### 6.2.5. Discussion

"In this [thesis] we presented a new method for large area oil palm detection on very high resolution satellite images, which is based on the U-Net. Overall, we reach $F_1$-Scores well comparable to the ones reported earlier [47, 48]. On our dataset, the U-Net outperforms previous approaches by 10-13 percentage points in terms of accuracy and by 6-8 on the $F_1$-Score. We hypothesize that this improvement has three reasons: First, the U-Net sees the entirety of the training images during the training, while the classifiers in [47, 48] only see the cutouts around the palm- or non-palms positions. Therefore, the U-Net effectively sees more training data from the same image source. Second, the U-Net is able to take larger contexts into account during segmentation. It 'sees' not only one palm, but several palms. This way, it can for example recognize plantation patterns and adjust the segmentation accordingly. Third, the U-Net is a more powerful network than the AlexNet or LeNet used in [47, 48, 40] since its internal structure suits the given task better, as it directly highlights the patterns it has been trained for and returns a probability map. Furthermore, U-Net B has 260 000 parameters in comparison to the 790 000 of our AlexNet implementation, so it is also smaller.

An important contribution of our approach is the computational efficiency, which comes from predicting entire segmentation maps instead of single probabilities, as well as from better utilizing the parallel computing capacity of graphics cards. Our method is able to reduce the computation time by an order of magnitude compared to the existing method, which enabled us to scan areas of several hundred square km within a reasonable amount of time and processing effort - without being restricted to pre-delineated plantations. As our Jambi dataset was collected in a large area, we were able to prove that the U-Net delivers high accuracy even on large scale. Furthermore, the U-Net is well-scalable and able to leverage the performance benefits from newer hardware generations, which would allow to increase the input window size even further.

We showed that it is possible to transfer a pre-trained model from one dataset to another with a reasonable amount of new training data. From our two models, the simpler one (U-Net B) had a higher performance on the new dataset (see Table 6.5). This might be due to the fact that it has less parameters and is therefore less prone to overfitting (see Figure A.7). The high accuracy after training the batch normalization layers proves that they play a key role in transferring the model. Atmospheric correction has not been used, as we wanted to assess how well the models generalize to new, raw data. Further studies have to be conducted in order to find out which role atmospheric correction can play in the process of palm detection." [22]

The comparison between the performance of humans and U-Net A showed that the network reaches a well comparable, or even better, accuracy. At the same time, the network is much faster. Labeling by hand took the test persons about 30 minutes, whereas the network prediction took 2-3 seconds. However, it has to be mentioned that these results are biased in favor of the network, as it has adapted to the author's labeling style during training, as it was trained with data generated by the author. In order to assess the significance of the results, a higher number of human participants would be required.

"Even though the accuracy of the new method is high, there were some failure cases. The results showed that the models fail when the signal to noise ratio becomes too low, which is the case in dark shadows or at the edges of clouds. In contrast, we have observed that lighter shadows or clouds had only little influence on the results. Further research, with designated test data for this case, has to be conducted in order to verify the accuracy below clouds or in shadows. The most common failure, young palms in shadows, has a minor effect on the overall accuracy, as they are rare in the dataset. This failure case could probably be ameliorated by acquiring more training data for this specific class. With respect to the high image resolution, we are confident that the datasets we generated are of high quality. Nevertheless, labeling errors can always occur and impair both, network training and accuracy assessment.

In spite of the good results obtained with the proposed approach, there is room for improvements and further work. For instance, it would be interesting to explore the combination of the U-Net and the networks in [47, 48], by applying them to the palm positions predicted by the U-Net. In this manner, it would be possible to reduce the number of false positives while keeping the computational efficiency. The combination of U-Net and classifier could also be used to provide an alternative to existing methods for the classification of diseased trees [72, 69]." [22]

The keyword "classification" brings us to our next topic: detecting the species of trees in aerial imagery.

## 6.3. Tree Species Classification

In this section we detected the species or species group of trees in aerial images from the region around Gartow, Lower Saxony. The tree positions were known from a Lidar mapping of the area and at the present state not inferred by our algorithm. The Lidar data was not used for the classification, as the ultimate goal was to arrive at a system that is able to function independently. As we saw in Section 2.3, conventional methods rely on multispectral data with eight or more bands and are restricted to five classes at maximum. The novelty of the following work is that we differentiated more species groups with less imaging bands. Furthermore, we detected heavily damaged trees, as the oak procession moth has infested oaks in the area.

### 6.3.1. Labeled Data

In this section we used the Gartow dataset, presented in Section 5.3. The aerial images have a resolution of 5 cm and four bands (RGB, NIR) with a precision of 8 bit. The labeled data comprises the positions and the species of 7323 trees. Table 6.6 lists each species along with its frequency of occurrence:

| Species group | Translation | Abbr. | Count |
|---|---|---|---:|
| Birch | Birke | Bi | 799 |
| Beech | Buche | Be | 504 |
| Douglas fir | Douglasie | Dg | 509 |
| Oak | Eiche | Oa | 334 |
| Damaged oak | Geschädigte Eiche | Da | 355 |
| Alder | Erle | Al | 730 |
| Spruce | Fichte | Sp | 506 |
| Pine | Kiefer | Pi | 2571 |
| Larch | Lärche | La | 509 |
| Background | Hintergrund | Bg | 506 |
| Total | | | 7323 |

**Table 6.6.:** Tree species groups and their respective quantity in the dataset.

Of the tree positions and labels, approximately 5000 were obtained by highly precise differential GPS measurements in the field. The number of samples in this initial dataset was too low to train a deep neural network and the classes within it were heavily unbalanced. As collecting more samples in the field is a (time) costly process, the remaining samples were collected in the aerial images. This labeling process was carried out by experienced researchers who are working in the field for several years and have been on-site. Around each of the 7323 positions, a window of $100 \times 100$ pixels ($5\,\text{m} \times 5\,\text{m}$) was cut out; this is the final dataset. That window size is appropriate, because most trees crowns fit well into it, without capturing too much of other trees.

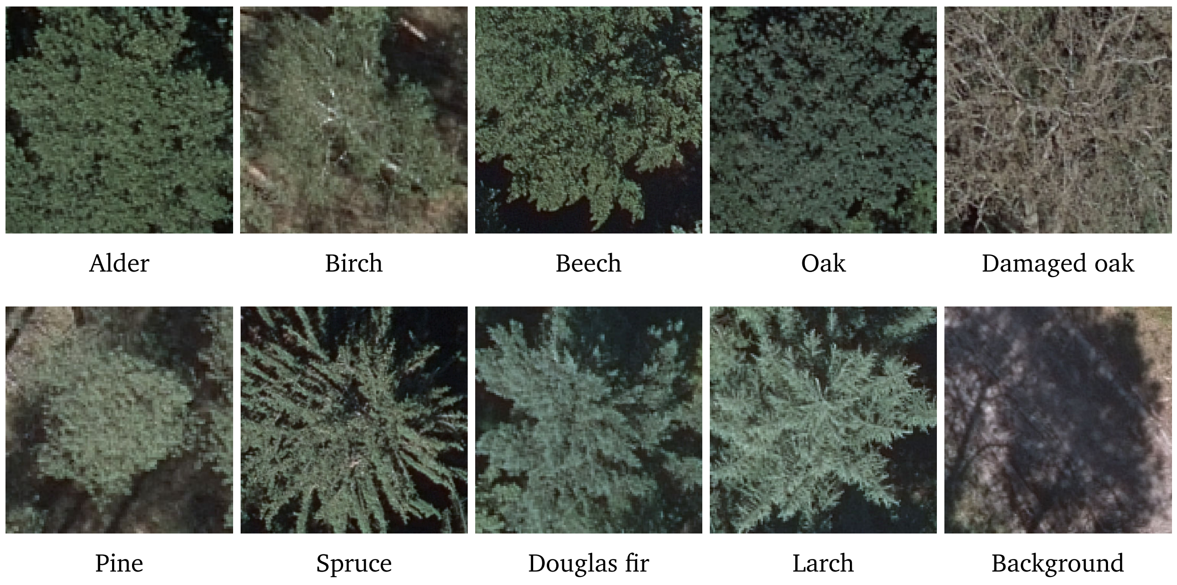


**Figure 6.20.:** Cutouts of the different tree species in the labeled data. Each image displays a 5×5 m window.

## 6.3.2. Experimental Setup

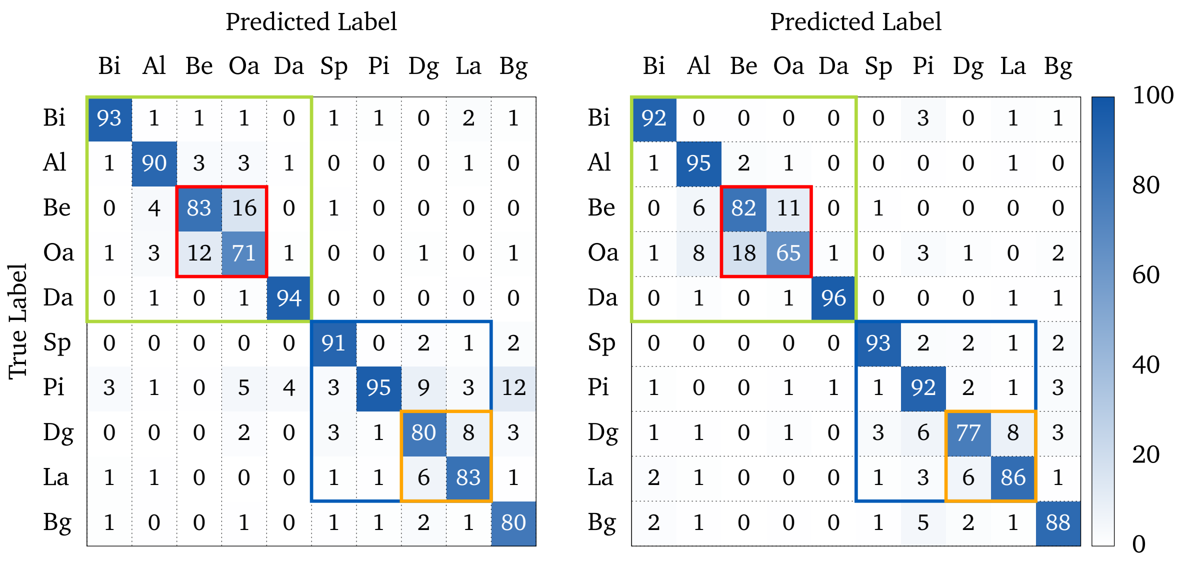
In order to perform the tree species classification, we used the ResNet18 model described in Section 4.2. We modified the standard ResNet in two ways: First, we adjusted the first convolutional layer, so that it now accepts input images with five bands (RGB, NIR, NDVI in this order), instead of the regular RGB images. Second, the standard ResNet18 supports 1000 classes; we reduced this to the required amount (8-10). As ResNet is a common model, other researchers have trained it on large datasets, such as ImageNet, and published the resulting weights. We initialized the weights of all layers that we have not changed with the weights from [81]. In preceding evaluations we found that this procedure increases the model's accuracy by about 1.8 percentage points.

