

UNIVERSITY OF MÜNSTER  
DEPARTMENT OF INFORMATION SYSTEMS

---

Deep Learning Semantic Segmentation of Tree  
Stock in South Africa Using Satellite Images

---

BACHELOR THESIS

submitted by

Leo Richard Hermann Giesen

CHAIR OF DATA SCIENCE:  
MACHINE LEARNING AND DATA ENGINEERING

<b>Principal Supervisor</b>	PROF. DR. FABIAN GIESEKE
<b>Supervisor</b>	SUPERVISOR, SUBST.-PROF. DR. FRIEDRICH CHASIN Chair for Data Science: Machine Learning and Data Engineering
<b>Student Candidate</b>	Leo Richard Hermann Giesen
<b>Matriculation Number</b>	462502
<b>Field of Study</b>	Information Systems
<b>Contact Details</b>	leo.giesen@uni-muenster.de
<b>Submission Date</b>	14.04.2022

## **Abstract**

A significant proportion of Africa’s drylands trees and shrubs (subsequently collectively referred to as trees) grow in an isolated manner without canopy closure. Nevertheless, these non-forest trees crucially contribute to the biodiversity and ecosystem because they store carbon, offer food resources, and shelter to humans and animals [Bra+20; Str+12; Bay+14]. Therefore, single trees are detected by a deep learning approach utilizing the U-Net architecture, which enables an analysis of the tree stock development in South Africa.

The number of trees also provides an estimate of the dryland’s biomass. Thus, the idea is to visualize these results on a website devoted to sustainability. These results can be used to create awareness concerning the fostering of sustainability by publishing the insights on an information system. For instance, a website could highlight land degradation, desertification, deforestation or reforestation, which may be derived from monitoring the tree count.

# Contents

1	Task Specification and Project Relevance .....	1
1.1	Project Motivation and Content of Task .....	1
1.2	Project Objectives and Goals.....	2
2	Ecological Situation in South Africa .....	3
2.1	South African Landscape .....	3
2.2	Vegetation Degradation in South Africa .....	3
2.3	Project Applicability in Sustainability Context .....	5
3	Satellite Data and Project Feasibility .....	9
4	Technological Approach .....	11
4.1	Classification of the Task in Computer Vision Context .....	11
4.2	Artificial Intelligence .....	12
4.3	Machine Learning .....	12
4.3.1	Definition.....	12
4.3.2	Distinction from Heuristic and Analytical Approaches .....	13
4.3.3	Types Machine Learning Systems .....	13
4.4	Deep Learning and Convolutional Neural Networks .....	14
4.5	Fully Convolutional Network .....	16
4.6	U-Net Architecture .....	17
5	Implementation and Insights for Future Projects.....	19
5.1	Project Pipeline.....	19
5.2	Data Creation, Processing and Export .....	20
5.3	Data Preparation: Patch Creation and Augmentation .....	20
5.4	U-Net Model and Training on GPU and Prediction .....	23
6	Limitations and Further Improvement .....	26
A	Appendix .....	27
A.1	Pipeline .....	27
A.2	QGIS Label Creation .....	28
A.3	Project Code .....	32
	Bibliography .....	37

# 1 Task Specification and Project Relevance

## 1.1 Project Motivation and Content of Task

The detection of trees is relevant because it enables an estimate of biomass and corresponding carbon sequestration or offset. What is more, it allows for a very detailed analysis of the vegetation. The tree stock development analysis can be interpreted in a larger context in combination with environmental news. For instance, how humans influence the environment's vegetation by farming and deforestation or how animals are affected by the rapid decline of habitat. Therefore, this detailed analysis offers more insights than a broad canopy cover analysis at large spatial scales.

The deep learning model developed in this project is designed to capture single trees and shrubs (hereafter collectively referred to as trees) in South Africa (see Figure 1). This is crucial as many trees in South Africa grow in an isolated manner without canopy closure and are not part of a more extensive forest (see Chapter 2.1). These non-forest trees crucially contribute to the biodiversity and ecosystem because they store carbon, offer food resources, and shelter to humans and animals [Bra+20; Str+12; Bay+14].

The analyzed data is accessed under the license from the Chief Directorate: National Geo-spatial Information (CD:NGI) [Chi22], which comprises regional satellite data from South Africa.

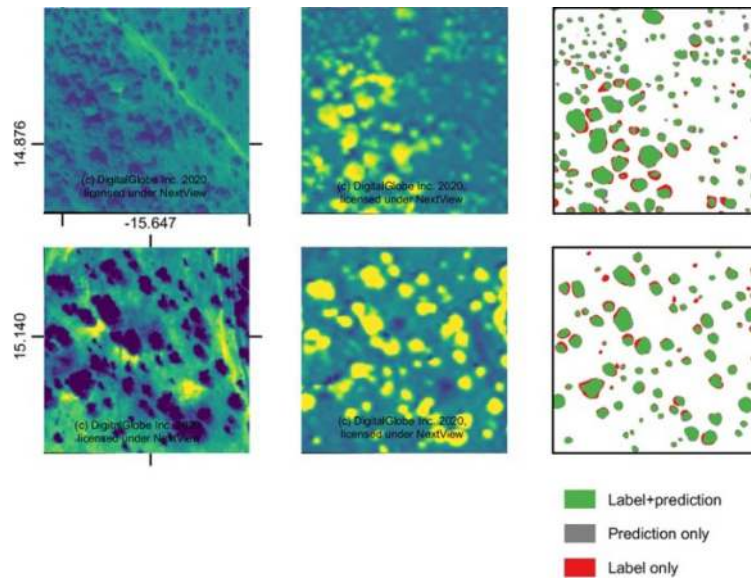


Figure 1 Tree Mapping with U-Net [Bra+20]

## 1.2 Project Objectives and Goals

The project builds upon Brandt et al. [Bra+20], who discovered a relatively high density of isolated trees in the Sahel, which challenges the desertification of central Africa. Because of a lack of public research concerning isolated tree coverage, this thesis aims to check if the insights and approach from Brandt et al. are transferable to South Africa with an implementation using the state-of-the-art technology U-Net [SKS15]. The transferability of the approach can be subdivided into three sub-goals to measure the overall progress.

The first objective is to investigate to what extent the deep learning U-Net architecture, which is applied by Brandt et al. is a suitable approach for segmenting trees with the given data (see Chapter 4.6). This is initially performed with a theoretical feasibility analysis (see Chapter 3) and confirmed practically with a U-Net implementation trained on the South African trees (see Chapter 5.4).

The second objective is to understand how to generate data for the deep learning model, which is performed in the Quantum Geographic Information System (QGIS). The data generation can be subdivided into the labeling process, export process, import and connection to the project pipeline (see Chapter 5.2). Furthermore, it is conceptualized how the data generation in QGIS can be connected and utilized in a deep learning context.

The third objective is the development of a deep learning model, which detects trees in South Africa. The deep learning model utilizes the convolutional neural network architecture U-Net, which is built for “fast and precise segmentation of images” [RFB15a]. An overview of the implementation and a step-wise explanation are shed light on in Chapter 5.

As an outlook, the output of the U-Net’s prediction can be used to determine various metrics affecting sustainability, such as tree density, crown size, and tree cover. These can be utilized in various use cases presented in Chapter 2.3. Moreover, a map of predicted trees in QGIS can provide an overview of tree coverage, which can be compared and evaluated with other sources such as Hansen et al. [Han+13]. Brandt et al. required a vast number of training samples to perform well. Due to a tight schedule, this project does not have as many labeled instances, which could be mitigated by transfer learning or generating new training instances by correcting the model’s prediction of unlabeled areas (see Chapter 6).

## 2 Ecological Situation in South Africa

### 2.1 South African Landscape

It is crucial to understand the analyzed type of data in the project to build foundational knowledge about what to predict and to comprehend the nature of the results. Moreover, knowledge about the landscape helps to evaluate if the results are correct and satisfactory, which is why the South African landscape is shed light on.

According to the National Geographic Society, most of South Africa’s landscape is covered with plateaus, which are a “flat, elevated landform that rises sharply above [the] surrounding area on at least one side” [Nat22]. There are many rolling grasslands and thorn trees at a high altitude in the northeast of South Africa, called highveld [Lex22; The14; Nat15; Siy21].

Additionally, southern Africa and especially the provinces of Limpopo, Western Botswana, and Southern Zimbabwe offer bushveld, which are plains with isolated low-growing thorn trees and bushes (see Figure 4b) [Nat15; Aca22; The15; Mer21]. These regions are vital, because the data inspection confirmed that the satellite images cover the provinces Limpopo and Mpumalanga including the Kruger National Park.

Surrounding the plateau lands is the Great Escarpment, which is made up of mountains [Nat15]. On the east side near the coast is the Drakensberg or Dragon’s Mountain, which has jagged peaks. The outlined landscape, including the bushveld and highveld regions and satellite coverage window, is displayed in Figure 2.

### 2.2 Vegetation Degradation in South Africa

It is sensible to capture the development of the vegetation to assess the situation. In the last decade, there have been contrasting statements from various sources concerning the development of South Africa’s vegetation. Thus, monitoring the growth or loss of South Africa’s vegetation is necessary and helpful.

For example, the Review of Forest and Landscape Restoration in Africa 2021 report outlines an average forest expansion of 162.600 hectares per year from 2010 to 2020 [MB21, Tab. 3]. The World Wildlife Fund mentions that the highveld grassland has “expanded or contracted in response to climate change. During the Quaternary, grassland expanded in response to glacial events to the north” [BF22]. Moreover,

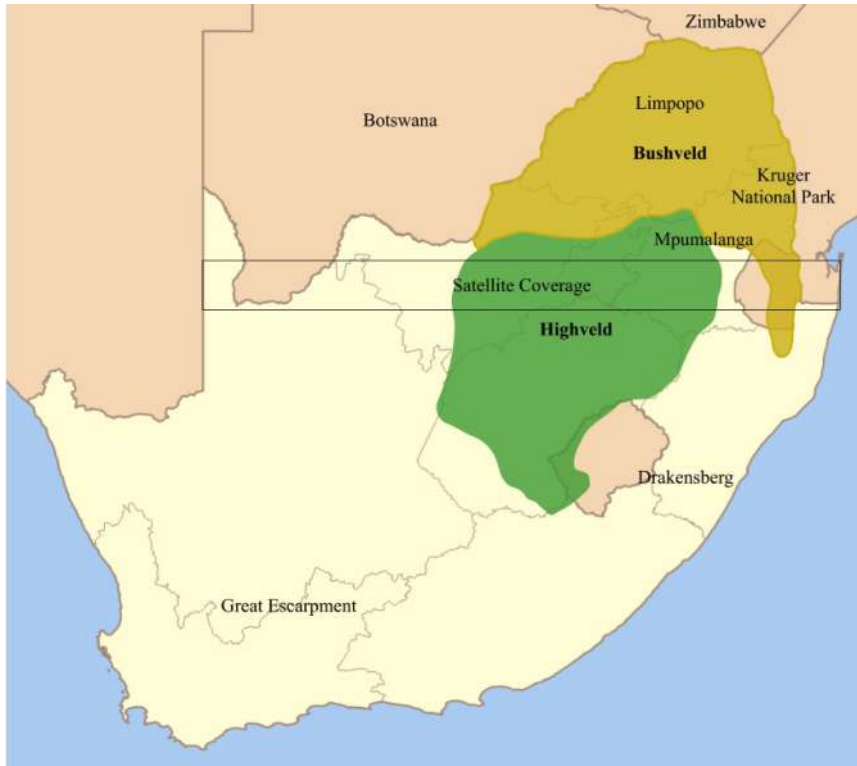


Figure 2 Map of South Africa’s Highveld and Bushveld (extends Htonl [Hto10] and summarizes Merwe [Mer19] and Kumirai [Kum12])

“vast areas of planted forest can be found consisting of fast-growing exotic species such as *Pinus* spp., *Eucalyptus* spp. and *Acacia mearnsii*” [MB21, p. 29].

Nevertheless, the same report accentuates a loss of vegetation in Southern Africa because of persistent drought cycles [MB21]. This is linked to dramatic habitat destruction resulting from anthropogenic alterations [BF22; Enc22]. These environmental changes can be traced back to human unsustainable land use practices, which minimize the ecological functions of South African soil, its fertility, productivity, food, and water supply [MB21, p. 29-30]. Those actions comprise “subsistence agriculture, fire, mining, overgrazing, fuelwood collection, illegal logging, unsustainable wood harvesting for timber and wood fuel and infrastructure development” [MB21, p. 29-30].

This land degradation also takes form in an increasing loss of forests due to climate change [Int22; BF22]. This stance is also taken by the WWF, which stresses that Africa’s vegetation has deteriorated fast in the last decade [BF22]. Because of the significant level of land degradation, the WWF evaluates the ecological situation of the highveld grassland as critical and endangered [BF22]. The loss of vegetation is also reflected in the satellite images from 2009 and 2016 from this project (see Figure 3).