In all following evaluations the network was trained for 40 epochs with 50 gradient updates per epoch, resulting in 2000 gradient steps in total. For the first 10 epochs the learning rate was set to $10^{-4}$, then lowered to $5 \cdot 10^{-5}$ until epoch 30, and for the last 10 steps it was set to $10^{-5}$. We used a batch size of 50. As the dataset is very imbalanced (Pine is the predominant species in the area), we have set the class weights according to equation (3.10) to counterbalance this. In the same way as for the other experiments we use the Adam optimizer, but here in conjunction with the regular categorical cross-entropy loss (eq. (3.15)). The datasets were again augmented by randomly rotating the images by 90 degrees, random flipping and random channel-wise color shifts.

In a first experiment we evaluated the network performance for all ten classes. For this, we performed a 10-fold cross-validation with a training - test split of 70-30%. This test revealed which classes are separable or not. In a subsequent iteration we merge two non-separable classes and re-train the network.

### 6.3.3. Results

The loss and accuracy curves for the training runs in this section can be found in the appendix (Sec. A.3). Figure 6.21 depicts two differently normalized versions of the confusion matrix obtained during the 10 cross-validation runs with all classes. For each run, separate confusion matrices were calculated and these were averaged to obtain the following tables:



| True \ Predicted | Bi | Al | Be | Oa | Da | Sp | Pi | Dg | La | Bg |
|---|---|---|---|---|---|---|---|---|---|---|
| Bi | 93 | 1 | 1 | 1 | 0 | 1 | 1 | 0 | 2 | 1 |
| Al | 1 | 90 | 3 | 3 | 1 | 0 | 0 | 0 | 1 | 0 |
| Be | 0 | 4 | 83 | 16 | 0 | 1 | 0 | 0 | 0 | 0 |
| Oa | 1 | 3 | 12 | 71 | 1 | 0 | 0 | 1 | 0 | 1 |
| Da | 0 | 1 | 0 | 1 | 94 | 0 | 0 | 0 | 0 | 0 |
| Sp | 0 | 0 | 0 | 0 | 0 | 91 | 0 | 2 | 1 | 2 |
| Pi | 3 | 1 | 0 | 5 | 4 | 3 | 95 | 9 | 3 | 12 |
| Dg | 0 | 0 | 0 | 2 | 0 | 3 | 1 | 80 | 8 | 3 |
| La | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 6 | 83 | 1 |
| Bg | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 2 | 1 | 80 |

**(a)** Precision [%]

| True \ Predicted | Bi | Al | Be | Oa | Da | Sp | Pi | Dg | La | Bg |
|---|---|---|---|---|---|---|---|---|---|---|
| Bi | 92 | 0 | 0 | 0 | 0 | 0 | 3 | 0 | 1 | 1 |
| Al | 1 | 95 | 2 | 1 | 0 | 0 | 0 | 0 | 1 | 0 |
| Be | 0 | 6 | 82 | 11 | 0 | 1 | 0 | 0 | 0 | 0 |
| Oa | 1 | 8 | 18 | 65 | 1 | 0 | 3 | 1 | 0 | 2 |
| Da | 0 | 1 | 0 | 1 | 96 | 0 | 0 | 0 | 1 | 1 |
| Sp | 0 | 0 | 0 | 0 | 0 | 93 | 2 | 2 | 1 | 2 |
| Pi | 1 | 0 | 0 | 1 | 1 | 1 | 92 | 2 | 1 | 3 |
| Dg | 1 | 1 | 0 | 1 | 0 | 3 | 6 | 77 | 8 | 3 |
| La | 2 | 1 | 0 | 0 | 0 | 1 | 3 | 6 | 86 | 1 |
| Bg | 2 | 1 | 0 | 0 | 0 | 1 | 5 | 2 | 1 | 88 |

**(b)** Recall [%]

**Figure 6.21.:** The confusion matrix for all ten classes, normalized column-wise (a) and row-wise (b). As a result, the diagonal contains the precision (a) and the recall (b) metric. The confusion matrices of all ten cross-validation runs were normalized and then averaged to obtain the above result. The green box frames deciduous species and the blue one coniferous. Off-diagonal values in the red and orange boxes are relatively high. (Bi: Birch, Be:Beech, Dg: Douglas Fir, Oa: Oak, Da: Damaged oak, Al: Alder, Sp: Spruce, Pi: Pine, La: Larch, Bg: Background)
**Reading example (a):** From all predictions the network made for the class “oak”, 71% were correct and 16% were wrongly classified as beech. The remainder was falsely classified into other classes.
**Reading example (b):** Of all douglas firs in the respective test set, 77% were classified correctly. Six percent were falsely classified as pines, 8% as larches and the remainder into other classes.

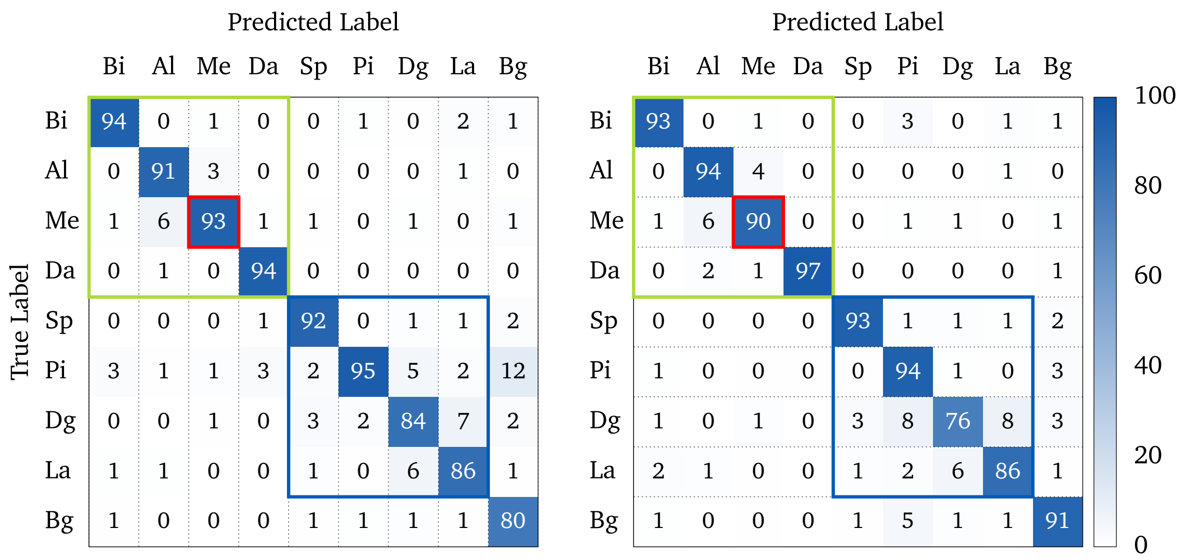
The entries framed green correspond to deciduous species and the entries framed blue to coniferous trees. A first observation is that between these two groups, little confusion appears. Nevertheless, there is confusion between species within one group. The entries highlighted red indicate a high confusion rate between oak and beech. Furthermore, 6-8% of the beeches and oaks were classified as alder. The orange box frames another, but less severe, case of confusion which occurs between douglas fir and larch. According to equation (3.11) we can calculate the $F_1$-Score from the ten confusion matrices, which averages to 88.9% ± 1.6%. Table 6.7 lists the resulting performance metrics along with their standard deviation, which was obtained from the ten cross-validation runs.

| | Precision | Recall | $F_1$-Score |
|---|---|---|---|
| Birch | 93.1 ± 2.2 | 92.5 ± 1.8 | 92.8 ± 1.0 |
| Alder | 89.7 ± 1.7 | 94.6 ± 1.8 | 92.0 ± 1.2 |
| Beech | 83.3 ± 4.5 | 82.3 ± 5.9 | 82.6 ± 2.9 |
| Oak | 70.8 ± 6.2 | 65.1 ± 7.9 | 67.2 ± 3.5 |
| D. Oak | 94.0 ± 2.6 | 95.7 ± 1.2 | 94.8 ± 1.8 |
| Spruce | 90.9 ± 2.6 | 92.6 ± 2.9 | 91.7 ± 1.3 |
| Pine | 95.1 ± 0.9 | 92.0 ± 1.7 | 93.5 ± 0.7 |
| Douglas F. | 79.8 ± 4.0 | 76.6 ± 4.2 | 78.0 ± 1.9 |
| Larch | 83.3 ± 6.2 | 86.3 ± 4.1 | 84.5 ± 2.4 |
| Backgr. | 79.6 ± 4.8 | 88.1 ± 3.8 | 83.4 ± 1.7 |
| Weighted avg. | 89.1 ± 3.2 | 88.9 ± 3.3 | 88.9 ± 1.6 |

**Table 6.7.:** The class-wise scores along with their standard deviations, which were derived from the 10-fold cross-validation. The bottom row gives the class-averaged values, where each class has been weighted according to its frequency. The standard deviations of the average values were calculated using the Gauß rule for uncertainty propagation.

With only 67.2% the class “oak” has the lowest $F_1$-Score, and at the same time, in its damaged form, the highest (94.8%). For beeches, douglas firs, larches and the background the $F_1$-Score is between 78% and 84.5%. Birch, alder, spruce and pine are classified with $F_1$-Scores between 91.7% and 93.5%.

On the basis of the low $F_1$-Score, and high confusion between the classes “oak” and “beech” we decided to merge these into one class (“Me” for merged) with the aim of improving the overall $F_1$-Score. Furthermore, these species are difficult to identify on aerial images by eye, which makes it unlikely that the amount of training data can be increased without more ground-validated forest inventory data. Douglas firs and larches are confused as well, but these classes can be differentiated with imagery taken in winter, as larches shed their needles and douglas firs do not. Therefore, these species will be kept separate. We trained the network again, with beech and oak merged; Figure 6.22 gives the resulting confusion matrix, again averaged over 10 cross-validation runs.



| | Bi | Al | Me | Da | Sp | Pi | Dg | La | Bg |
|---|---|---|---|---|---|---|---|---|---|
| Bi | 94 | 0 | 1 | 0 | 0 | 1 | 0 | 2 | 1 |
| Al | 0 | 91 | 3 | 0 | 0 | 0 | 0 | 1 | 0 |
| Me | 1 | 6 | 93 | 1 | 1 | 0 | 1 | 0 | 1 |
| Da | 0 | 1 | 0 | 94 | 0 | 0 | 0 | 0 | 0 |
| Sp | 0 | 0 | 0 | 1 | 92 | 0 | 1 | 1 | 2 |
| Pi | 3 | 1 | 1 | 3 | 2 | 95 | 5 | 2 | 12 |
| Dg | 0 | 0 | 1 | 0 | 3 | 2 | 84 | 7 | 2 |
| La | 1 | 1 | 0 | 0 | 1 | 0 | 6 | 86 | 1 |
| Bg | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 80 |