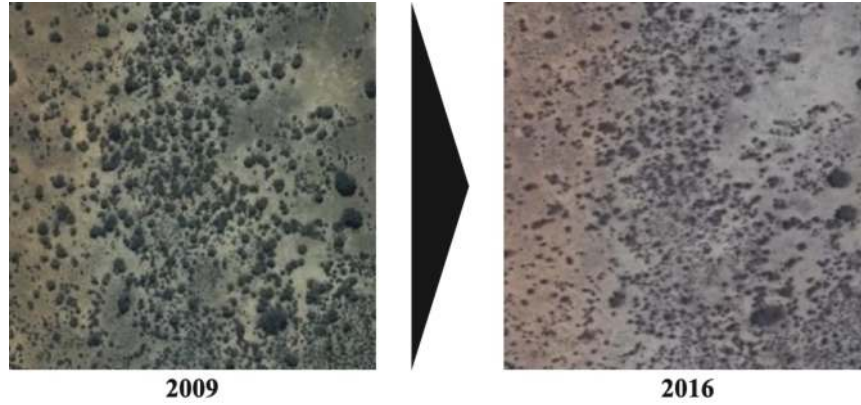


Figure 3 Decline in Vegetation in South Africa (licensed by CD:NGI [Chi22])

Only half a percent of the ecoregion is in a near-untouched state and these nature reserve habitats continuously lose in size and quality [BF22]. Consequently, the African Forest Landscape Restoration Initiative aims to build up the vegetation by its full restoration potential of 3,6 mha [MB21, Fig. 3, Tab. 8]. This counteraction of degradation is addressed and tackled with ecological restoration and maintenance measures in the Working for Ecosystems project.

With funding of USD 147,6 million, this project’s objective is to “regain natural ecosystem composition, structure and function and enhance ecosystem services. At the same time, the aim is to improve livelihoods security and the productive potential of land, improve natural species diversity, and promote the development of a market for ecosystem services and pro-poor economic development and empowerment in rural areas” [MB21, p. 31].

The International Climate Initiative criticizes that the landscape restoration is currently hindered from scaling by “poor resource governance, little access to innovation and resources and unfavorable policies” [Int22]. Therefore more funding is required for projects, such as the Working for Ecosystems to rebuild the vegetational landscape of the highveld grassland. The preservation and reforestation of it are paramount because it inhabits species, which are endangered, such as the blue crane (*anthropoides paradisea*) (see Figure 4a) [BF22].

### 2.3 Project Applicability in Sustainability Context

An objective and measurable status and development of the South African vegetation is necessary for its monitoring, preservation, and restoration. This can be partly addressed by capturing the current tree stock, which is performed by the deep learning model presented in this thesis.



(a) Highveld-Endemic Blue Crane [Mun22] (b) Highveld Umbrella Thorn Tree [Siy19]

Figure 4 Landscape and Wildlife Characteristics in South Africa

In detail, the U-Net model takes a high-resolution satellite image with red, green and blue channels as an input, predicts the trees in that image and outputs the labeled area (see Figure 5).



Figure 5 Model Input and Output (licensed by CD:NGI [Chi22])

This output can be analyzed regarding various helpful and vital figures, which provide insights into the vegetational situation. For example, the tree count, canopy coverage, or average tree diameter can be extracted from the model’s prediction.

These key figures can be used to determine the above-ground biomass (AGB), which is a meaningful indicator because “accurate biomass measurements and analyses are critical components in quantifying carbon stocks and sequestration rates, assessing potential impacts due to climate change, locating bio-energy processing plants, and mapping and planning fuel treatments. To this end, biomass equations will remain a key component of future carbon measurements and estimation” [Tem+15].

The AGB can be determined by either direct or indirect estimation methods. The former is performed by collecting a biomass sample, so trees are extracted and weighed. The weight is then extrapolated to larger areas. The direct method is not chosen in this project because of its high destruction and effort.

Therefore, the insights from this project may be used to indirectly estimate the AGB by using a formula, which is relatively accurate (see Formula 2.1) [MHD99]. It is

worth noting that these formulas may be more or less accurate in some areas than others because some tree species' wood varies in weight.

The output of the project model favors a tree-level-based formula because trees are captured. Maia Araújo, Higuchi, and De Carvalho [MHD99] and Lin et al. [Lin+18, p. 8] both utilize the same formula with slightly different regression coefficients (see Formula 2.1). They are initialized differently because different types of trees were inspected. I. e. in the case of Lin et al., the diameter at breast height has a more significant impact on the overall biomass than in the other project. This might be the case because the inspected trees from Lin et al. are wider than the ones from Maia Araújo, Higuchi, and De Carvalho.

The diameter at breast height is impossible to grasp from the given satellite images. Hence, Lin et al.'s formula requiring only the tree height is more suitable but could also be less accurate (see Formula 2.2).

$$AGB_1 = \alpha_1 \cdot D^{\beta_1} \cdot H^{\gamma} \quad (2.1)$$

2.1 Estimation of Above-Ground Biomass (AGB) (AGB in kg, D = Diameter at Breast Height in cm, H = Tree Height in m) [MHD99; Lin+18]

$$AGB_2 = \alpha_2 \cdot H^{\beta_2} \quad (2.2)$$

## 2.2 Estimation of Above-Ground Biomass [Lin+18]

The AGB and the previously mentioned key figures might be presented in an information system accessible to local authorities and environmental initiatives. This would enable them to take more targeted action toward preserving the landscape and wildlife in the South African highveld and bushveld.

The provided satellite image data spans over more than a decade. Hence, the tree stock development can be captured, which is critical for identifying successful environmental measures. This map of tree stock development and current status could be presented on a website to create awareness for South Africa's deforestation. Moreover, other key figures could be presented, such as deforestation, reforestation, or desertification indicators, which could be observed and measured closely with the satellite image analysis powered by deep learning.

South Africa could present the results from the website's vegetation development analysis at climate conferences, such as the United Nations Climate Change Conference. For example, South Africa could voice very informative and convincing arguments on reforestation support when they present a hypothetical total net loss of four million trees in South Africa's 40 million hectares of forest [Sou14]. This loss

could be drilled down to areas with more rapid tree stock decline to identify regions of need and quickly determine specific measures.

The model is oriented and developed for tree stock segmentation, but further trained models could be incorporated to extend the website to South Africa’s land loss, live-stock, river masses, and biodiversity. A draft for the website is provided in Figure 6, which is inspired by similar approaches, such as Global Forest Watch [Glo20] and Hansen et al. [Han+13; Han+19]. There is also a complementary project, which monitors and analyzes the health of trees upon planting. However, this project is not focused on sustainability as it is oriented toward forestry clients to “monitor the growth, health and activities occurring in their forests [and plantations]” [Swi22].

Next to the website, other forms of awareness creation could be utilized. For example, a Twitter bot could post weekly, monthly or yearly updates on the tree stock development. However, this approach is limited because present data from the same region is multiple years apart. Thus, access to more satellite images is required for this idea. Nevertheless, frequent updates on the tree stock development are insightful because they enable to capture if a forest vanishes, allowing for fast counter-measures before too much forest is lost.



Figure 6 Website Layout (licensed by CD:NGI [Chi22])

### 3 Satellite Data and Project Feasibility

The first objective mentioned in Chapter 1 is the transferability of Brandt et al.’s project to this one. One key factor in the transferability analysis is the technical structure of the data because deep learning heavily relies on the quality of the data. So a low image resolution can only produce low-quality results, which can be summarized as “garbage in, garbage out” [Gér19, p. 55]. Another factor is the transferability of the imagery’s content because the examined type of vegetation affects the model’s performance. Thus, these two factors are considered in the transferability analysis.

According to Brandt et al., it is possible to create “excellent models for detecting isolated trees over large areas” [Bra+20, p. 81] with a deep learning approach combined with very high-spatial-resolution satellite image data. Both Brandt et al. and this project use satellite images with submeter resolution [Bra+20, p. 78]. This project has access to images with an extremely high resolution of 23 cm to 50 cm per pixel, which exceeds the 50 cm per pixel resolution from Brandt et al.

Moreover, the satellite data from this project offers the standard three red-green-blue bands. In contrast, the satellite data from Brandt et al. has two bands, namely a normalized difference vegetation index (NDVI) and a black and white panchromatic band [Bra+20, p. 79].

The NDVI “quantifies vegetation by measuring the difference between near-infrared (NIR) (which the vegetation strongly reflects) and red light (which the vegetation absorbs/has a low reflectance)” [GIS21; Dut+21, p. 329-342]. Formula 3.1 outlines that the NDVI is determined by dividing the difference of NIR and red light’s reflectances  $\rho$  by their addition. Hence, the NDVI band is the most reliable for capturing trees since it is suited for that purpose [Swi22; GIS21].

$$NDVI = \frac{\rho_{NIR} - \rho_{red}}{\rho_{NIR} + \rho_{red}} \quad (3.1)$$

#### 3.1 Calculation of NDVI [GIS21; Cam17; Nik20]

The chlorophyll absorbs red and blue light and reflects other wavelengths such as green light and NIR more strongly [GIS21; Ear21]. Therefore, the NDVI has a more accurate detection of a tree’s healthiness level and can more easily differentiate between vegetation and non-vegetation [Ear19; Nik20]. As an example, Swift Geospatial Solutions evaluates and monitors the healthiness level of plants on a plantation and presents the NDVI band as essential because it makes trees stand out significantly

more even for the human eye and makes the differentiation between trees and their shadows more clearly (see Figure 7) [Swi22].

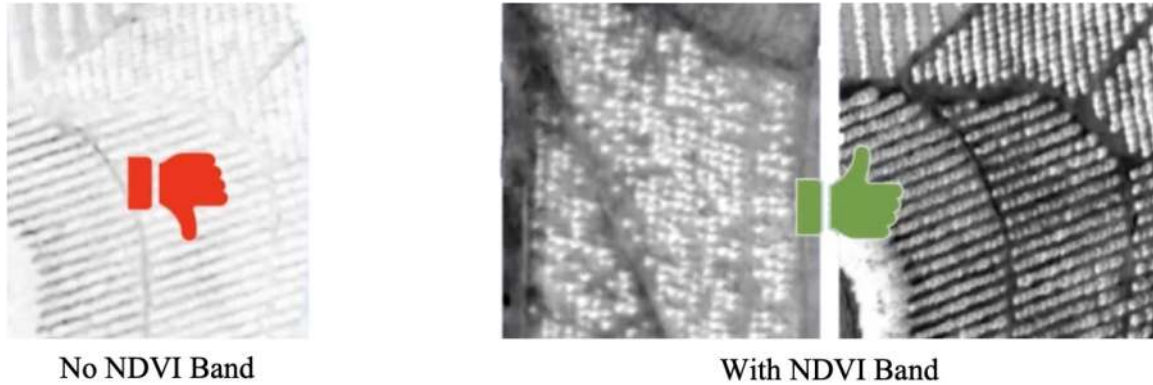


Figure 7 Vegetation Detection With NDVI [Swi22]

As mentioned before, the satellite data from this project does not have the benefit of a NIR or NDVI channel. Hence, the transferability of Brandt et al. and the detection of trees are technically limited.

The second factor of transferability is the degree of similarity in landscape characteristics. This implies that the difficulty in analyzing the present vegetation has a significant effect on the performance of the deep learning model. E. g., if the trees grow very small in size and with few leaves, it is challenging for the model to detect it because the trees do not stand out from the surroundings. This is supported by Brandt et al., who point out that the “transferability of the model across regions can be low” [Bra+20, p.81]. Consequently, it is paramount to understand to what degree the nature of the landscapes differ from one another and to what degree the trees stand out.

Sahel’s landscape from Brandt et al.’s project is covered in drylands, deserts, and semi-arid grassland, where low-growing grass, thorn scrubs, and tall trees are predominant [Bra+20; SA04; Edi22; Ler22]. In comparison, the vegetation in the inspected area of this project is characterized by isolated thorn trees and bush plains because the satellite images cover highveld and bushveld as outlined in Chapter 2.1 [Nat15]. Therefore, a sufficient number of visible trees are available, which build the foundation for the deep learning model [Sou14].

Concluding, the slight deviation in landscape characteristics and the difference in the data structure leads to a low to medium transferability from Brandt et al. to this project. Nevertheless, the realization of a functioning implementation is not ruled out.



## 4 Technological Approach

### 4.1 Classification of the Task in Computer Vision Context

The project task must be defined and understood from a technological standpoint to evaluate suitable technological approaches and assess their feasibility. From the technological perspective, this project can be categorized in the field of computer vision, which “enables computers and systems to derive meaningful information from digital images, videos and other visual inputs” [IBM20].

One way of analyzing digital images of videos is semantic segmentation, which associates each pixel of an image with a label [Nbr20; LJY17; Gér19, p. 648 ff.]. As visualized in Figure 5, the project performs that classification of a pixel with the label tree or non-tree. In semantic segmentation, there is no differentiation of instances. So all trees have the same label. If instances should be distinguished, it becomes instance segmentation. In this case, each tree (element of the same class) would receive a unique label (see Figure 8) [Nbr20; LJY17; Gér19, p. 324]. The task of classifying and localizing multiple objects is called object detection (see Figure 8) [Gér19, p. 640]. The latter two approaches are not relevant for this project because the precise localization is not the focus of this project and the output structure of the semantic segmentation offers more detailed and important information for the analysis of trees, such as the total tree coverage or above-ground biomass.

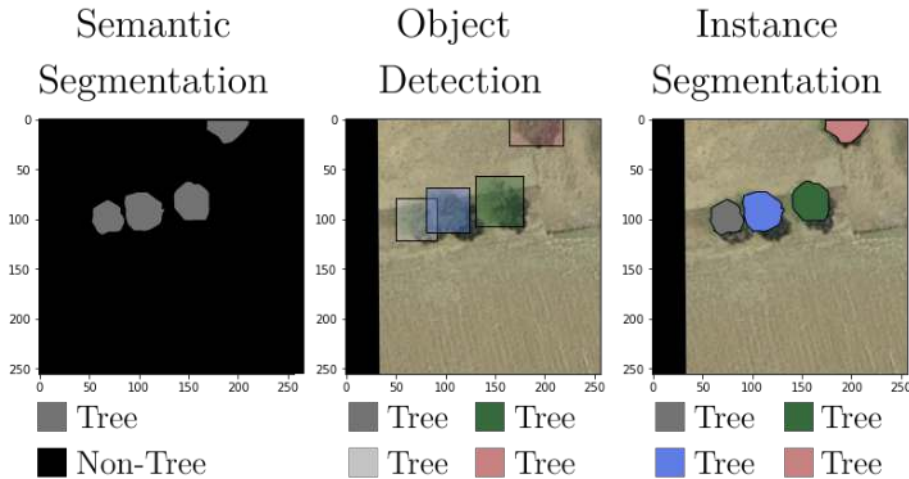


Figure 8 Computer Vision Tasks (adjusted from Li, Johnson, and Yeung [LJY17] and licensed by CD:NGI [Chi22])

## 4.2 Artificial Intelligence

The project is realized with a deep learning (DL) approach, which is classified as a subfield of machine learning (ML) and artificial intelligence (AI) (see Figure 10). Therefore, the overriding research fields are explored.

Artificial intelligence is a polarizing topic as it is widely discussed in the media and appears to be a promising technology for solving complex problems more efficiently. For example, it enables the automation of tasks performed by humans and hence reduces costs. The definition of artificial intelligence by its so-called father McCarthy is “the science and engineering of making intelligent machines” [Sci22a; McC07; Art20]. New techniques and approaches to solve problems with artificial intelligence are constantly developed. Hence, artificial intelligence evolves quickly and is not static, which requires a contemporary definition [CG22].

According to Ermakova and Frolova, “to date, no generally accepted definition of artificial intelligence has been created” [EF22]. Nevertheless, IBM explains that “artificial intelligence leverages computers and machines to mimic the problem-solving and decision-making capabilities of the human mind” [IBM20]. Similar definitions are used by encyclopedias, such as Encyclopedia Britannica, which defines artificial intelligence as “the ability of a computer [...] to do tasks that are usually done by humans because they require human intelligence and discernment” [Cop20].

## 4.3 Machine Learning

### 4.3.1 Definition

An early definition of machine learning portrays it as a field of study that gives “computers the ability to learn without being explicitly programmed” [Sam59; AK15; KS17]. Since then, machine learning has significantly changed and hence newer definitions are more fitting and apply the mutually exclusive collectively exhaustive principle (see [Hac22; MBA21]). Géron defines machine learning as the “science (and art) of programming computers so they can learn from data” [Gér19, p. 22]. Mitchell is more specific by outlining that it is a “(computer program’s ability) to learn from experience  $E$  with respect to some task  $T$  and some performance measure  $P$ , if its performance on  $T$ , as measured by  $P$ , improves with experience  $E$ ” [Mit97].

By comparing the definition of artificial intelligence and machine learning, it crystallizes that machine learning is a branch and subfield of artificial intelligence. It builds upon statistical instruments “to build systems that have the ability to automatically learn and improve from experiences without being explicitly programmed”



[Roy20]. Hence, machine learning is powered by statistics, algebra and calculus [DFO21; Cha21; CG22]. It is applied to problems, which can also be solved by heuristic and analytical approaches, such as classification, prediction, clustering, association analysis, dimension reduction, and state machine definition.

### 4.3.2 Distinction from Heuristic and Analytical Approaches

These statistical problems can also be solved manually by a human with long-established heuristic and analytical solutions. For instance, Thomas Bayes coined classification in the 18<sup>th</sup> century [CG22]. Thus, it is essential to understand when it shifts from heuristic or analytical solutions to machine learning.

This is the case when a machine demonstrates that it has learned. For example, it adjusts and saves model parameters to predict or classify more accurately [CG22]. Thus, future decisions are more accurate, e. g. identifying a class for land use in a region [CG22]. In clustering, the machine saves cluster characteristics, such as centroids to aid in assigning an observation to a cluster [CG22]. Similarly, association rules are identified to determine future associations and interdependencies of external events and triggered responses to improve the state machine [CG22].

Thus, the differentiating factor of heuristic and analytical approaches compared to machine learning is that machines learn and identify patterns in data to solve problems more accurately and efficiently. Therefore, the machine's learning improves future classifications, predictions, clustering, association analysis, or state machine definitions.

### 4.3.3 Types Machine Learning Systems

Type of Learning	Type of Learning Category			
	Label-Based			Environment-Based
	Supervised	Semi-supervised	Unsupervised	Reinforcement
	Labeled Data	Partly Labeled Data	Unlabeled Data	Trial and Error
Addressed Problem	Prediction Classification		Clustering Association Analysis	
			State Machine	

Figure 9 Machine Learning Systems Overview

These mathematical tasks differ in the structure of their problems. Therefore, different approaches to machine learning are used to solve them. First, the possible range

of input and output data is set in prediction and classification problems [Moo+21, Fig. 1]. For example, in this project, pixels in a satellite image are classified as either tree or non-tree. Therefore only the range of tree or non-tree can be predicted. This type of machine learning is called **supervised learning** because the desired solution, called labels, is part of the data set on which the model is trained [Gér19, p. 29].

Second, in clustering problems, the resulting clusters are unknown before the clustering is performed. So the model “group[s] and interpret[s] data based only on input data” [Moo+21, Fig. 1]. Similarly, the associations are not defined before the association analysis is completed. Because the training data is unlabeled, the machine learns without a human supervisor [Gér19, p. 31]. Therefore, this subfield of machine learning is called **unsupervised learning**.

Third, there can also a mixture of labeled and unlabeled data. Thus this type of learning system is called **semi-supervised learning**. So the system can progress without labeled data but needs it to complete the learning [Gér19, p. 37].

Fourth, state machines capture state changes by defining all possible finite numbers of states and “based on the current state and a given input the machine performs state transitions and produces outputs” [Ite22]. Similarly, a machine can also capture state changes and learn from interactions with the environment through trial and error. Rewards and punishments reinforce a specific successful behavior [Moo+21, Fig. 1]. Therefore, it is called **reinforcement learning**. It is defined as a “learning system (agent), [which] can observe the environment, select and perform actions, and get rewards or penalties in return. It must then learn by itself what [...] the best strategy (policy [= action in a given situation]) [is], to get the most reward over time [Gér19, p. 38].

#### 4.4 Deep Learning and Convolutional Neural Networks

Artificial neural networks (ANN) is a machine learning model consisting of one or more tunable artificial neurons inspired by the networks of biological neurons found in the human brain [Kaz18; Gér19, p. 364]. These neurons can be connected to pass numerical signals from one unit to another and are arranged in a one or more layered network architecture [Kaz18; Gér19, p. 364].

As presented in Chapter 4.3.2, the learning is demonstrated in the model parameters. For instance, the net of neuron connections has feature weights, which express what logical connections have been made. I. e., to what extent features, such as the NDVI value, affect the output (see Chapter 3). So a high NDVI value (feature) could

significantly impact the classification of a tree or non-tree (output) by having a high feature weight.

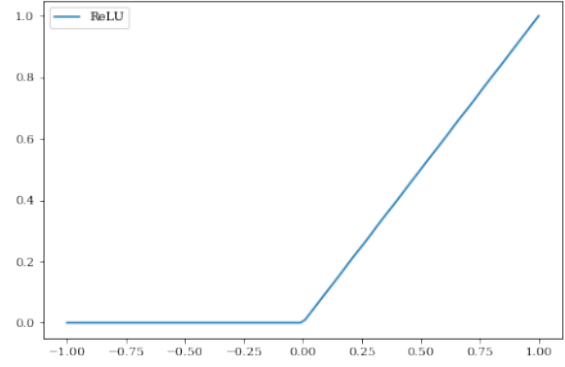
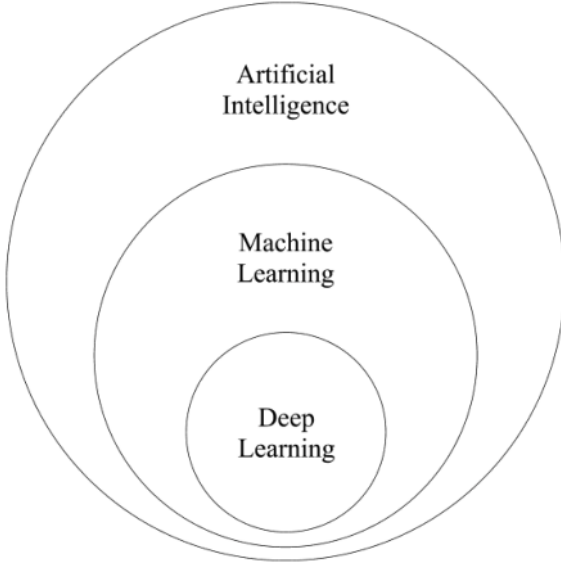
The simplest ANN architecture is the Perceptron, invented in 1957 by Frank Rosenblatt [Gér19, p. 370]. It can be used for basic linear binary classification because it takes a linear combination of inputs with corresponding feature weights to determine the confidence of a class prediction with a step function. If that confidence exceeds the threshold, it outputs the class. Continuing from the last example, an input NDVI value of 0,9 is evaluated considering a high feature weight of 0,8. The result of 0,72 surpasses the threshold of 0,5 to output one, which is interpreted as the class tree.

Many layers enable to grasp more complex logical connections, which makes it versatile and powerful enough to tackle complicated machine learning tasks, such as speech recognition in Apple's Siri [Gér19, p. 364].

The hierarchy of multiple deep layers forms abstractions and representations, which may be summarized as the learned knowledge of the deep learning approach [Zha+18]. Deep learning is characterized by neural networks that have more than three layers [IBM20]. Thus, it is a subset of machine learning as visualized in Figure 10 [Cha+20]. Moreover, deep learning is scalable and less dependent on human intervention to learn than shallow machine learning [IBM20; Gér19, p. 364]. The deep neural net is able to extract and learn the features by itself instead of being dependent on manual human input [Pai20].

Convolutional neural networks (CNN) are ANN capturing spatial information, which means the network understands the pixel arrangement and their relationship [Pai20; ON15, p. 1]. Consequently, it is suited, applied to process and analyze images or videos [Pai20; ON15; LSD15, p. 1]. The relevance of technical approaches using CNN crystallizes in a study from McKinney et al. In practice, a CNN outperformed six radiologists in breast cancer prediction, which is expressed by a greater absolute margin of 11,5% in to the average human experts concerning the area under the receiver operating characteristic curve [McK+20].

CNN are neural networks with at least one convolution layer [Kaz18]. The CNN architecture can be divided into feature learning and classification layers (see Figure 12). Usually, feature learning is performed by multiple successive convolutional and pooling layers. The convolutional layers apply a non-linear activation function, such as the rectified linear activation unit (ReLU) to prevent the optimization gradient from vanishing (see Figure 12). ReLU is a simple function solving the vanishing gradient problem because it returns the input if it is positive and zero in any other case (see Formula 11) [AMA17].



$$f(x) = \max(0, x) \quad (4.1)$$

Figure 10 Relationship of AI, ML and DL [IBM20]

Figure 11 Rectified Linear Activation Unit (ReLU) [AMA17]

The pooling layers, such as max pooling or strided convolution abstract from the image by applying a filter (kernel), which summarizes the features of the filter's region [Kho21; Gér19, p. 649]. The feature learning result is used to perform the classification with a few fully connected layers (see Figure 12) [Mat21]. One convolutional unit receives multiple adjacent units from the previous layer, which share their weights [Kaz18]. The neighboring inputs enable to process local information, which is enabled by a filter moving across the data. For example, the filter captures the surrounding pixels in a specific area of an image [Kaz18].

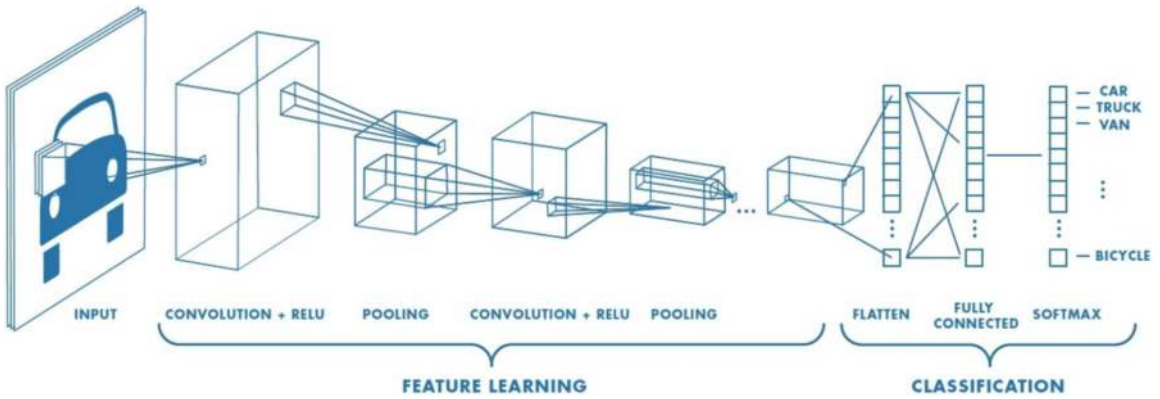


Figure 12 CNN Architecture [Sah18]

#### 4.5 Fully Convolutional Network

There are various architectures utilizing a CNN approach, such as fully convolutional networks (FCN). FCN consists of convolutional and pooling layers and are applied

to semantic segmentation and object detection problems [Dwi19; Mat21]. It does not have dense or fully connected layers because they are interchangeable with convolutional layers, if the convolutional filter size and input feature map size are identical [Dwi19; Mat21; Arn+18, p. 6].