**(a)** Precision [%]

| | Bi | Al | Me | Da | Sp | Pi | Dg | La | Bg |
|---|---|---|---|---|---|---|---|---|---|
| Bi | 93 | 0 | 1 | 0 | 0 | 3 | 0 | 1 | 1 |
| Al | 0 | 94 | 4 | 0 | 0 | 0 | 0 | 1 | 0 |
| Me | 1 | 6 | 90 | 0 | 0 | 1 | 1 | 0 | 1 |
| Da | 0 | 2 | 1 | 97 | 0 | 0 | 0 | 0 | 1 |
| Sp | 0 | 0 | 0 | 0 | 93 | 1 | 1 | 1 | 2 |
| Pi | 1 | 0 | 0 | 0 | 0 | 94 | 1 | 0 | 3 |
| Dg | 1 | 0 | 1 | 0 | 3 | 8 | 76 | 8 | 3 |
| La | 2 | 1 | 0 | 0 | 1 | 2 | 6 | 86 | 1 |
| Bg | 1 | 0 | 0 | 0 | 1 | 5 | 1 | 1 | 91 |

**(b)** Recall [%]

**Figure 6.22.:** Confusion matrix with beech and oak merged into one class (Me). The confusion matrices of each cross-validation run were normalized row or column-wise (a/b) and then averaged. Hence, the diagonal elements correspond to the precision (a) and the recall (b). The green box frames deciduous species, the blue one coniferous. The merged class is labeled red.

| | Precision | Recall | $F_1$-Score |
|---|---|---|---|
| Birch | 93.6 ± 1.2 | 93.0 ± 1.3 | 93.3 ± 1.0 |
| Alder | 90.6 ± 2.0 | 94.4 ± 1.6 | 92.4 ± 1.2 |
| Merged | 92.9 ± 1.2 | 89.5 ± 1.5 | 91.2 ± 1.1 |
| D. Oak | 94.0 ± 4.8 | 96.7 ± 0.8 | 95.3 ± 2.5 |
| Spruce | 91.6 ± 3.8 | 93.4 ± 1.9 | 92.4 ± 2.2 |
| Pine | 95.3 ± 1.4 | 93.6 ± 1.4 | 94.4 ± 0.6 |
| Douglas F. | 84.4 ± 4.7 | 76.4 ± 4.3 | 80.0 ± 2.6 |
| Larch | 85.6 ± 5.1 | 86.5 ± 3.2 | 85.8 ± 1.9 |
| Backgr. | 79.8 ± 2.3 | 90.7 ± 3.9 | 84.8 ± 1.9 |
| Weighted avg. | 91.6 ± 2.7 | 91.4 ± 2.2 | 91.4 ± 1.4 |

**Table 6.8.:** The updated class-wise scores with beech and oak merged into one class (Me).

Table 6.8 lists the results for the cross-validation runs with beeches and oaks merged into one class. The newly created class reaches an $F_1$-Score of 91.2%, compared to 82.6% (beech) and 67.2% (oak) before. The $F_1$-Score of the other classes benefited between 0.4 and 2 percentage points. In total, merging the two classes results in an overall $F_1$-Score of 91.4% ± 1.4%, compared to 88.9% ± 1.6%.

Finally, we applied the ResNet18 to all tree positions which were previously inferred from the Lidar data. We employed the full classifier with 10 classes to make eventually appearing stands of beech and oak visible. Figure 6.23 displays the resulting tree species map.

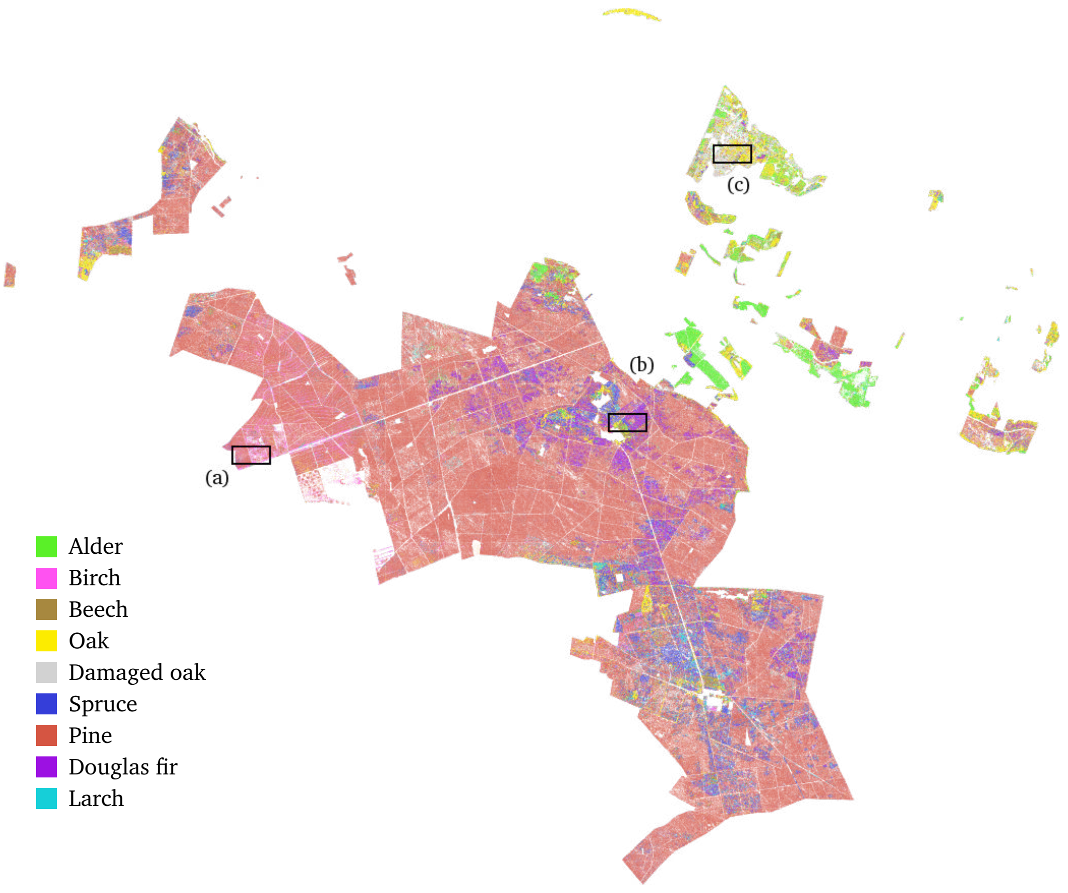
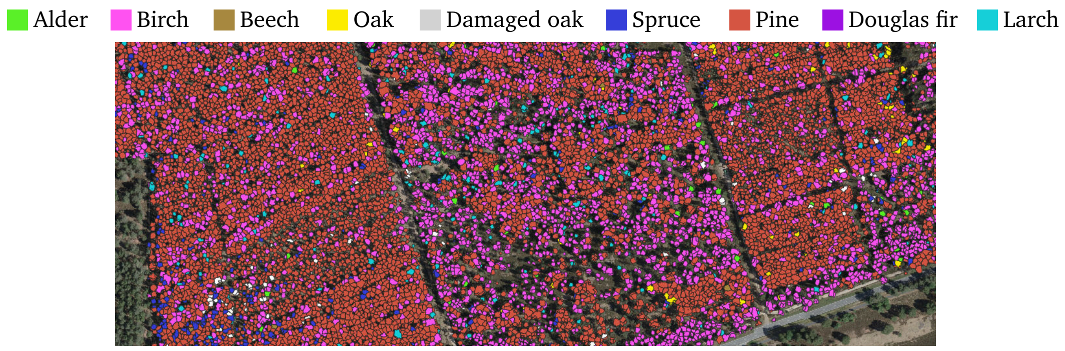


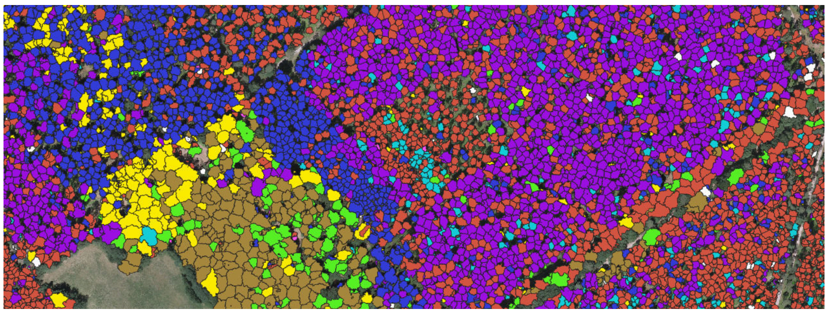
**Figure 6.23.:** The classification of the entire gartow study area into nine classes. The boxes frame areas which are presented enlarged in Figure 6.24. North is up.

With a predicted share of 75%, pines are the predominant species in the study site. Spruces take a share of 5%, oaks and douglas firs 4%. The other species account for the remaining 12% with each one sharing 3% or less.

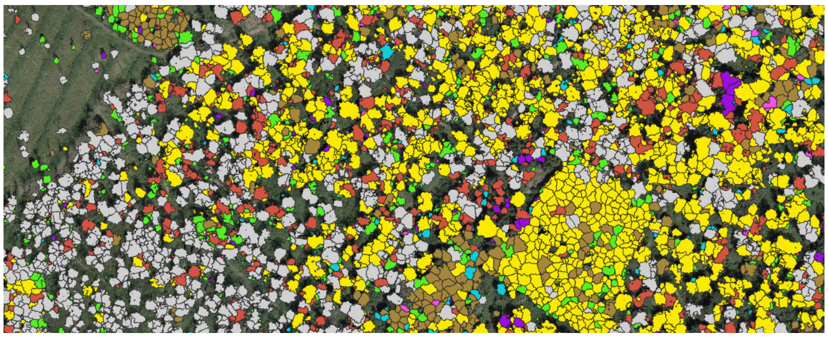
The large, continuous forest area is mainly comprised of pines, interspersed with douglas firs and larches. Deciduous trees are found in smaller groups in the north-eastern part of the study area. Figure 6.24 depicts the three enlarged example areas marked in Figure 6.23. Please note that the classification uncertainties from Table 6.7 apply, meaning that there can be confusion between beeches and oaks, as well as between douglas firs and larches.



**(a)** Birches along with pines.



**(b)** A mixed region with different species.



**(c)** Deciduous trees with many damaged oaks among them.

**Figure 6.24.:** The three enlarged example areas from Figure 6.23.

### 6.3.4. Discussion

The results obtained indicate that it is very well possible to classify tree species in high resolution aerial images using a deep convolutional neural network. Depending on whether beeches and oaks are merged into one class, the ResNet18 is capable of differentiating between 8 or 9 classes (excluding background) with an averaged $F_1$-Score of 91.4% and 88.9%, respectively. To put these results into context, we compared them to other work [59], which used aerial imagery with the same resolution. It turns out, that our approach yields a comparable $F_1$-Score for the four classes present in [59], but at the same time is capable of distinguishing between more classes. Furthermore, our dataset included only four imaging bands, instead of 33, and we did not use the Lidar height information in the classification.

During our experiments we saw that beeches and oaks are confused most. This is in line with the visual inspection of the aerial imagery: differentiating these classes is difficult for humans as well. The confusion between larches and douglas firs is also relatively high. These species are easier to distinguish by eye and it has to be investigated further, why the neural network confuses them. However, larches and douglas firs can be differentiated by using imagery from different seasons, as larches shed their needles in winter and firs do not. This was also the reason why those two classes were not merged into one.

The dataset at hand was imbalanced, with 350-500 instances for most classes, but with approximately 2500 pines. As a result, the measured metrics fluctuated more for the classes with less instances (beech and oak) and least for pines. Expanding the training dataset to incorporate more of the now underrepresented classes would ameliorate this, improving the performance in general. Apart from the uncertainty introduced by the different cross-validation splits, labeling errors might diminish the network performance as well. This is likely a minor issue for pines, as most of the on-ground validated trees were pines. In contrast, most of the beeches and oaks were collected in the aerial imagery, which might have introduced errors due to their poor distinctness. Hence, this *could* be the root cause for the high confusion rates between the two, and requires further examination of the dataset itself.

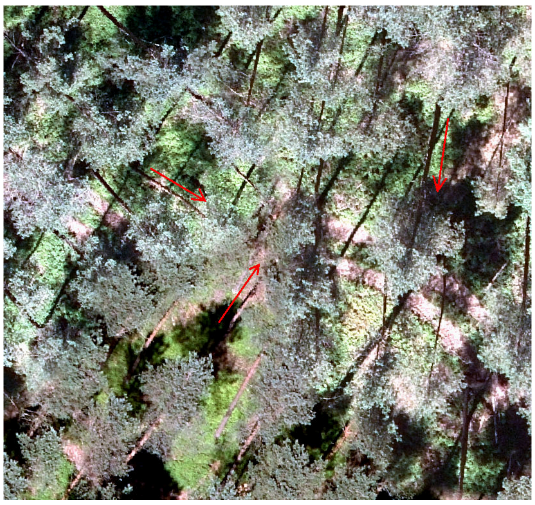


**Figure 6.25.:** Image alignment errors. The stem directions are marked red.

Another issue occurring in aerial imagery is the alignment error of image tiles. Figure 6.25 depicts a severe case. The tree stems and their shadows in different portions of the image are oriented in different directions, as they were photographed from different positions at different times of day. This makes the classification unreliable at certain points in the imagery, but the exact effect is hard to quantify. Furthermore, eventually occurring misalignment of the Lidar and optical data introduces another source of error which cannot be quantified easily.

There are several options how the presented approach can be improved and the following list is not exhaustive. First of all, one could capture and use more imaging bands, which could possibly allow to discriminate between more classes. However, this would contradict the aim of simplifying the entire classification procedure and the aim of lowering the costs. For the same reason, we have up to now not included the Lidar height information into the training procedure, as one of the objectives was to cope without this information. Anyhow, as long as this information is collected, it could be included as additional feature during training, but this is up to further studies. By the watershed transform, which is applied to the height profile, the tree crown outlines are known during training and test time. A further increase in accuracy could be achieved by cutting out nothing but these crowns, eliminating pixels not belonging to the tree currently presented from the network's field of view.

To conclude, we would like to point out that the ResNet performed best for the class “damaged oak”. During the application to the entire dataset of approximately 1.8 million trees, it has precisely identified regions infested with the oak procession moth. Therefore this neural network has proved to be a valuable tool for forest monitoring. One can easily imagine other applications, such as detecting damaged trees near railways or power grid lines, thereby contributing to their safety. Consequently, further research may lead to relevant results.

# 7. Conclusion and Outlook

In the ablation study we saw, that four of the eight bands available from *WorldView* are sufficient to perform tree cover segmentation. In general, this finding can help to save disk space and computation time. The benchmark of the three networks, U-Net A, B and U-ResNet showed that all of them reach intersection over union values above 88% in a cross-validation test. U-ResNet scored only 2.4 percentage points higher than U-Net B, at the expense of having 50 times more parameters. This indicates that tree cover can as well be segmented by shallower, and therefore faster, neural networks.

In the second part of the experiments, we demonstrated that it is possible to infer the positions of palm trees from probability maps generated by a U-Net. This approach proved to be one order of magnitude faster than the current state of the art, while maintaining a comparable accuracy of 88%. We showed, that it is possible to transfer a network which was trained in one region of the earth to another one with a small amount of new training data. As expected, a smaller network with less parameters generalized better to the new data, as it is less prone to overfitting. The presented approach is now fast enough to be applied to entire countries and has a high practical relevance for plantation owners, governments, the palm oil using industry and NGOs that want to monitor illegal activities.

The third computer experiment proved that it is possible to differentiate between certain tree species groups in aerial images. Our analysis showed that beeches and oaks are not well separable in our dataset. Consequently, these two classes were merged into one, resulting in a weighted class-averaged $F_1$-Score of 91.4%. As one could expect, the confusion between deciduous and coniferous species was low. A moderate level of confusion between douglas firs and larches, as well as between pines and the background class remains, which requires further investigation and a thorough check of the labeled data. Our approach relies on very high resolution aerial images, but in return it was able to distinguish more classes using fewer imaging bands than any other of the works mentioned in Section 2.3 (which was a selection of the best works the author was able to find).

There are different ways in which the presented work can be enhanced or expanded. One key point which requires further investigation is the normalization of the imagery in order to compensate atmospheric effects, changes of the local lighting conditions or different sensors. The effects of the atmosphere of course only play a role for satellite imagery. Several tools are capable of correcting the raw data to a normalized ground surface reflectance. However, these tools can fail and it has to be evaluated if it is possible to train a network which can cope with vastly varying conditions, e.g. by training on more diverse data or with better augmentation schemes. In the context of aerial images, the local lighting conditions play a role, as they change within one image sequence due to the time of day or clouds. These effects can only be partly compensated by software and require additional hardware calibration during the capturing, which should be considered in future campaigns.

Another improvement could be achieved by incorporating the basic ResNet building blocks into smaller U-Nets as well. As we saw in Section 6.3, pre-trained weights for these blocks are available and could be leveraged.

The most interesting possible future work is the combination of the three approaches presented in this work in order to perform a fully convolutional segmentation of trees, including their species. The aim is to arrive at a procedure, which does not rely on Lidar data and can therefore be applied to optical images. Such optical imagery is available from many land registry offices in Germany in high resolution. The best possible starting point would be the Gartow dataset and the methodological outline looks as follows: The Gartow dataset contains optical and Lidar information. From the Lidar information, the tree positions are extracted and a canopy height model is created, as it was done already. By performing a watershed segmentation on this height model, the tree crown outlines can be derived. First, an appropriate U-Net is trained for the segmentation of tree cover using these tree crown polygons. Employing the ResNet classifier, all trees within the dataset are classified along with the corresponding polygon. In this manner, a fully annotated dataset containing about one million trees could be created, covering approximately 140 km$^2$. In a second step, one U-Net is trained per tree species, departing from the pre-trained one. These U-Nets can then be applied to the raw optical information, without the need for Lidar data. Instead of performing the watershed segmentation on the canopy height model, one can perform it on the resulting probability maps, thereby separating the tree crowns. The tree positions could then be found in a similar way as was done to obtain the palm positions. In this manner, a detailed forest inventory for the entirety of Germany could be created.

We have performed first trials in the city of Göttingen, as the tree species and positions are known from cadastral data. The dataset however, was too small to allow for a rigorous accuracy assessment and therefore it could not be included in this work. Figure 7.1 shows a preliminary result: A map of copper beeches (Blutbuchen), linden and maple trees in the city of Göttingen.

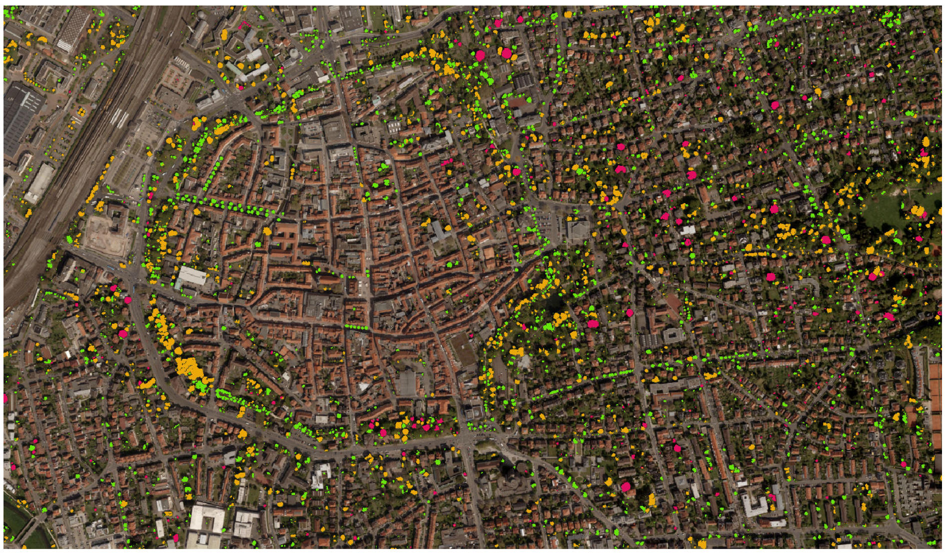


**Figure 7.1.:** A classification of the city trees in Göttingen. Copper beeches are shown in red, linden in green and maple trees in yellow. The underlying image was acquired by the *Landesamt für Geoinformation und Landesvermessung Niedersachsen* on 2016-05-06.

With regard to these early results, and in the context of the fast moving field of deep learning, it becomes clear that the work presented in this thesis is indeed expandable and leaves room for further research.

# Bibliography

[1] URL: https://directory.eoportal.org/web/eoportal/satellite-missions/s/skysat#sensors (visited on 2019-01-14).

[2] URL: https://www.nasa.gov/content/landsat-8-instruments (visited on 2019-01-14).

[3] URL: https://www.satimagingcorp.com/satellite-sensors/worldview-3/ (visited on 2018-10-15).

[4] URL: https://www.pveducation.org/pvcdrom/appendices/standard-solar-spectra (visited on 2018-10-15).

[5] United Nations Population Division. URL: https://population.un.org (visited on 2018-10-06).

[6] V. Badrinarayanan, A. Kendall, and R. Cipolla. „SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation“. In: *ArXiv e-prints* (2015). arXiv: 1511.00561.

[7] M. F. Bear, B. W. Connors, and M. A. Paradiso. *Neuroscience: Exploring the Brain*. 4th Edition. Wolters Kluwer, 2016.

[8] G. A. Carter. „Responses of Leaf Spectral Reflectance to Plant Stress“. In: *American Journal of Botany* 80 (1993), pp. 239–243. DOI: 10.1002/j.1537-2197.1993.tb13796.x.

[9] M. Castelluccio, G. Poggi, C. Sansone, and L. Verdoliva. „Land Use Classification in Remote Sensing Images by Convolutional Neural Networks“. In: *ArXiv e-prints* abs/1508.00092 (2015). arXiv: 1508.00092.