The network layers are organized in a downsampling and upsampling path. The former path consists of a sequence of convolution and pooling layers called the encoder [Mat21; ML21]. This downsampling path aims to extract contextual information (features) by decreasing the image resolution through filters [SLD15; ML21; Mat21]. This spatial contraction helps to process large images and provides more details about what is portrayed [RFB15b]. However, spatial information is lost on that path resulting in lower localization [RFB15b]. This abstraction from the image is performed by filters, such as max pooling or strided convolution (see Chapter 4.4).

The upsampling path enables precise localization by restoring the condensed feature map and employing concatenations, also called skip connections [SLD15; ML21; RFB15b; Arc19]. This expansion path is called the decoder, which consists of a sequence of upsampling convolutions (upconvolutions) and mentioned skip connections [Mat21; ML21]. The upconvolutions or also called transposed convolutions increase the resolution by unpooling with the nearest neighbor interpolation, bed of nails or max unpooling technique. The last technique reverts the effect of the max pooling layers in the downsampling path and therefore works as an upsampler (see Chapter 4.4 and Figure 13) [Zaf+18].

The concatenations bypass at least one layer as portrayed in gray in Figure 13 and combine the outputs of each downsampling step with the corresponding convolution inputs of the upsampling path [Dwi19; Mat21; Bra+20, p. 83]. Therefore, this architecture enables “the simultaneous processing of an input image at different spatial scales” [Bra+20, p. 83].

Thus, the contextual information from the shallow downsampling layers is merged with the spatial information from the deeper upsampling layers [Dwi19]. This “recover[s] the fine-grained spatial information lost in the downsampling path” [SLD15]. Skip connections are also utilized because the combination of appearance features in the lower layers and semantic features in the deeper layers enable a precise and accurate image segmentation [LSD15; SLD15].

## 4.6 U-Net Architecture

One variation of a FCN is the U-Net, which was initially built to find brain or lung tumors [Mat21; RFB15a; Nbr20]. Since its publication, it has been a state-of-the-art

approach to solving semantic segmentation problems exceeding medical application only [Mat21; RFB15a]. As mentioned previously, this project tackles a semantic segmentation, which is why the U-Net was evaluated as a suitable approach. Its architecture consists of an encoder (downsampling), a decoder (upsampling) and a bottleneck in between them [Kap21]. This structure forms the shape of the letter “U”, which is the origin of the architecture’s name (see Figure 13). In this project, the convolutions utilized batch normalization layers and ReLU as the activation function (see Chapter 4.4 and Figure 13).

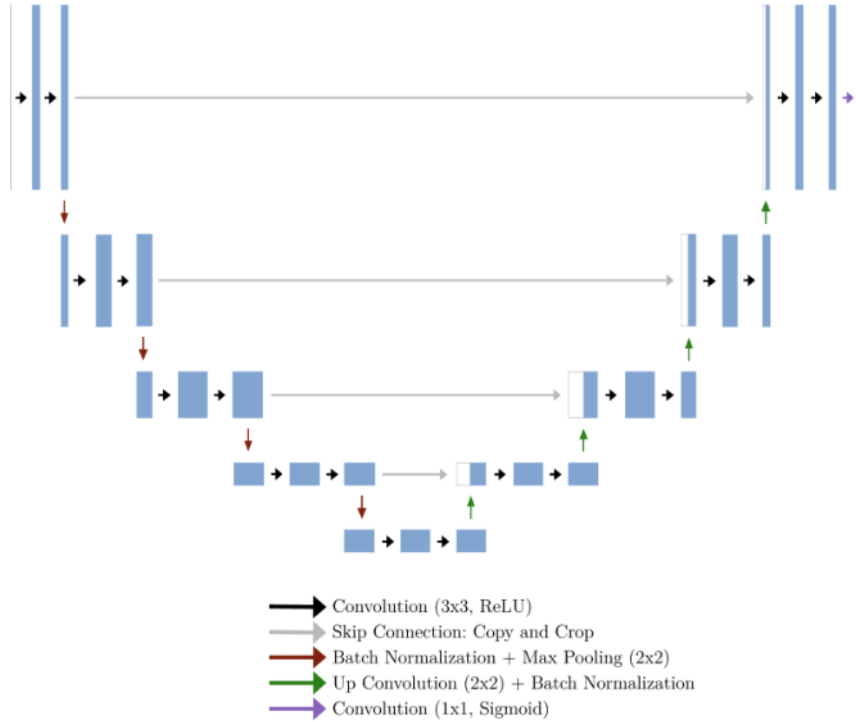


Figure 13 U-Net (adapted from Ronneberger, Fischer, and Brox [RFB15a])

In summary, machine learning comprises the field where machines learn from data with the basis of statistics, algebra and calculus to solve a range of problems, which are performed unsupervised, semi-supervised or supervised or with reinforcement. Deep learning is a subset of machine learning and is characterized by high network complexity, which is suitable for processing input with high media-richness, such as videos, images or audio. It may utilize convolutions in neural network architectures to capture spatial information. One CNN variant is the U-Net, which has a down-sampling and upsampling path with skip connections to enable fast and accurate predictions.

## 5 Implementation and Insights for Future Projects

### 5.1 Project Pipeline

The project is comprised of a “sequence of data processing components [...] called a data pipeline” [Gér19, p. 70] to facilitate and automate the usage of the deep learning algorithm for further usage. This enables automation and a simple prediction of other areas in the existing dataset.

Each step of the pipeline is realized in a separate and reusable component, which performs data manipulations and transformations to achieve its objective. This clear structure of meaningfully chained and orchestrated components makes the project code easier to understand (see Appendix 19). Components are self-contained and relatively independent from one another to ensure code maintainability [Gér19, p. 70].

Nevertheless, the output data from previous steps needs to be accessible through a data store (see Appendix 19), which is the only interface of the components [Gér19, p. 70]. This data store saves the intermediate results from the components. So in case of a component break down, the progress is saved [Gér19, p. 70].

The components are usually executed asynchronously and monitored to check if a component is broken [Gér19, p. 70]. The monitoring is performed through tests and sanity checks in the components. In case of a breakdown, the downstream components can often continue to run as they do not depend on the broken-down component [Gér19, p. 70].

The pipeline of this project is structured by the three states of new input (see Appendix 19). The first scenario is that there is no new input. So the pipeline does not perform any action. The second scenario is that new satellite data is added to the server. This input is imported and split into patches. The trained U-Net model is loaded and predicts based on the satellite patches. The predicted patches are put back together to form a segmentation of the whole image. The third and last scenario is the creation of new label input, which causes the execution of data preparation including the patch creation process and data splitting into training, validation and test dataset. Subsequently, the U-Net model is defined or loaded, trained and evaluated. Then the test data is predicted and the model performance is measured. After the pipeline is completed, it is restarted each day to check if any new input has been added.

Currently, the label data and their spatial extent must be uploaded to the server. Ideally, the labels could be stored in the PostGIS database connected to QGIS. A connection to that database could asynchronously trigger the pipeline’s training process. However, this exceeds the scope of this project. At the moment, there is no need for an automatic execution of the pipeline every day because the dataset is not extended. Nevertheless, a rerun trigger feature could be considered in projects with continuous data input (see Appendix 19).

## 5.2 Data Creation, Processing and Export

As mentioned in Chapter 4.3.3, this project solves a classification problem and accordingly falls under supervised learning and hence requires labels for the model’s training. In this project, these labels are created locally in QGIS, which is an open-source geographic information system application. It is suited for labeling as it allows for efficient viewing, editing, and analysis of geospatial data. These actions are not supported in other applications such as macOS Preview, which even crashes when visualizing satellite image files. This happens because it is not built to handle large geospatial files in a rasterized tag image file format (TIFF), taking up about two gigabytes on average in this project.

The complete labeling process is outlined in Appendix A.2 to facilitate the process in future projects. It is worth mentioning that the labeling in QGIS cannot be performed while specific programs are running because it may lead to temporary faulty visualization. As portrayed in Figure 14, this may cause a displacement and change in label layer size as the pink labels are zoomed out and moved to the upper left corner. This phenomenon occurred in the case of the video conference application Zoom and the vector graphics editor and prototyping tool Figma. The origin of this bug might be that Zoom and Figma try to utilize and lock the same resources as QGIS. For instance, Figma, Zoom and QGIS work closely with various layers. As a result, QGIS is prevented from displaying the layers correctly.

## 5.3 Data Preparation: Patch Creation and Augmentation

The data preparation and processing, including patch creation and data augmentation, posed some obstacles. Therefore, sharing the project’s insights in working with geospatial data can help and facilitate future projects. The patch creation posed a challenge as the creation of the correct image and mask patches were visualized differently by two libraries. Namely, in Rasterio, the patches were visualized correctly, while Matplotlib exhibited an unnatural behavior displaying the image multiple times (see Figure 15).





Figure 14 Misplaced and Zoomed Layer Bug in QGIS (licensed by CD:NGI [Chi22])

This phenomenon happened with the first and easiest method to create patches: patchify can also put the created patches back together so that the original image size is preserved (see patchify documentation [Pat22]). For future projects, it is still advisable to try out patchify due to its simplicity. Nevertheless, the exact source of the problem is unknown in this project. Hence, more control over the patch creation was necessary. Thus, the patch creation is implemented manually to ensure that the data is correct (see Appendix 3).

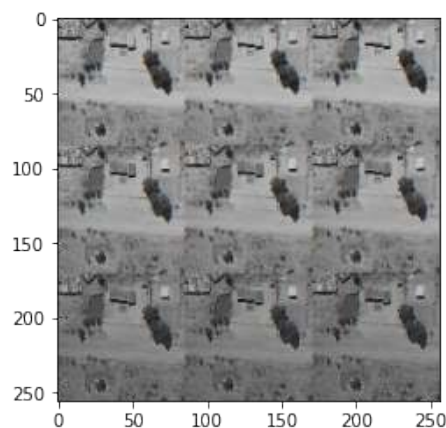


Figure 15 Patch with Duplicated Data (licensed by CD:NGI [Chi22])

The manual patch creation can be performed by a windowed reading approach, which utilizes coordinates. This enables an accurate patch creation without displacements of labels, which occurs if the coordinate window of the read satellite image and label coverage do not match. Nevertheless, the final version of the patch creation uses windowed reading with a step size to guarantee that each patch has the same size of  $256 \times 256$  pixels and an overlap of 32 pixels (see Appendix 3).

QGIS exports each label with a unique color value. Hence, the label data has to be corrected to values of zero or one before performing other data preparation actions, such as the color value scaling. This feature scaling is crucial for the model to perform well, which may be performed with the StandardScaler or MinMaxScaler from Scikit-learn (see documentations [Sci22b; Sci22c]) [Gér19, p. 108-109]. But in this project, it is sufficient to divide the color values by the maximum RGB color value of 255 because the satellite image color values were close to the minimum and maximum RGB color values.

Before training the model on the prepared data, it is helpful to have as much training data as possible so that the model can learn from more data. This is achieved by increasing the training dataset size by “generating many realistic variants of each training instance” [Gér19, p. 613]. The most realistic method is to resize, rotate or shift the image enforcing the model’s tolerance towards the object’s size, orientation and position [Gér19, p. 613].

This concept is summarized under the term data augmentation, which is performed in this project by rotating the patches by 90, 180 and 270 degrees. The vertical augmentation is implemented with slicing and the horizontal flip is achieved with the help of the library NumPy (see Appendix 4).

If the patches are flipped vertically with NumPy or TensorFlow, the image becomes blue (see Figure 16a and 16b). This occurred with TensorFlow’s RandomFlip layer, single image flipping with NumPy and TensorFlow and ImageDataGenerator from Keras. The color change might be the result of flipped color channels so that red is interpreted as blue and vice versa. Moreover, the patches became blurry after using TensorFlow and the corresponding labels were not augmented (see Figure 16b). The image may also be rotated by an angle to create more batches. With TensorFlow, it resulted in a distorted image, which suggests that the data structure was different from the one that was expected (see Figure 16c).

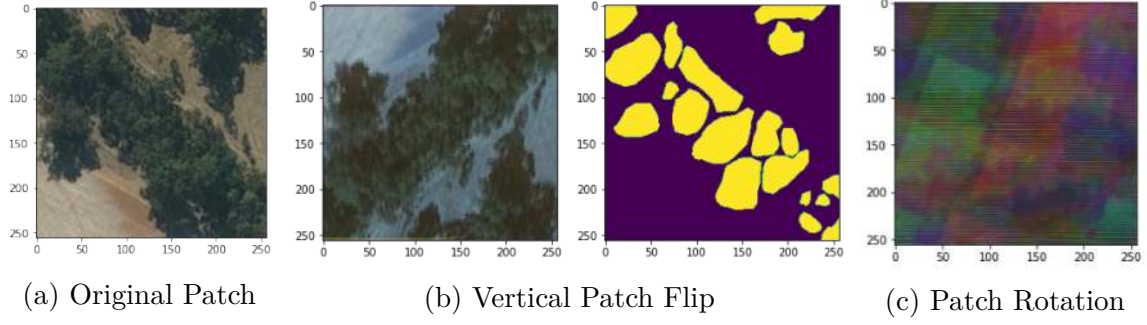
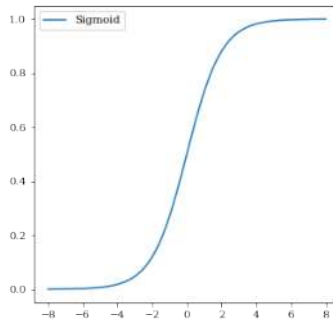


Figure 16 Faulty Data Augmentation in TensorFlow (licensed by CD:NGI [Chi22])

#### 5.4 U-Net Model and Training on GPU and Prediction

It is advisable to train the model on the graphics processing unit (GPU) instead of the default central processing unit (CPU) because it significantly increases training speed [DAT19; Sha18]. CUDA enables to push resources to the GPU to utilize it [Ten22]. The U-Net model is defined as a function similar to Mader [Mad18] and Kariryaa [Kar20]. Since its functional, CUDA is unable to move it to the GPU. Consequently, the model is redefined as a class, which enables it to be moved to the GPU. However, this forces a manual implementation of the training method including the forward pass method, which requires another restructuring of the model resources [Hel21]. The effort for these multiple adjustments to the model's structure and definition exceeded the expectations and planned time window. Hence, the model training in this project was performed on CPU, which took two days and four hours to complete. The final layer of the U-Net model utilizes a sigmoid activation function, which outputs the confidence for the class tree (confidence  $\in [0; 1]$ , see Figure 5.1). However, the model only puts out values between 0,8395 and one. Hence, it has a strong tendency towards trees.



$$S(x) = \frac{1}{1 + \exp(-x)} \quad (5.1)$$

##### 5.1 Sigmoid Activation Function

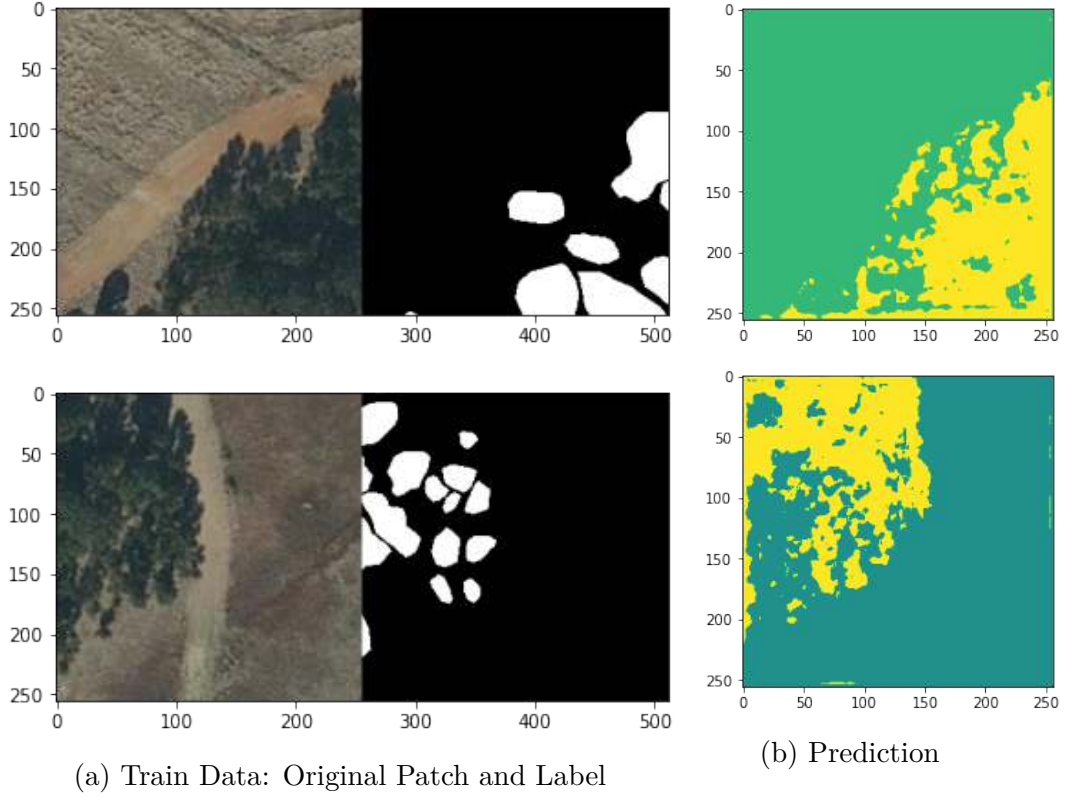


Figure 17 Output Prediction of Test Patches (licensed by CD:NGI [Chi22])

Nevertheless, the prediction is relatively accurate when visualized because of the scaling of color values, which enables a visual analysis of the data (see Figure 17). This is also reflected in the evaluation, where the confidence is interpreted as a tree class. This interpretation was performed by scaling the confidences using Formula 5.2 and subsequently evaluating all confidences below or equal to 0,5 as class non-tree and all other confidences as class tree.

$$Scaled\ Data = \frac{data - \min(data)}{\max(data) - \min(data)} \quad (5.2)$$

## 5.2 Confidences Scaled From Zero to One (inspired by Scikit-learn [Sci22b])

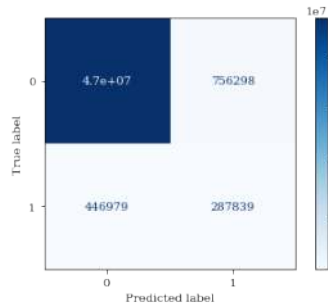


Figure 18 Confusion Matrix With Scaled Confidences

The evaluation of the training data resulted in an accuracy of 97,50%. Because of the given class imbalance, it is not a representative key figure [Olu22]. Thus, the uneven class distribution has to be considered in the metric, which is the case with the F1 score, the area under the receiver operating characteristic curve (ROC AUC) and balanced accuracy. The latter assesses the model's performance by calculating the arithmetic mean of its sensitivity and specificity (see Formula 5.7).

Sensitivity and specificity represent the true-positive and false-positive rate (see Formula 5.5 and 5.6), which is plotted in the ROC. In this project, the balanced accuracy and ROC AUC are 68,79% and the dice or F1 score resulted in 32,36%. The precision and recall are 27,57% and 39,17% (see Figure 18, Formula 5.4 and 5.5).

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad (5.3)$$

$$Precision = \frac{TP}{TP + FP} \quad (5.4)$$

$$Sensitivity = Recall = TP - Rate = \frac{TP}{TP + FN} \quad (5.5)$$

$$Specificity = FP - Rate = \frac{FP}{FP + TN} \quad (5.6)$$

$$Balanced\ Accuracy = \frac{Sensitivity + Specificity}{2} \quad (5.7)$$

$$F1\ or\ Dice\ Score = \frac{2 \times Precision \times Recall}{Precision + Recall} \quad (5.8)$$

5.3 Performance Metrics (True-Positive (TP), False-Positive (FP), True-Negative (TN), False-Negative (FN)) [Mat21; Goo20; Olu22; Hal20]

## 6 Limitations and Further Improvement

As touched on in Chapter 5.3 and 5.4, there have been various unforeseeable problems in the course of this project with the data and GPU utilization, which had to be overcome. Further challenges comprise that working with the server was not ideal. For example, in the week before the project deadline, the access token expired several times in two days so that the server was only temporarily accessible in the crucial last week. This inefficiency of asking personnel for a new token was present throughout the project. Moreover, the document and folder structure on the server was altered during the project and even the utilized satellite images were temporarily taken down.

Another challenge is the familiarization with QGIS, which was overcome. For instance, it took a significant amount of time to find out the correct export procedure of the labels to achieve the correct data format. In the early stages of the project, another export procedure was utilized. The work of two weeks with that procedure was lost and insignificant due to the incorrect data structure. Thus, the data processing and preparation required more time than expected. The number and significance of contingencies took up more time than accounted for in the project buffer.

These challenges leave room for further improvement of the deep learning approach. For instance, the gaps between close trees can be penalized with extra weights to crystallize borders of trees so that each tree is semantically segmented, which is currently not achieved (see Figure 17).

Moreover, the model's accuracy could be increased by creating more labels, which can be facilitated with point supervision (see Bearman et al. [Bea+16]). Another way of improving the model is transfer learning, where the current model can learn from similar models, which are already trained. For instance, the model parameters from Brandt et al. [Bra+20] could be utilized. Though, it is worth noting that there is a difference in the data structure, which limits the transfer of the learning process (see Chapter 2.3). Moreover, the model could be improved with hyperparameter grid search and fine tuning, which finds better values for hyperparameters, such as the learning rate and batch size.

# A Appendix

## A.1 Pipeline

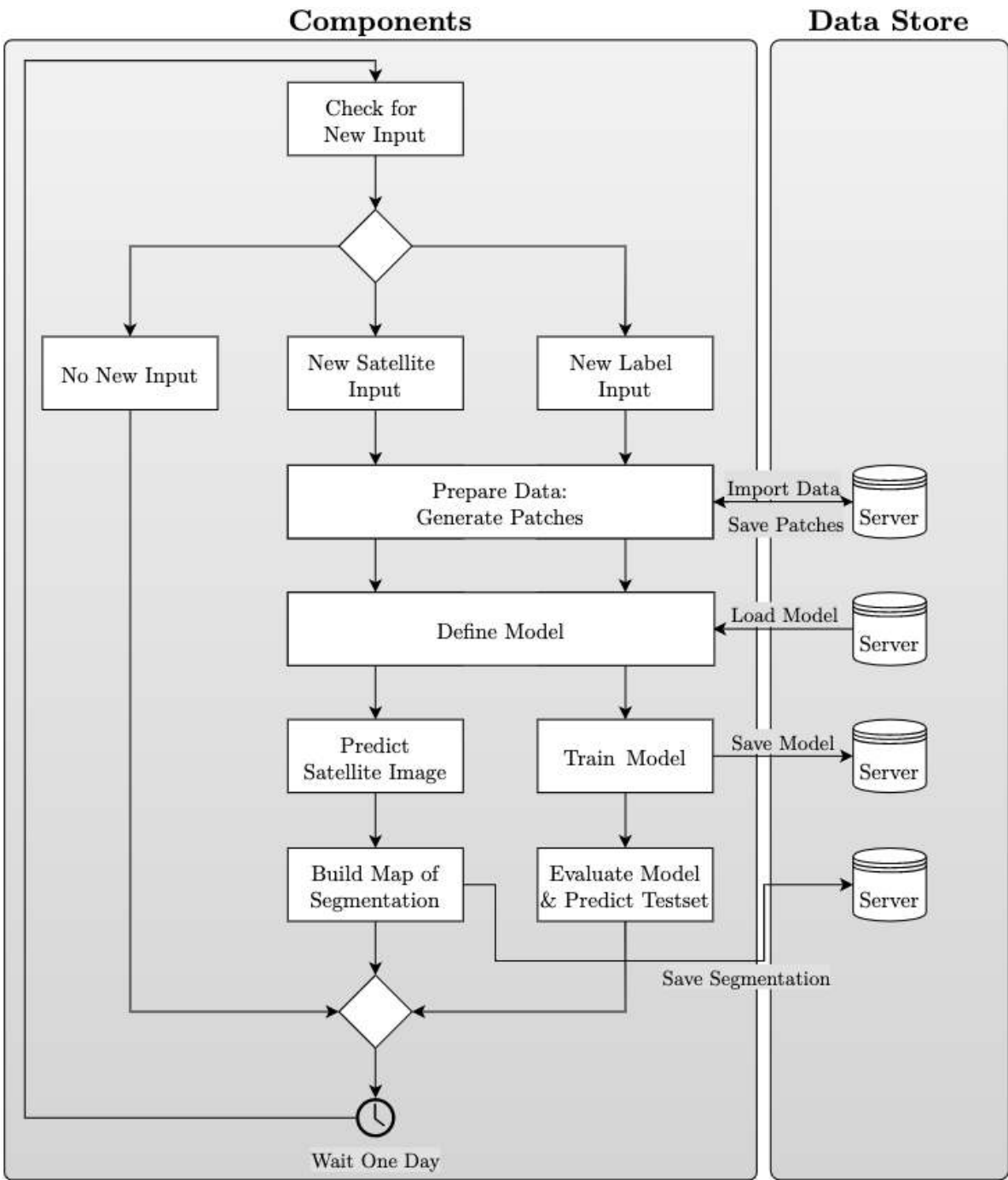



Figure 19 Project Pipeline

## A.2 QGIS Label Creation

The labeling preparation comprises a download and import of the satellite images into QGIS. A temporary scratch layer is created for the data creation by clicking  in the menu panel. Next, the labels are created by clicking on the outline of the label (here: a single tree) and confirming its completion by right-clicking. The commands are similar to other applications, for instance one can move around by clicking the space bar.

Labeling checkpoints can be stored in shapefiles with georeferencing (see Figure 20). However, it is inexpedient to label in a shapefile because QGIS asks for a field identification after each label is confirmed. This can be time-consuming when thousands of labels are created. Hence, after loading the checkpoint shapefile, the labels are copied and pasted to a temporary scratch layer.