[10] J. Chen et al. „Global land cover mapping at 30m resolution: A POK-based operational approach“. In: *ISPRS Journal of Photogrammetry and Remote Sensing* 103 (2015). Global Land Cover Mapping and Monitoring, pp. 7–27. DOI: https://doi.org/10.1016/j.isprsjprs.2014.09.002.

[11] Y. Chen, Z. Lin, X. Zhao, G. Wang, and Y. Gu. „Deep learning-based classification of hyperspectral data“. In: *IEEE Journal of Selected topics in applied earth observations and remote sensing* 7 (2014), pp. 2094–2107.

[12] F. Chollet et al. *Keras*. https://keras.io. 2015.

[13] R. Corley. „How much palm oil do we need?“ In: *Environmental Science & Policy* 12 (2009), pp. 134–139. DOI: https://doi.org/10.1016/j.envsci.2008.10.011.

[14] R. DeFries and J. Townshend. „NDVI-derived land cover classifications at a global scale“. In: *International Journal of Remote Sensing* 15 (1994), pp. 3567–3586.

[15] DigitalGlobe. 2016. URL: https://dg-cms-uploads-production.s3.amazonaws.com/uploads/document/file/207/Radiometric_Use_of_WorldView-3_v2.pdf (visited on 2019-02-20).

[16] H. Dong, G. Yang, F. Liu, Y. Mo, and Y. Guo. „Automatic brain tumor detection and segmentation using U-Net based fully convolutional networks“. In: *annual conference on medical image understanding and analysis*. Springer. 2017, pp. 506–517.

[17] T. Dozat. „Incorporating Nesterov momentum into Adam“. In: *ICLR Workshop* (2016).

[18] J. Drescher et al. „Ecological and socio-economic functions across tropical land use systems after rainforest conversion“. Eng. In: *Philosophical Transactions of the Royal Society B: Biological Sciences* 371 (2016).

[19] V. Dumoulin and F. Visin. „A guide to convolution arithmetic for deep learning“. In: *ArXiv e-prints* (2016). arXiv: 1603.07285.

[20] M. Eggen, M. Ozdogan, B. F. Zaitchik, and B. Simane. „Land Cover Classification in Complex and Fragmented Agricultural Landscapes of the Ethiopian Highlands“. In: *Remote Sensing* 8 (2016). DOI: 10.3390/rs8121020.

[21] F. Fassnacht, H. Latifi, K. Stereńczak, A. Modzelewska, M. Lefsky, L. Waser, C. Straub, and A. Ghosh. „Review of studies on tree species classification from remotely sensed data“. In: *Remote Sensing of Environment* 186 (2016), pp. 64–87. DOI: 10.1016/j.rse.2016.08.013.

[22] M. Freudenberg, N. Nölke, A. Agostini, K. Urban, F. Wörgötter, and C. Kleinn. „Large Scale Palm Tree Detection In High Resolution Satellite Images Using U-Net“. In: *Remote Sensing* 11 (2019), p. 312.

[23] K. Fukushima. „Cognitron: A self-organizing multilayered neural network“. In: *Biological cybernetics* 20 (1975), pp. 121–136.

[24] B.-C. Gao. „NDWI - A normalized difference water index for remote sensing of vegetation liquid water from space“. In: *Remote sensing of environment* 58 (1996), pp. 257–266.

[25] F. A. Gers, J. Schmidhuber, and F. A. Cummins. „Learning to Forget: Continual Prediction with LSTM“. In: *Neural Computation* 12 (2000), pp. 2451–2471.

[26] A. Geymen and I. Baz. „Monitoring urban growth and detecting land-cover changes on the Istanbul metropolitan area“. In: *Environmental monitoring and assessment* 136 (2008), pp. 449–459.

[27] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016.

[28] K. He, X. Zhang, S. Ren, and J. Sun. „Deep Residual Learning for Image Recognition“. In: *arXiv e-prints* (2015). arXiv: 1512.03385.

[29] K. He, X. Zhang, S. Ren, and J. Sun. „Identity Mappings in Deep Residual Networks“. In: *arXiv e-prints* (2016). arXiv: 1603.05027.

[30] V. Iglovikov, S. Mushinskiy, and V. Osin. „Satellite Imagery Feature Detection using Deep Convolutional Neural Network: A Kaggle Competition“. In: *ArXiv e-prints* (2017). arXiv: 1706.06169.

[31] V. I. Iglovikov, S. S. Seferbekov, A. V. Buslaev, and A. Shvets. „TernausNetV2: Fully Convolutional Network for Instance Segmentation“. In: *ArXiv e-prints* abs/1806.00844 (2018). arXiv: 1806.00844.

[32] S. Ioffe and C. Szegedy. „Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift“. In: *ArXiv e-prints* (2015). arXiv: 1502.03167.

[33] S. Jacques and S. Prahl. *Spectra for various materials*. 2019. URL: https://omlc.org/spectra/PhotochemCAD/index.html (visited on 2019-01-28).

[34] E. Jones, T. Oliphant, P. Peterson, et al. *SciPy: Open source scientific tools for Python*. 2001.

[35] Y. Ke and L. J. Quackenbush. „A review of methods for automatic individual tree-crown detection and delineation from passive remote sensing“. In: *International Journal of Remote Sensing* 32 (2011), pp. 4725–4747. DOI: 10.1080/01431161.2010.494184.

[36] J. Kiefer and J. Wolfowitz. „Stochastic Estimation of the Maximum of a Regression Function“. In: *Ann. Math. Statist.* 23 (1952), pp. 462–466. DOI: 10.1214/aoms/1177729392.

[37] Y. Kim. „Convolutional neural networks for sentence classification“. In: *arXiv preprint* (2014). arXiv: 1408.5882.

[38] D. P. Kingma and J. Ba. „Adam: A Method for Stochastic Optimization“. In: *ArXiv e-prints* abs/1412.6980 (2014). arXiv: 1412.6980.

[39] B. Koch, U. Heyder, and H. Weinacker. „Detection of individual tree crowns in airborne lidar data“. In: *Photogrammetric Engineering & Remote Sensing* 72 (2006), pp. 357–363.

[40] E. Koon Cheang, T. Koon Cheang, and Y. Haur Tay. „Using Convolutional Neural Networks to Count Palm Trees in Satellite Images“. In: *ArXiv e-prints* (2017). arXiv: 1701.06462.

[41] A. Krizhevsky, I. Sutskever, and G. E. Hinton. „Imagenet classification with deep convolutional neural networks“. In: *Advances in neural information processing systems*. 2012, pp. 1097–1105.

[42] C. Kubitza, V. V. Krishna, K. Urban, Z. Alamsyah, and M. Qaim. „Land Property Rights, Agricultural Intensification, and Deforestation in Indonesia“. In: *Ecological Economics* 147 (2018), pp. 312–321. DOI: https://doi.org/10.1016/j.ecolecon.2018.01.021.

[43] N. Kussul, M. Lavreniuk, S. Skakun, and A. Shelestov. „Deep learning classification of land cover and crop types using remote sensing data“. In: *IEEE Geoscience and Remote Sensing Letters* 14 (2017), pp. 778–782.

[44] Y. LeCun, Y. Bengio, and G. Hinton. „Deep learning“. In: *Nature* 521 (2015), p. 436.

[45] D. Li, Y. Ke, H. Gong, and X. Li. „Object-based urban tree species classification using bi-temporal WorldView-2 and WorldView-3 images“. In: *Remote Sensing* 7 (2015), pp. 16917–16937.

[46] R. Li, W. Liu, L. Yang, S. Sun, W. Hu, F. Zhang, and W. Li. „DeepUNet: A Deep Fully Convolutional Network for Pixel-level Sea-Land Segmentation“. In: *ArXiv e-prints* abs/1709.00201 (2017). arXiv: 1709.00201.

[47] W. Li, H. Fu, and L. Yu. „Deep convolutional neural network based large-scale oil palm tree detection for high-resolution remote sensing images“. In: *2017 IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*. 2017, pp. 846–849. DOI: 10.1109/IGARSS.2017.8127085.

[48] W. Li, H. Fu, L. Yu, and A. Cracknell. „Deep Learning Based Oil Palm Tree Detection and Counting for High-Resolution Remote Sensing Images“. In: *Remote Sensing* 9 (2016), p. 22. DOI: 10.3390/rs9010022.

[49] Z. Liu, J. An, and Y. Jing. „A simple and robust feature point matching algorithm based on restricted spatial order constraints for aerial image registration“. In: *IEEE Transactions on Geoscience and Remote Sensing* 50 (2012), pp. 514–527.

[50] L. Loncan et al. „Hyperspectral pansharpening: A review“. In: *ArXiv e-prints* (2015). arXiv: 1603.07285.

[51] D. Lu. „The potential and challenge of remote sensing-based biomass estimation“. In: *International journal of remote sensing* 27 (2006), pp. 1297–1328.

[52] L. Ma, M. Li, X. Ma, L. Cheng, P. Du, and Y. Liu. „A review of supervised object-based land-cover image classification“. In: *ISPRS Journal of Photogrammetry and Remote Sensing* 130 (2017), pp. 277–293. DOI: https://doi.org/10.1016/j.isprsjprs.2017.06.001.

[53] G. Marcus. „Deep Learning: A Critical Appraisal“. In: *ArXiv e-prints* abs/1801.00631 (2018). arXiv: 1801.00631.

[54] Martin Abadi et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. 2015.

[55] T. Mikolov, M. Karafiát, L. Burget, J. Černocky, and S. Khudanpur. „Recurrent neural network based language model“. In: *Eleventh annual conference of the international speech communication association*. 2010.

[56] H. Nagendra and D. Gopal. „Street trees in Bangalore: Density, diversity, composition and distribution“. In: *Urban Forestry & Urban Greening* 9 (2010), pp. 129–137.

[57] H. Nagendra, S. Nagendran, S. Paul, and S. Pareeth. „Graying, greening and fragmentation in the rapidly expanding Indian city of Bangalore“. In: *Landscape and Urban Planning* 105 (2012), pp. 400–406.

[58] Y. E. Nesterov. „A method for solving the convex programming problem with convergence rate $O(1/k^2)$“. In: *Dokl. Akad. Nauk SSSR* 269 (1983), pp. 543–547.

[59] O. Nevalainen et al. „Individual Tree Detection and Classification with UAV-Based Photogrammetric Point Clouds and Hyperspectral Imaging“. In: *Remote Sensing* 9 (2017), p. 185. DOI: 10.3390/rs9030185.

[60] C. Padwick, M. Deskevich, F. Pacifici, and S. Smallwood. „WorldView-2 pan-sharpening“. In: *Proceedings of the ASPRS 2010 Annual Conference, San Diego, CA, USA*. Vol. 2630. 2010.

[61] M. Plotke. *3D Image-Kernel Convolution Animation*. URL: https://commons.wikimedia.org/wiki/File:3D_Convolution_Animation.gif (visited on 2018-12-02).

[62] D. Pouliot, D. King, F. Bell, and D. Pitt. „Automated tree crown detection and delineation in high-resolution digital camera imagery of coniferous forest regeneration“. In: *Remote Sensing of Environment* 82 (2002), pp. 322–334. DOI: https://doi.org/10.1016/S0034-4257(02)00050-0.

[63] W. Rawat and Z. Wang. „Deep convolutional neural networks for image classification: A comprehensive review“. In: *Neural computation* 29 (2017), pp. 2352–2449.

[64] M. Rezaee, Y. Zhang, R. Mishra, F. Tong, and H. Tong. „Using a VGG-16 Network for Individual Tree Species Detection with an Object-Based Approach“. In: *2018 10th IAPR Workshop on Pattern Recognition in Remote Sensing (PRRS)*. IEEE. 2018, pp. 1–7.

[65] O. Ronneberger, P. Fischer, and T. Brox. „U-Net: Convolutional Networks for Biomedical Image Segmentation“. In: *ArXiv e-prints* (2015). arXiv: 1505.04597.

[66] F. Rosenblatt. „The perceptron: a probabilistic model for information storage and organization in the brain.“ In: *Psychological review* 65 (1958), p. 386.

[67] J. W. Rouse Jr, R. H. Haas, J. Schell, and D. Deering. *Monitoring the vernal advancement and retrogradation (green wave effect) of natural vegetation*. Texas A&M University, Remote Sensing Center, 1973.

[68] S. Ruder. „An overview of gradient descent optimization algorithms“. In: *ArXiv e-prints* abs/1609.04747 (2016). arXiv: 1609.04747.

[69] H. Santoso, T. Gunawan, R. H. Jatmiko, W. Darmosarkoro, and B. Minasny. „Mapping and identifying basal stem rot disease in oil palms in North Sumatra with QuickBird imagery“. In: *Precision Agriculture* 12 (2011), pp. 233–248. DOI: 10.1007/s11119-010-9172-7.

[70] G. J. Scott, M. R. England, W. A. Starms, R. A. Marcum, and C. H. Davis. „Training deep convolutional neural networks for land–cover classification of high-resolution imagery“. In: *IEEE Geoscience and Remote Sensing Letters* 14 (2017), pp. 549–553.

[71] H. Z. M. Shafri, N. Hamdan, and M. I. Saripan. „Semi-automatic detection and counting of oil palm trees from high spatial resolution airborne imagery“. In: *International Journal of Remote Sensing* 32 (2011), pp. 2095–2115. DOI: 10.1080/01431161003662928.

[72] H. Z. Shafri and N. Hamdan. „Hyperspectral imagery for mapping disease infection in oil palm plantationusing vegetation indices and red edge techniques“. In: *American Journal of Applied Sciences* 6 (2009), p. 1031.

[73] P. Srestasathiern and P. Rakwatin. „Oil Palm Tree Detection with High Resolution Multi-Spectral Satellite Imagery“. In: *Remote Sensing* 6 (2014), pp. 9749–9774. DOI: 10.3390/rs6109749.

[74] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. „Dropout: a simple way to prevent neural networks from overfitting“. In: *The Journal of Machine Learning Research* 15 (2014), pp. 1929–1958.

[75] H. Sudhira and H. Nagendra. „Local assessment of Bangalore: graying and greening in Bangalore–Impacts of urbanization on ecosystems, ecosystem services and biodiversity“.

In: *Urbanization, Biodiversity and Ecosystem Services: Challenges and Opportunities*. Springer, 2013, pp. 75–91.

[76] P. Thanh Noi and M. Kappas. „Comparison of random forest, k-nearest neighbor, and support vector machine classifiers for land cover classification using Sentinel-2 imagery“. In: *Sensors* 18 (2018), p. 18.

[77] The Forest Trust. „All of Nestlé’s palm oil supply chain to be 100% satellite monitored“. In: (2018). 2018-12-14. DOI: http://www.tft-earth.org/stories/news/nestlesatellite/.

[78] M. Thyssen. URL: https://commons.wikimedia.org/wiki/File:Grevys_zebra.jpg (visited on 2018-11-20).

[79] M. Treml et al. „Speeding up semantic segmentation for autonomous driving“. In: *MLITS, NIPS Workshop*. 2016.

[80] Y. Wu et al. „Google’s Neural Machine Translation System: Bridging the Gap between Human and Machine Translation“. In: *arXiv preprint* (2016). arXiv: 1609.08144v2.

[81] P. Yakubovskiy. *Classification Models Zoo*. URL: https://github.com/qubvel/classification_models (visited on 2019-01-15).

[82] P. Yakubovskiy. *Segmentation Models Zoo*. 2018. URL: https://github.com/qubvel/segmentation_models (visited on 2018-12-16).

[83] L. Zhang, L. Zhang, and B. Du. „Deep learning for remote sensing data: A technical tutorial on the state of the art“. In: *IEEE Geoscience and Remote Sensing Magazine* 4 (2016), pp. 22–40.

[84] Z. Zhang, Q. Liu, and Y. Wang. „Road Extraction by Deep Residual U-Net“. In: *ArXiv e-prints* abs/1711.10684 (2017). arXiv: 1711.10684.

[85] X. X. Zhu, D. Tuia, L. Mou, G. Xia, L. Zhang, F. Xu, and F. Fraundorfer. „Deep Learning in Remote Sensing: A Comprehensive Review and List of Resources“. In: *IEEE Geoscience and Remote Sensing Magazine* 5 (2017), pp. 8–36. DOI: 10.1109/MGRS.2017.2762307.

[86] B. Zitova and J. Flusser. „Image registration methods: a survey“. In: *Image and vision computing* 21 (2003), pp. 977–1000.

# A. Appendix

## A.1. Appendix A

This section is related to Section 6.1, which deals with the segmentation of tree cover.

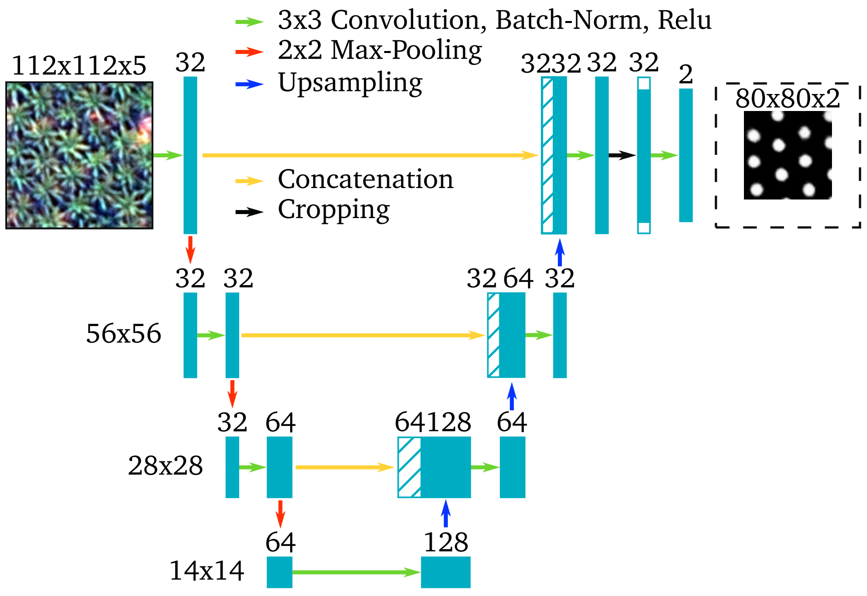


**Figure A.1.:** The architecture of U-Net B. It features four stages and has approximately 260 000 parameters.

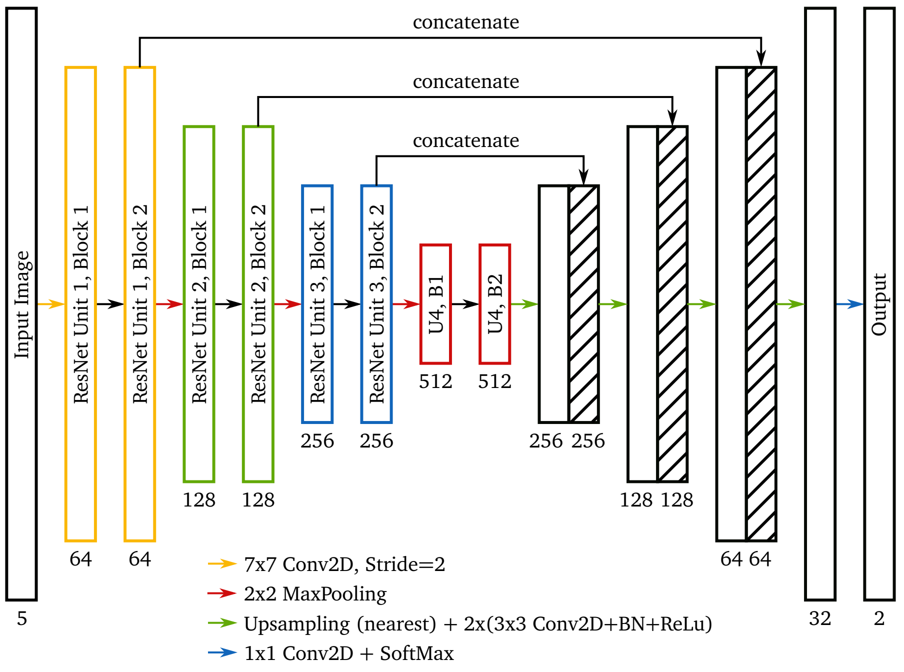


**Figure A.2.:** Architecture of U-ResNet with a ResNet-18 encoder [82], sketched in a linear way. The colors of boxes and arrows are not related. The boxes in the encoder are colored in the same way as in Figure 4.5 and perform the same operations. The numbers at each block indicate the number of filters in the feature maps.

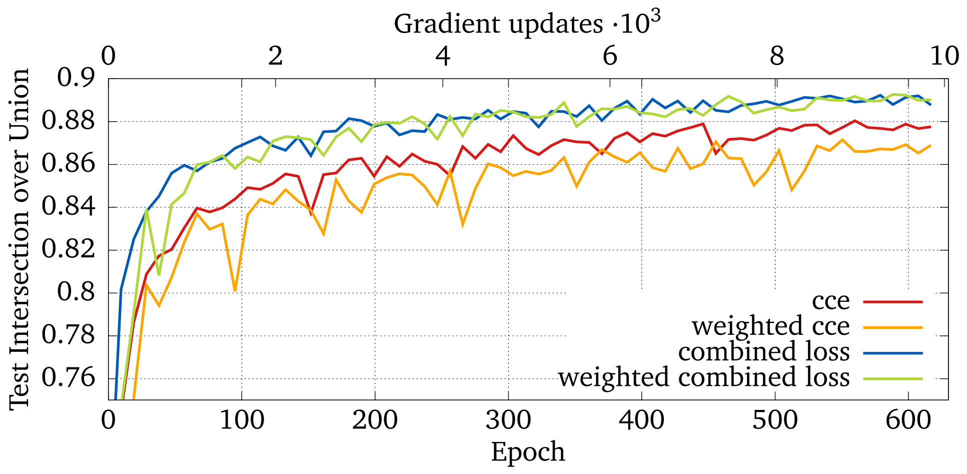


**Figure A.3.:** Intersection over union on the test set for U-Net B trained with different loss functions. The categorical cross entropy loss and the combined loss function presented in Section 3.1.6 were tested in original form and in a weighted form, accounting for class imbalance.