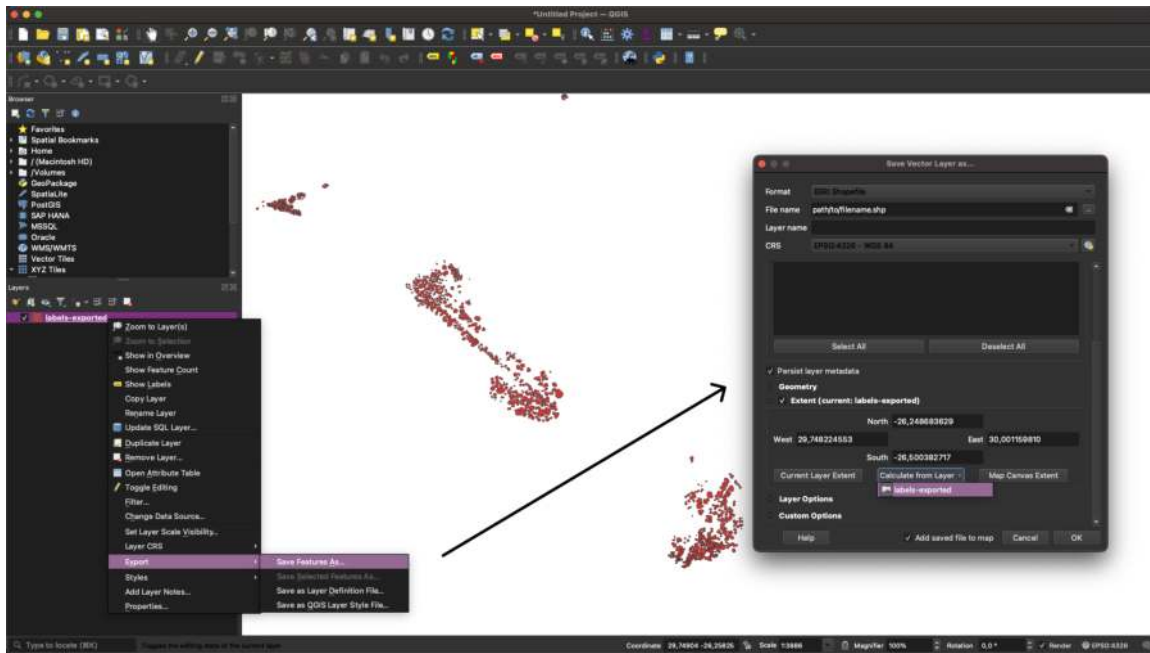



Figure 20 Save Labels to Georeferenced Shapefile

The export in QGIS may result in an error when the subsequent export procedure is not followed correctly. First, the last checkpoint of labels are loaded as a shapefile, which is not a temporary layer. Second, navigate from the menu bar as such: **Raster** > **Conversion** > **Rasterize (Vector to Raster)...** Third, the export parameters are defined. Select the label file as the **input layer** and use the **fid** field as a burn-in value. Since the output should include the geospatial information, the georeferenced units are selected as the output raster size unit. Moreover, initialize the horizontal and vertical resolution using the ones from the satellite image. Right-click on the satellite image layer and navigate **Properties...** > **Information** to find out the resolution



under Pixel Size. The first value is the horizontal and the latter is the vertical resolution. Next, assign the output extent of the satellite image by clicking on  > Calculate from Layer and confirm with Run (see Figure 21). These settings rasterize the labels so that the bounds and resolution align with the satellite image.

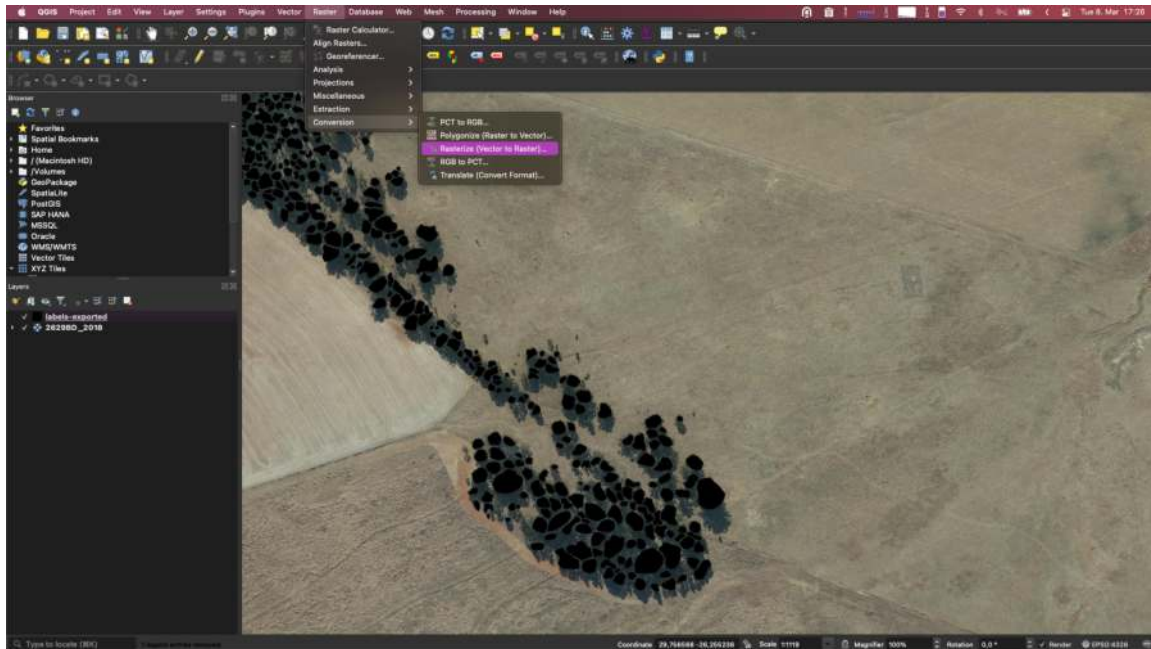


Figure 21 1. QGIS Rasterization of Labels: Open Conversion Settings (licensed by CD:NGI [Chi22])

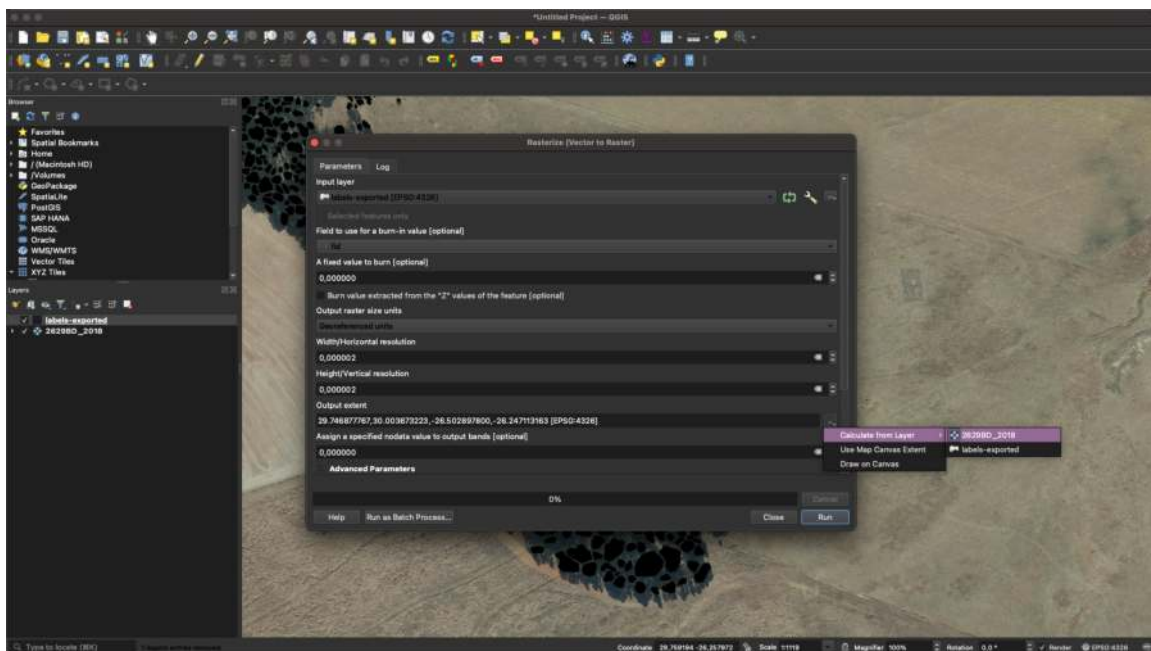


Figure 22 2.1 QGIS Rasterization of Labels: Set Conversion Settings (licensed by CD:NGI [Chi22])

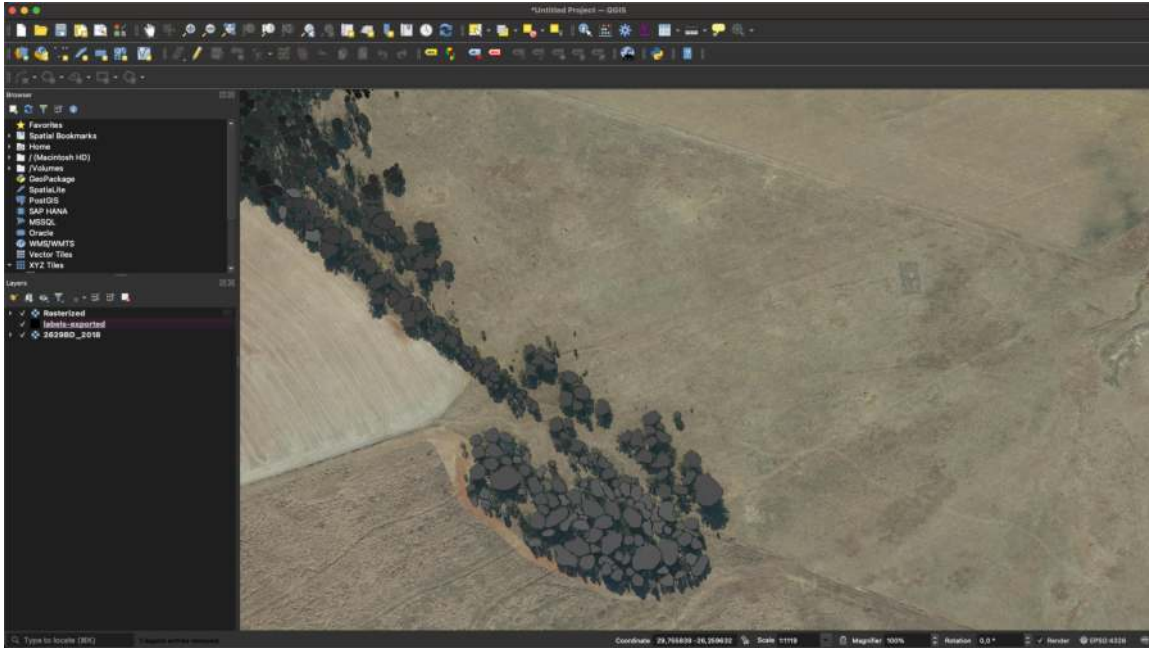


Figure 23 2.2 QGIS Rasterization of Labels: Result of Conversion (licensed by CD:NGI [Chi22])

Fourth, the labels vary in color depending on their field ID. This inconsistency can be corrected within QGIS by setting the raster layer's color value minimum and maximum to 255 (black). These settings can be found by right-clicking the layer and selecting **Properties... > Symbology** (see Figure 24). However, this does not make the labels uniform during the export. Therefore the color change has to be programmed (see Code 3, l. 36). This conformity and consistency in data from step three and four are crucial for enabling better predictions because the model performs better when the input and output of the training data are uniform [Gér19, p. 55].

The rasterization results in a temporary layer, which is exported by right-clicking the layer and selecting **Export > Save As...**. Next, in the subsequent export settings define the output filepath and select the layer extent by selecting the satellite image when clicking on calculate from layer. Lastly, select **High Compression** under **Profile** to keep the file size relatively low and prevent QGIS from saving redundant information. In this project this compression reduced the output file size from more than 45 GB to under 5 GB.

It should be noted that QGIS crops or rounds the coordinate values of the image bounding box are at the ninth decimal point. This can be ignored because the patches are not affected because of a windowed reading approach. Thus, the patches are not moved.

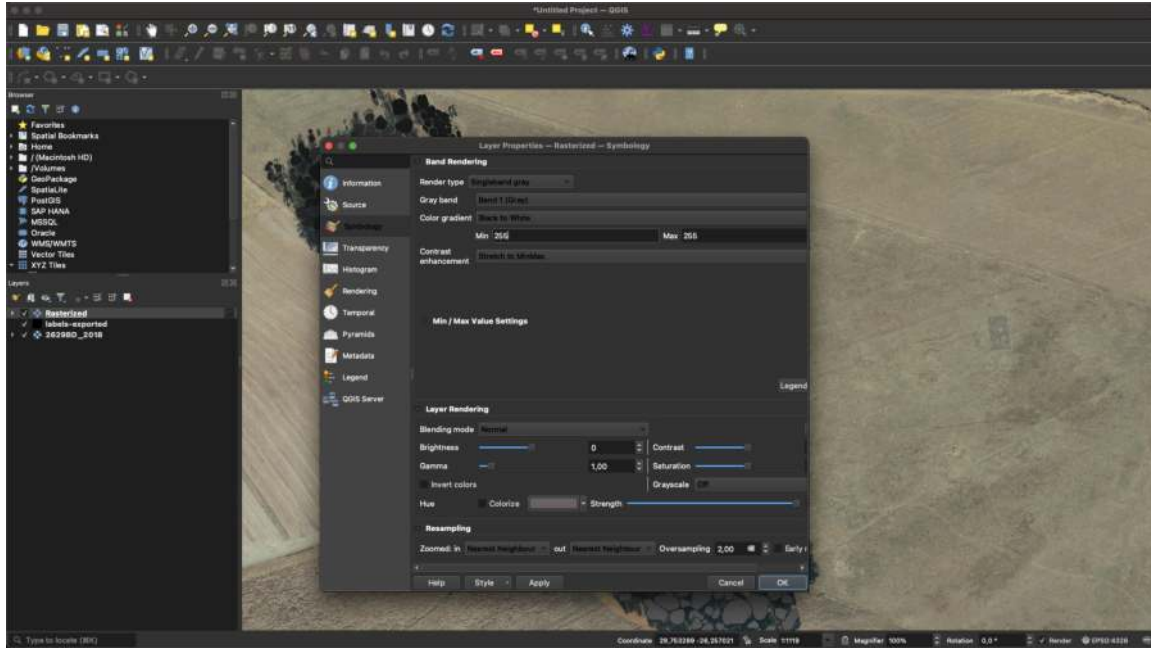


Figure 24 3. QGIS Rasterization of Labels: Change Label Polygon Color (licensed by CD:NGI [Chi22])

Furthermore, the export also has a very slight effect in resolution. Nevertheless, this potential inconsistency could be averted by exporting the satellite images too. So the export changes from QGIS are applied to both satellite images and labels. However, this would hinder the automation of the pipeline because the images would have to be manually downloaded, imported into and exported from QGIS and finally uploaded to the server. An alternative would be to change the resolution of the images with GDAL (Geospatial Data Abstraction Library), which was not tested in this project because the difference in resolution was very small (less than 0,1 nm).