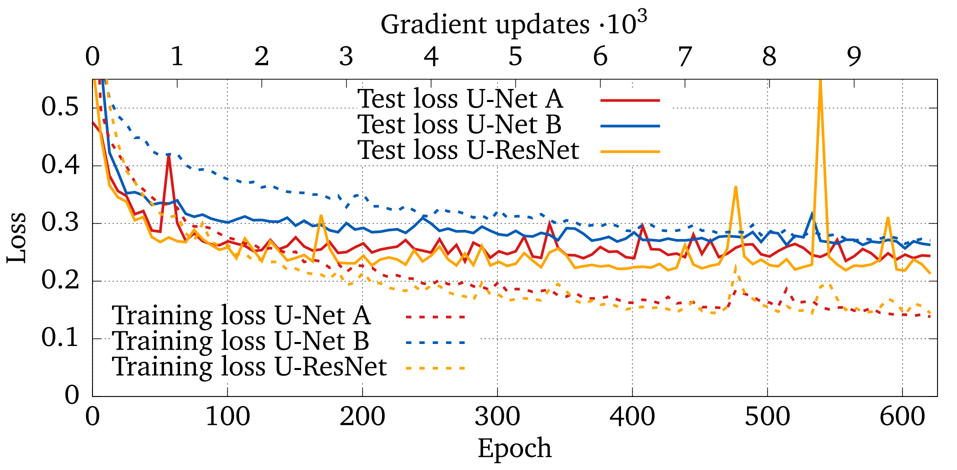


**Figure A.4.:** Training (dashed) and test loss (solid) in the final run for the three tested networks.

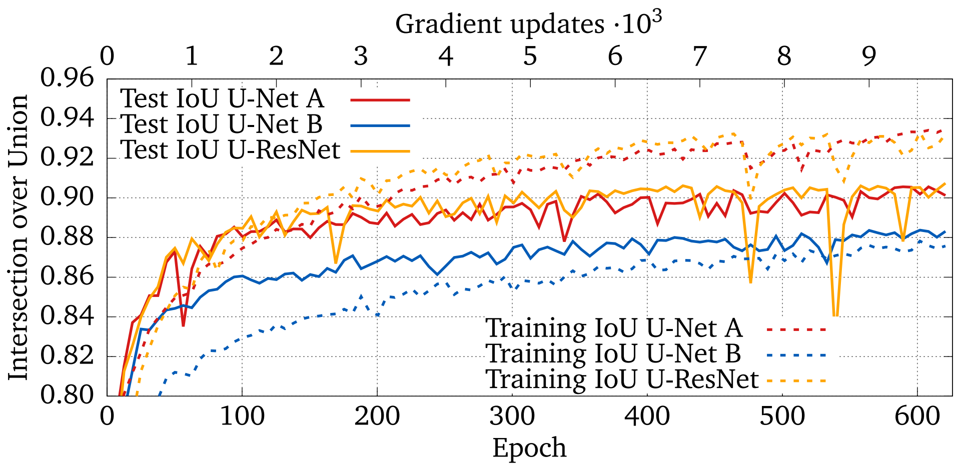


**Figure A.5.:** Training (dashed) and test (solid) intersection over union in the final run for the three tested networks.

## A.2. Appendix B

This section is related to Section 6.2, which deals with locating and counting palm trees.

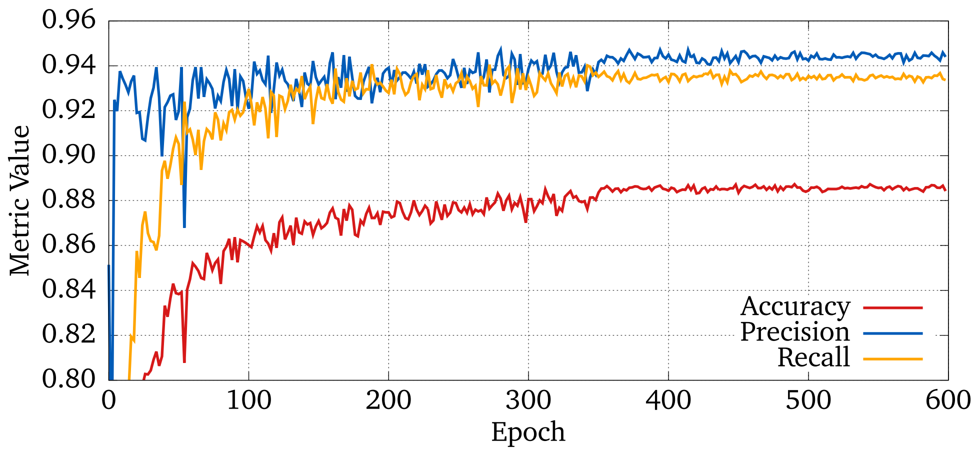


**Figure A.6.:** Test accuracy, precision, and recall of architecture A on the Jambi dataset. The graphs show the metrics derived from the predicted positions averaged over all k-fold runs. The curves for U-Net B look similar.

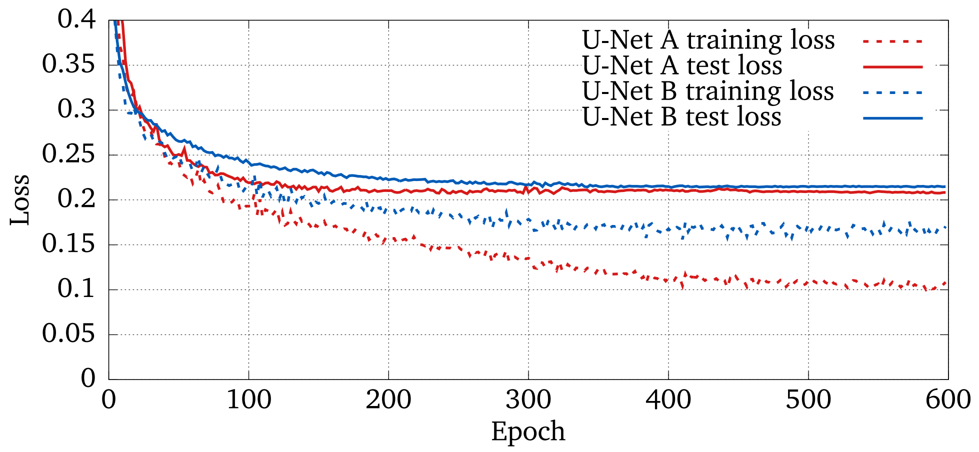


**Figure A.7.:** Training and test loss for U-Net A and B, averaged over 10 cross-validation runs. U-Net B has a lower spread between training and test loss, which indicates that it is less prone to overfitting. This is due to the lower number of parameters (260 000 vs. 7.8 million).