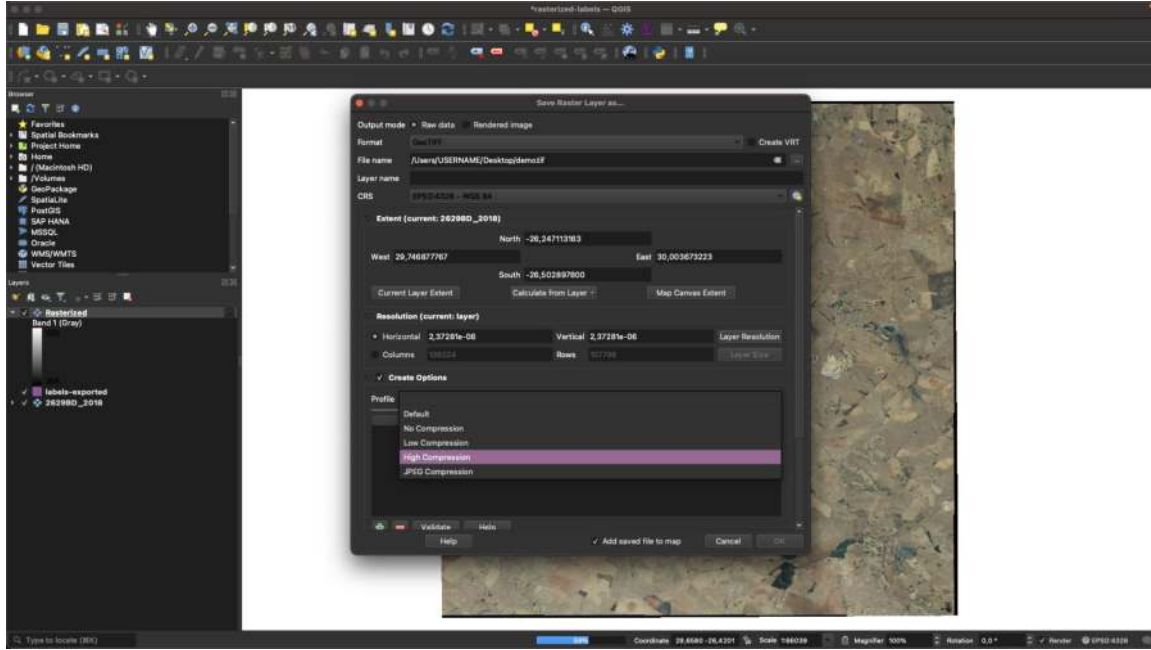


Figure 25 4. QGIS Export With Compression (licensed by CD:NGI [Chi22])

### A.3 Project Code

Data labelling is performed in QGIS (3.22.1) and the deep learning implementation is written in Python (3.9.5) with the help of various libraries, such as TensorFlow (2.6.1) and Keras (2.4.0). The source code can be retrieved from GitHub.

---

**Algorithm 1** Rasterio Basic Commands

---

```

1  import rasterio
2  from rasterio.plot import show
3  from rasterio.windows import from_bounds, Window
4  from config import UNetTraining
5  config = UNetTraining.Configuration()
6  img = rasterio.open(config.filepath_label_compressed)
7  # 1. inspect image
8  # get bounding box coordinates, resolution and size of image
9  img_info = (img.bounds, img.res, img.width, img.height)
10 print(img_info)
11 # 2. windowed reading
12 start_x = start_y = steps_x = steps_y = 512
13 img_cutout_steps = img.read(None,
14     window=Window(start_x,start_y,steps_x,steps_y))
15 show(img_cutout_steps)
16 west, south = 29.74801671786931, -26.24955085491147
17 east, north = 29.74923159879657, -26.248335973984698
18 img_cutout_coord = img.read(None,
19     window=from_bounds(west, south, east, north, img.transform))
20 show(img_cutout_coord)

```

---

**Algorithm 2** Satellite Coverage and Range of Resolution

---

```

1  from os import listdir
2  from os.path import isfile, join
3  import rasterio
4
5  path_satellite = "/home/jovyan/work/satellite_data/"
6  all_tif = [f for f in listdir(path_satellite)
7             if isfile(join(path_satellite, f)) and f.endswith('.tif')]
8  # furthest west, furthest east = min_x, max_x
9  # furthest north, furthest south = max_y, min_y
10 # initialize with first window
11 filepath = join(path_satellite, all_tif[0])
12 src_img = rasterio.open(filepath)
13 (min_x, min_y, max_x, max_y) = src_img.bounds
14 west_img = south_img = east_img = north_img = all_tif[0]
15 xRes, yRes = src_img.res
16 min_res, max_res = xRes, xRes
17 # update coordinate extend, if cur extend exceeds the window
18 for file in all_tif:
19     filepath = join(path_satellite, file)
20     src_img = rasterio.open(filepath)
21     # bounds
22     (west, south, east, north) = src_img.bounds
23     if west < min_x: min_x = west; west_img = file
24     if east > max_x: max_x = east; east_img = file
25     if south < min_y: min_y = south; south_img = file
26     if north > max_y: max_y = north; north_img = file
27     # resolution
28     xRes, yRes = src_img.res
29     if min_res > xRes: min_res = xRes; min_res_img = file
30     elif max_res < xRes: max_res = xRes; max_res_img = file
31
32 satellite_coverage = (min_x, min_y, max_x, max_y)
33
34 def get_res(filename):
35     src_img = rasterio.open(path_satellite + filename)
36     return src_img.res
37
38 print("Satellite Coverage:", satellite_coverage)
39 print("Edge Files:", west_img, south_img, east_img, north_img)
40 print(f"Resolution Range: {min_res_img}: {get_res(min_res_img)}")
41 print(f"{max_res_img}: {get_res(max_res_img)}")

```

---



---

**Algorithm 3** Patch Creation (1/2)

---

```

1  from enum import Enum
2  import numpy as np
3  from matplotlib import pyplot as plt
4  from osgeo import gdal
5  from config import config
6  config = config.Configuration()
7
8  class Channel(Enum):
9      GRAYSCALE = 0
10     RED = 0
11     GREEN = 1
12     BLUE = 2
13
14  class TiffLoader:
15     def __init__(self, path):
16         self._dataset = gdal.Open(path, gdal.GA_ReadOnly)
17         self._bands = [self._dataset.GetRasterBand(
18             x) for x in range(1, self._dataset.RasterCount + 1)]
19
20     def load_rgb(self, x, y, xs, ys):
21         r = self.load(Channel.RED, x, y, xs, ys)
22         g = self.load(Channel.GREEN, x, y, xs, ys)
23         b = self.load(Channel.BLUE, x, y, xs, ys)
24         return np.dstack((r, g, b))
25
26     def load(self, channel: Channel, x, y, xs, ys):
27         return self._bands[channel.value].ReadAsArray(
28             x, y, win_xsize=xs, win_ysize=ys)
29
30     def gray_to_rgb(grayscale_img):
31         assert len(grayscale_img.shape) == 2
32         return np.stack((corrected_labels(grayscale_img),) * 3,
33             axis=-1).astype(np.uint8)
34
35     def corrected_labels(labels):
36         labels = np.where(labels != 0, 255, 0)
37         return labels
38
39     def show(data, label):
40         plt.imshow(np.hstack((data, gray_to_rgb(label))))
41         plt.show()

```

---

---

**Algorithm 4** Patch Creation (2/2)

---

```

1  def main():
2      data_loader = TiffLoader(config.filepath_satellite_nw)
3      label_loader = TiffLoader(config.filepath_label_nw)
4      size = config.size # step size
5      x_size, y_size = data_loader.size
6      x_offset, y_offset = 0, 0
7      satellite_patches, label_patches = [], []
8
9      while y_offset + size <= y_size:
10         while x_offset + size <= x_size:
11             data = data_loader.load_rgb(
12                 x_offset, y_offset, size, size)
13             label = corrected_labels(label_loader.load(
14                 Channel.GRAYSCALE, x_offset, y_offset, size, size))
15             satellite_patches.append(data)
16             label_patches.append(label)
17
18             # horizontal augmentation
19             satellite_patches.append(np.fliplr(data))
20             label_patches.append(np.fliplr(label))
21
22             # vertical augmentation
23             satellite_patches.append(data[::-1])
24             label_patches.append(label[::-1])
25
26             # horizontal & vertical augmentation
27             satellite_patches.append(np.fliplr(data[::-1]))
28             label_patches.append(np.fliplr(label[::-1]))
29             x_offset += size - config.overlap
30
31         x_offset = 0
32         y_offset += size - config.overlap
33
34         np.save(config.path_patches_satellite, satellite_patches)
35         np.save(config.path_patches_labels, label_patches)
36
37     if __name__ == '__main__':
38         main()

```

---



## Bibliography

- [Aca22] Academic Dictionaries and Encyclopedias. *Bushveld*. 2022. URL: <https://en-academic.com/dic.nsf/enwiki/765456> (visited on 03/03/2022).
- [AK15] Mariette Awad and Rahul Khanna. “Machine Learning”. In: *Efficient Learning Machines* (2015), pp. 1–18.
- [AMA17] Saad Albawi, Tareq Abed Mohammed, and Saad Al-Zawi. “Understanding of a Convolutional Neural Network”. In: *Proceedings of the International Conference on Engineering and Technology*. Institute of Electrical and Electronics Engineers Inc., 2017, pp. 1–6.
- [Arc19] ArcGIS Developer. *How U-Net Works?* 2019. URL: <https://developers.arcgis.com/python/guide/how-unet-works/> (visited on 04/04/2022).
- [Arn+18] Anurag Arnab et al. “Conditional Random Fields Meet Deep Neural Networks for Semantic Segmentation: Combining Probabilistic Graphical Models with Deep Learning for Structured Prediction”. In: *IEEE Signal Processing Magazine* 35.1 (2018), pp. 37–52.
- [Art20] Artificial Solutions. *Homage to John McCarthy, the Father of Artificial Intelligence*. 2020. URL: <https://www.artificial-solutions.com/blog/homage-to-john-mccarthy-the-father-of-artificial-intelligence> (visited on 03/09/2022).
- [Bay+14] J. Bayala et al. “Parklands for Buffering Climate Risk and Sustaining Agricultural Production in the Sahel of West Africa”. In: *Current Opinion in Environmental Sustainability* 6.1 (2014), pp. 28–34.
- [Bea+16] Amy Bearman et al. “What’s the Point: Semantic Segmentation With Point Supervision”. In: *Proceedings of the European Conference on Computer Vision*. Cham, DE: Springer, 2016, pp. 549–565.
- [BF22] Rauri Bowie and Aliette Frank. *Highveld Grasslands*. 2022. URL: <https://www.worldwildlife.org/ecoregions/at1009> (visited on 03/04/2022).
- [Bra+20] Martin Brandt et al. “An Unexpectedly Large Count of Trees in the West African Sahara and Sahelian Unexpectedly Large Count of Trees in the West African Sahara and Sahel”. In: *Nature* 587.7832 (2020), pp. 78–82.
- [Cam17] Colin Campbell. *NDVI: How to Get Much More From Your NDVI Sensor*. 2017. URL: <https://www.environmentalbiophysics.org/ndvi-sensor/> (visited on 03/09/2022).
- [CG22] Friedrich Chasin and Leo Giesen. *Machine Learning and Business Applications*. 2022.