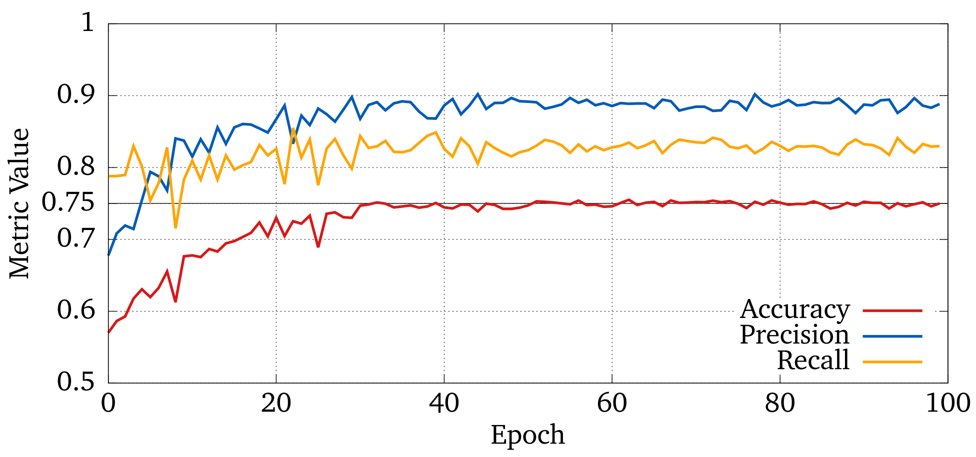


**Figure A.8.:** Test accuracy, precision, and recall for the AlexNet classifier [47], averaged over all k-fold runs.

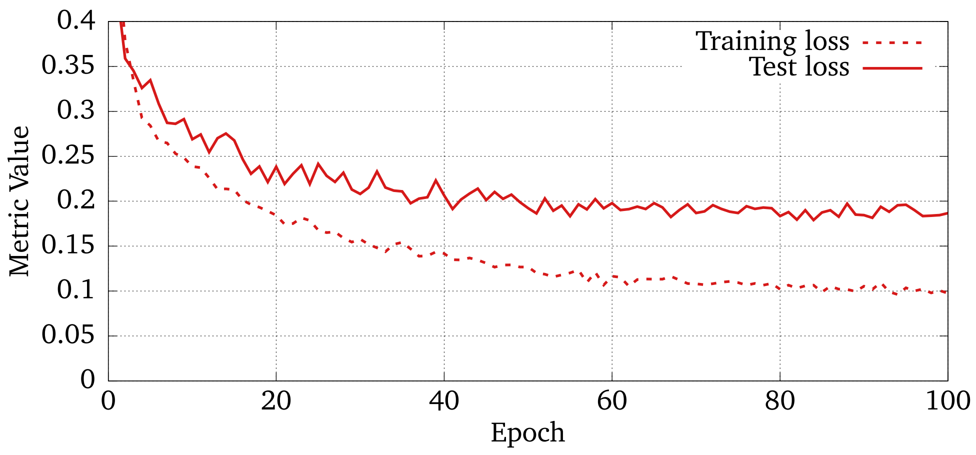
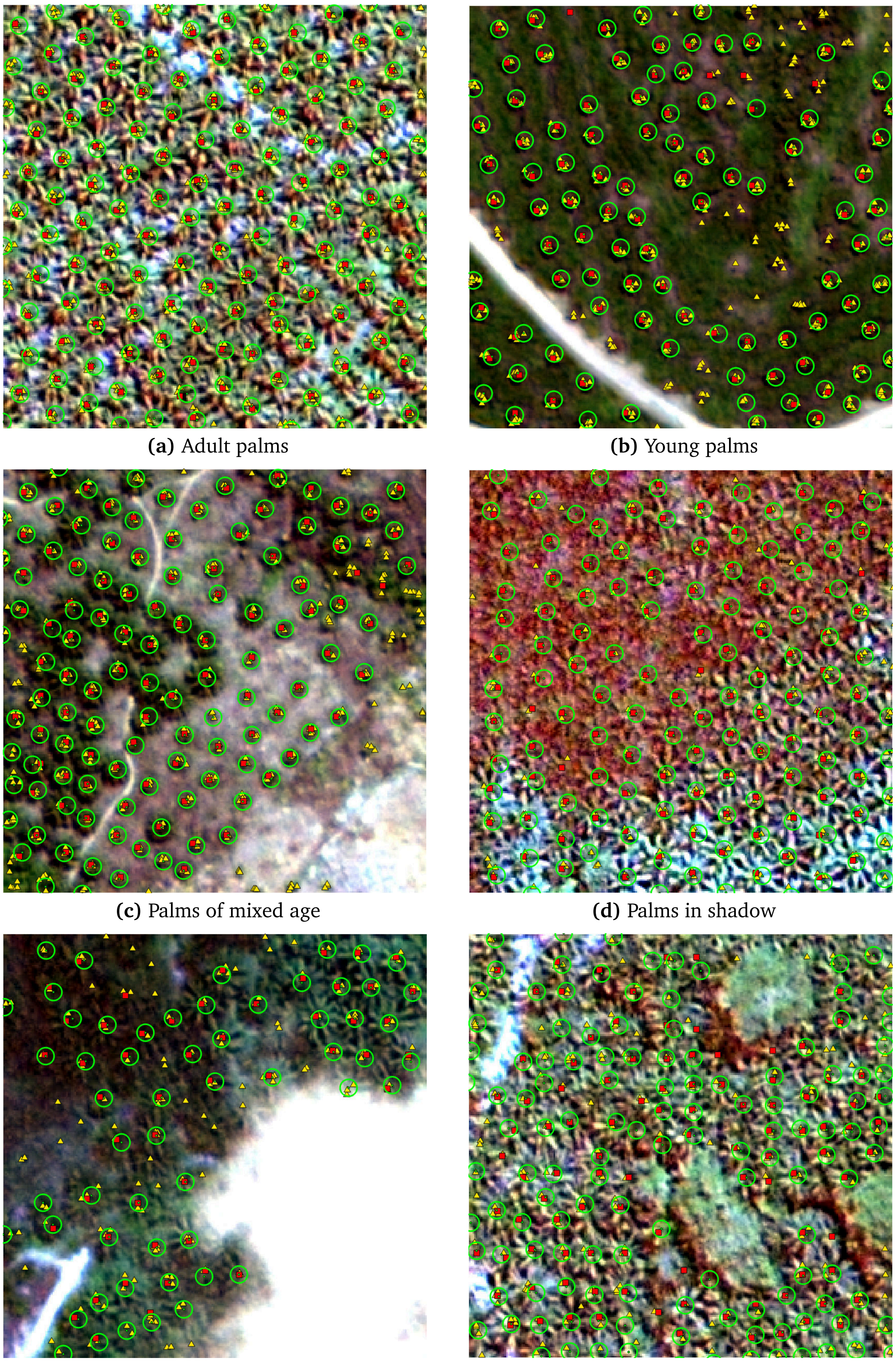


**Figure A.9.:** Categorical cross entropy loss of the classifier.



**(a)** Adult palms

**(b)** Young palms

**(c)** Palms of mixed age

**(d)** Palms in shadow

**(e)** Palms in shadow and under clouds

**(f)** Mixed vegetation

**Figure A.10.:** The results of the comparison humans versus U-Net. Ground truth labels shown green, including the radius for correct detection. The network prediction is shown in red and the human labels in yellow.

## A.3. Appendix C

This section is related to Section 6.3, which deals with classifying tree species groups.

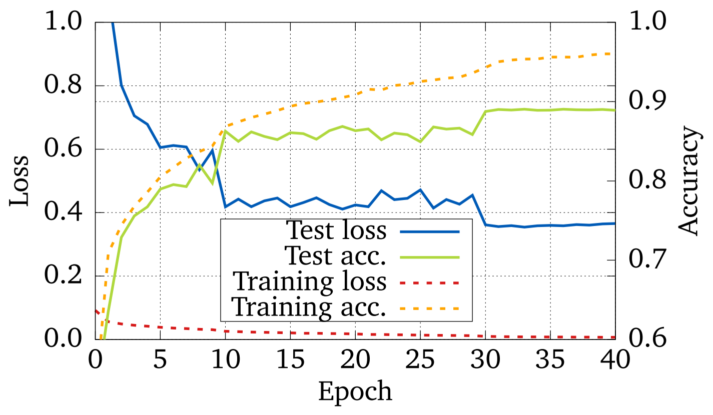
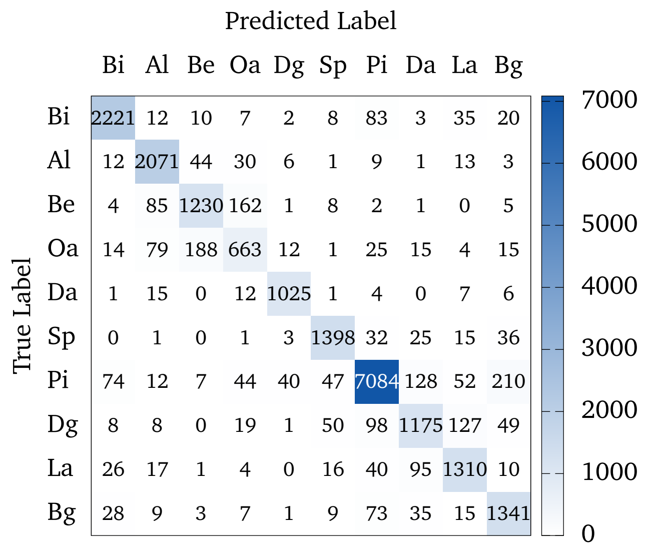


**Figure A.11.:** Loss and accuracy for training with all ten classes, averaged over ten cross-validation runs.



| | Bi | Al | Be | Oa | Dg | Sp | Pi | Da | La | Bg |
|---|---|---|---|---|---|---|---|---|---|---|
| Bi | 2221 | 12 | 10 | 7 | 2 | 8 | 83 | 3 | 35 | 20 |
| Al | 12 | 2071 | 44 | 30 | 6 | 1 | 9 | 1 | 13 | 3 |
| Be | 4 | 85 | 1230 | 162 | 1 | 8 | 2 | 1 | 0 | 5 |
| Oa | 14 | 79 | 188 | 663 | 12 | 1 | 25 | 15 | 4 | 15 |
| Da | 1 | 15 | 0 | 12 | 1025 | 1 | 4 | 0 | 7 | 6 |
| Sp | 0 | 1 | 0 | 1 | 3 | 1398 | 32 | 25 | 15 | 36 |
| Pi | 74 | 12 | 7 | 44 | 40 | 47 | 7084 | 128 | 52 | 210 |
| Dg | 8 | 8 | 0 | 19 | 1 | 50 | 98 | 1175 | 127 | 49 |
| La | 26 | 17 | 1 | 4 | 0 | 16 | 40 | 95 | 1310 | 10 |
| Bg | 28 | 9 | 3 | 7 | 1 | 9 | 73 | 35 | 15 | 1341 |

**Figure A.12.:** Confusion matrix for all ten classes, aggregated over ten cross-validation runs.

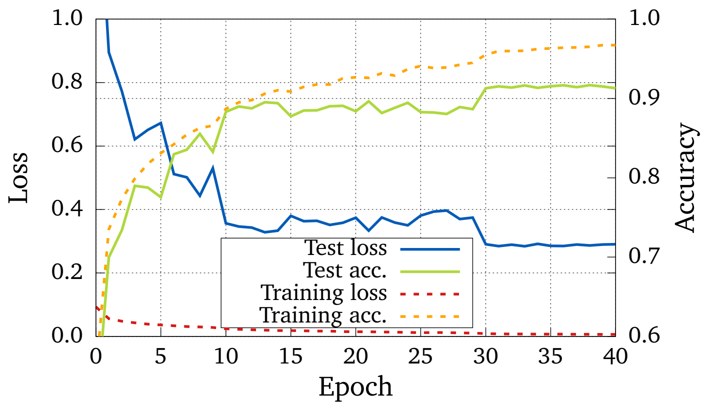
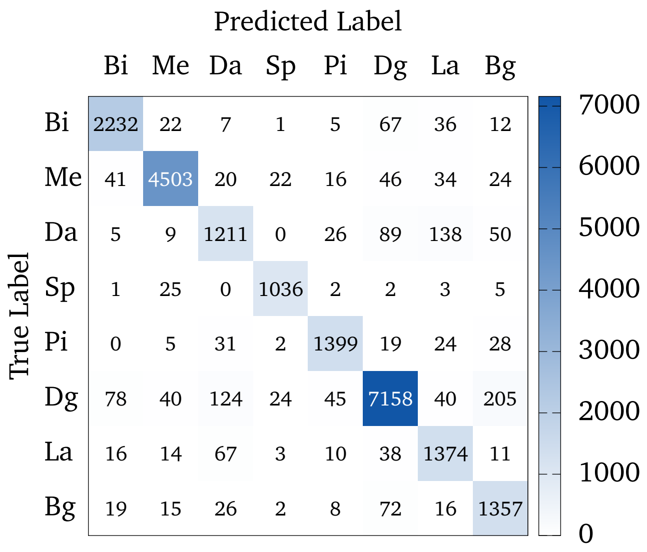


**Figure A.13.:** Loss and accuracy for training with beech and oak merged into one class, averaged over ten cross-validation runs.



| | Bi | Me | Da | Sp | Pi | Dg | La | Bg |
|---|---|---|---|---|---|---|---|---|
| Bi | 2232 | 22 | 7 | 1 | 5 | 67 | 36 | 12 |
| Me | 41 | 4503 | 20 | 22 | 16 | 46 | 34 | 24 |
| Da | 5 | 9 | 1211 | 0 | 26 | 89 | 138 | 50 |
| Sp | 1 | 25 | 0 | 1036 | 2 | 2 | 3 | 5 |
| Pi | 0 | 5 | 31 | 2 | 1399 | 19 | 24 | 28 |
| Dg | 78 | 40 | 124 | 24 | 45 | 7158 | 40 | 205 |
| La | 16 | 14 | 67 | 3 | 10 | 38 | 1374 | 11 |
| Bg | 19 | 15 | 26 | 2 | 8 | 72 | 16 | 1357 |

**Figure A.14.:** The confusion matrix for the new training run with beech and oak merged into one class (Me). The matrix was aggregated over ten cross-validation runs.

# Declaration of Originality - Eigenständigkeitserklärung

Ich versichere hiermit, dass ich die vorliegende Arbeit ohne fremde Hilfe selbstständig verfasst und nur die von mir angegebenen Quellen und Hilfsmittel verwendet habe. Wörtlich oder sinngemäß aus anderen Werken entnommene Stellen habe ich unter Angabe der Quellen kenntlich gemacht. Die Richtlinien zur Sicherung der guten wissenschaftlichen Praxis an der Universität Göttingen wurden von mir beachtet. Eine gegebenenfalls eingereichte digitale Version stimmt mit der schriftlichen Fassung überein. Mir ist bewusst, dass bei Verstößen gegen diese Grundsätze die Prüfung mit nicht bestanden bewertet wird.

Göttingen, den 28.03.2019