- [Cha+20] Guillaume Chassagnon et al. “Deep Learning: Definition and Perspectives for Thoracic Imaging”. In: *Proceedings of the 2020 European Radiology* 30.4 (2020), pp. 2021–2030.
- [Cha21] Ananya Chakraborty. *How to Learn Mathematics for Machine Learning? What Concepts Do You Need to Master in Data Science?* 2021. URL: <https://www.analyticsvidhya.com/blog/2021/06/how-to-learn-mathematics-for-machine-learning-what-concepts-do-you-need-to-master-in-data-science/> (visited on 04/12/2022).
- [Chi22] Chief Directorate: National Geo-spatial Information. *CDNGI Geospatial Portal*. 2022. URL: <http://www.cdngiportal.co.za/CDNGIPortal/> (visited on 04/05/2022).
- [Cop20] B. J. Copeland. *Artificial Intelligence*. 2020. URL: <https://www.britannica.com/technology/artificial-intelligence> (visited on 03/09/2022).
- [DAT19] DATAmadness. *TensorFlow 2 - CPU vs GPU Performance Comparison*. 2019. URL: <https://datamadness.github.io/TensorFlow2-CPU-vs-GPU> (visited on 04/11/2022).
- [DFO21] Marc Peter Deisenroth, Aldo Faisal, and Cheng Soon On. *Mathematics for Machine Learning*. Cambridge University Press, 2021. URL: <https://mml-book.com..>
- [Dut+21] Varun Dutta Gupta et al. “Assessing Habitat Suitability of Leopards (Panthera Pardus) In Unprotected Scrublands of Bera, Rajasthan, India”. In: *Forest Resources Resilience and Conflicts* (2021), pp. 329–342.
- [Dwi19] Priya Dwivedi. *Semantic Segmentation - Popular Architectures*. 2019. URL: <https://towardsdatascience.com/semantic-segmentation-popular-architectures-dff0a75f39d0> (visited on 03/16/2022).
- [Ear19] Earth Observing System. *NDVI FAQ: All You Need to Know About Index*. 2019. URL: <https://eos.com/blog/ndvi-faq-all-you-need-to-know-about-ndvi/> (visited on 02/28/2022).
- [Ear21] Earth Observing System. *NDVI: Normalized Difference Vegetation Index for Agriculture*. 2021. URL: <https://eos.com/make-an-analysis/ndvi/> (visited on 03/09/2022).
- [Edi22] Editors of Encyclopaedia. *Sahel*. 2022. URL: <https://www.britannica.com/place/Sahel> (visited on 03/09/2022).
- [EF22] Elena P. Ermakova and Evgenia E. Frolova. “Using Artificial Intelligence in Dispute Resolution”. In: *Smart Innovation, Systems and Technologies* 254 (2022), pp. 131–142.

- [Enc22] Encyclopedia.com. *Anthropogenic Change*. 2022. URL: <https://www.encyclopedia.com/environment/energy-government-and-defense-magazines/anthropogenic-change> (visited on 03/04/2022).
- [Gér19] Aurélien Geron. *Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow*. 2nd ed. O'Reilly Media, Inc., 2019.
- [GIS21] GIS Geography. *What is NDVI (Normalized Difference Vegetation Index)?* 2021. URL: <https://gisgeography.com/ndvi-normalized-difference-vegetation-index/> (visited on 03/09/2022).
- [Glo20] Global Forest Watch. *Interactive World Forest Map and Tree Cover Change Data*. 2020. URL: <https://www.globalforestwatch.org/map/> (visited on 03/06/2022).
- [Goo20] Google Developers. *Classification: ROC Curve and AUC*. 2020. URL: <https://developers.google.com/machine-learning/crash-course/classification/roc-and-auc> (visited on 04/13/2022).
- [Hac22] Hacking the Case Interview. *MECE Principle: The Ultimate Guide to MECE Frameworks*. 2022. URL: <https://hackingthecaseinterview.thinkific.com/pages/mece> (visited on 03/12/2022).
- [Hal20] Jeff Hale. *The Three Most Important Composite Classification Metrics*. 2020. URL: <https://towardsdatascience.com/the-3-most-important-composite-classification-metrics-b1f2d886dc7b> (visited on 04/13/2022).
- [Han+13] M. C. Hansen et al. "High-Resolution Global Maps of 21st-Century Forest Cover Change". In: *Science* 342.6160 (2013), pp. 850–853.
- [Han+19] M. C. Hansen et al. *Global Forest Change*. 2019. URL: <https://glad.earthengine.app/view/global-forest-change> (visited on 03/08/2022).
- [Hel21] Thorben Hellweg. *Introduction to PyTorch*. 2021. URL: [https://github.com/thllwg/uvadlc\\_notebooks/blob/master/docs/tutorial\\_notebooks/tutorial2/Introduction\\_to\\_PyTorch.ipynb](https://github.com/thllwg/uvadlc_notebooks/blob/master/docs/tutorial_notebooks/tutorial2/Introduction_to_PyTorch.ipynb) (visited on 03/02/2022).
- [Hto10] Htonl. *South Africa Blank Locator Map*. 2010. URL: [https://commons.wikimedia.org/wiki/File:South\\_Africa\\_blank\\_locator\\_map.svg](https://commons.wikimedia.org/wiki/File:South_Africa_blank_locator_map.svg) (visited on 03/03/2022).
- [IBM20] IBM Cloud Education. *What Is Artificial Intelligence?* 2020. URL: <https://www.ibm.com/cloud/learn/what-is-artificial-intelligence> (visited on 02/04/2022).

- [Int22] Internationale Climate Initiative. *Large-Scale Forest Landscape Restoration in Africa*. 2022. URL: [https://www.international-climate-initiative.com/en/details/project/largescale-forest-landscape-restoration-in-africa-20\\_III\\_110-3132](https://www.international-climate-initiative.com/en/details/project/largescale-forest-landscape-restoration-in-africa-20_III_110-3132) (visited on 03/04/2022).
- [Ite22] Itemis AG. *What Is a State Machine?* 2022. URL: [https://www.itemis.com/en/yakindu/state-machine/documentation/user-guide/overview\\_what\\_are\\_state\\_machines](https://www.itemis.com/en/yakindu/state-machine/documentation/user-guide/overview_what_are_state_machines) (visited on 03/10/2022).
- [Kap21] Lohit Kapoor. *Semantic Image Segmentation Using U-Net*. 2021. URL: <https://medium.com/geekculture/semantic-image-segmentation-using-unet-28dbc247d63e> (visited on 03/02/2022).
- [Kar20] Ankit Kariryaa. *An Unexpectedly Large Count of Trees in the West African Sahara and Sahel*. 2020. URL: <https://zenodo.org/record/3978185> (visited on 03/02/2022).
- [Kaz18] Borhan Kazimipour. *What Is the Difference Between a Convolutional Neural Network and a Regular Neural Network?* 2018. URL: <https://ai.stackexchange.com/questions/5546/what-is-the-difference-between-a-convolutional-neural-network-and-a-regular-neur> (visited on 03/17/2022).
- [Kho21] Savya Khosla. *CNN - Introduction to Pooling Layer*. 2021. URL: <https://www.geeksforgeeks.org/cnn-introduction-to-pooling-layer/> (visited on 04/04/2022).
- [KS17] Neeraj Kumar and Diksha Sharma. “A Review on Machine Learning Algorithms, Tasks and Applications”. In: *International Journal of Advanced Research in Computer Engineering and Technology* 6.10 (2017), pp. 2278–1323.
- [Kum12] Tichaona Kumirai. “The Creation of a South African Climate Map for the Quantification of Appropriate Passive Design Responses”. In: *Proceedings of the CIB International Conference on Smart and Sustainable Built Environments*. Sao Paulo, Brazil, 2012.
- [Ler22] Lernhelfer Schuelerlexikon. *Sahelzone in Geografie*. 2022. URL: <https://www.lernhelfer.de/schuelerlexikon/geografie/artikel/sahelzone> (visited on 03/09/2022).
- [Lex22] Lexico. *Highveld - Meaning and Definition for UK English*. 2022. URL: <https://www.lexico.com/definition/highveld> (visited on 03/03/2022).

- [Lin+18] Jiayuan Lin et al. “Aboveground Tree Biomass Estimation of Sparse Sub-alpine Coniferous Forest With UAV Oblique Photography”. In: *Remote Sensing* 10.11 (2018), p. 1849.
- [LJY17] Fei-Fei Li, Justin Johnson, and Serena Yeung. *Lecture 11: Detection and Segmentation*. 2017. URL: [http://cs231n.stanford.edu/slides/2017/cs231n\\_2017\\_lecture11.pdf](http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture11.pdf).
- [LSD15] Jonathan Long, Evan Shelhamer, and Trevor Darrell. “Fully Convolutional Networks for Semantic Segmentation”. In: *Proceedings of the IEEE Transactions on Pattern Analysis and Machine Intelligence*. Vol. 39. 4. IEEE Computer Society, 2015, pp. 640–651.
- [Mad18] Scott Mader. *Data Preprocessing and UNet Segmentation (GPU)*. 2018. URL: <https://www.kaggle.com/code/kmader/data-preprocessing-and-unet-segmentation-gpu/notebook> (visited on 04/11/2022).
- [Mat21] Anil Chandra Naidu Matcha. *A 2021 Guide to Semantic Segmentation*. 2021. URL: <https://nanonets.com/blog/semantic-image-segmentation-2020/> (visited on 03/17/2022).
- [MB21] Stephanie Mansourian and Nora Berrahmouni. *Review of Forest and Landscape Restoration in Africa 2021*. 2021. URL: <https://reliefweb.int/report/world/review-forest-and-landscape-restoration-africa-2021> (visited on 03/04/2022).
- [MBA21] MBA Crystal Ball. *MECE Framework McKinsey*. 2021. URL: <https://www.mbacrystalball.com/blog/strategy/mece-framework/> (visited on 03/12/2022).
- [McC07] John McCarthy. *What is AI? Basic Questions*. 2007. URL: <http://jmc.stanford.edu/artificial-intelligence/what-is-ai/index.html> (visited on 03/09/2022).
- [McK+20] Scott Mayer McKinney et al. “International Evaluation of an AI System for Breast Cancer Screening”. In: *Nature* 577.7788 (2020), pp. 89–94.
- [Mer19] Willem van der Merwe. *A Rainbow of Landscapes*. 2019. URL: <http://thebdi.org/2019/07/09/a-rainbow-of-landscapes-part-one/> (visited on 03/03/2022).
- [Mer21] Merriam-Webster. *Bushveld Definition and Meaning*. 2021. URL: <https://www.merriam-webster.com/dictionary/bushveld> (visited on 03/03/2022).

- [MHD99] Thaís Maia Araújo, Niro Higuchi, and João Andrade De Carvalho. “Comparison of Formulae for Biomass Content Determination in a Tropical Rain Forest Site in the State of Pará, Brazil”. In: *Forest Ecology and Management* 117.1-3 (1999), pp. 43–52.
- [Mit97] Tom Mitchell. *Machine Learning*. 2nd ed. 1997.
- [ML21] Derrick Mwit and Katherine Yi Li. *Image Segmentation in 2021: Architectures, Losses, Datasets, and Frameworks*. 2021. URL: <https://neptune.ai/blog/image-segmentation> (visited on 04/02/2022).
- [Moo+21] Jaewoong Moon et al. “Study on Machine Learning Techniques for Malware Classification and Detection”. In: *KSII Transactions on Internet and Information Systems* 15.12 (2021).
- [Mun22] Munich Zoo Hellabrunn. *Tierpark Hellabrunn: Blue Crane*. 2022. URL: <https://www.hellabrunn.de/en/hellabrunn-wildlife/africa/blue-crane/animal-dictionary/> (visited on 04/09/2022).
- [Nat15] National Geographic Society. *South Africa*. 2015. URL: <https://kids.nationalgeographic.com/geography/countries/article/south-africa> (visited on 12/29/2021).
- [Nat22] National Geographic Society. *Plateau Definition*. 2022. URL: <https://www.nationalgeographic.org/encyclopedia/plateau/print/> (visited on 03/03/2022).
- [Nbr20] Nbro. *What Is a Fully Convolution Network?* 2020. URL: <https://ai.stackexchange.com/questions/21810/what-is-a-fully-convolution-network> (visited on 04/02/2022).
- [Nik20] Nikita Kraetzig. *Five Things to Know About NDVI (Normalized Difference Vegetation Index)*. 2020. URL: <https://up42.com/blog/tech/5-things-to-know-about-ndvi> (visited on 02/28/2022).
- [Olu22] Motunrayo Olugbenga. *Balanced Accuracy: When Should You Use It?* 2022. URL: <https://neptune.ai/blog/balanced-accuracy> (visited on 04/13/2022).
- [ON15] Keiron O’Shea and Ryan Nash. “An Introduction to Convolutional Neural Networks”. In: (2015).
- [Pai20] Aravind Pai. *ANN vs CNN vs RNN - Analyzing Three Types of Neural Networks in Deep Learning*. 2020. URL: <https://www.analyticsvidhya.com/blog/2020/02/cnn-vs-rnn-vs-mlp-analyzing-3-types-of-neural-networks-in-deep-learning/> (visited on 03/17/2022).

- [Pat22] Patchify. *patchify*. 2022. URL: <https://pypi.org/project/patchify/> (visited on 04/10/2022).
- [RFB15a] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. “U-Net: Convolutional Networks for Biomedical Image Segmentation”. In: *Proceedings of the International Conference on Medical Image Computing and Computer-Assisted Intervention*. Cham, DE: Springer, 2015, pp. 234–241.
- [RFB15b] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. *U-Net: Convolutional Networks for Biomedical Image Segmentation*. 2015. URL: <https://lmb.informatik.uni-freiburg.de/people/ronneber/u-net/> (visited on 04/02/2022).
- [Roy20] Rupali Roy. *Difference Between AI, ML and DL*. 2020. URL: <https://towardsdatascience.com/understanding-the-difference-between-ai-ml-and-dl-cceb63252a6c> (visited on 03/09/2022).
- [SA04] Robert Simmon and Jesse Allen. *Vegetation and Rainfall in the Sahel*. 2004. URL: <https://earthobservatory.nasa.gov/images/7277/vegetation-and-rainfall-in-the-sahel> (visited on 12/29/2021).
- [Sah18] Sumit Saha. *A Comprehensive Guide to Convolutional Neural Networks - The ELI5 Way*. 2018. URL: <https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53> (visited on 03/18/2022).
- [Sam59] Arthur Samuel. “Some Studies in Machine Learning Using the Game of Checkers”. In: *IBM Journal of Research and Development* 3.3 (1959), pp. 210–229.
- [Sci22a] ScienceDaily. *Artificial Intelligence*. 2022. URL: [https://www.sciencedaily.com/terms/artificial\\_intelligence.htm](https://www.sciencedaily.com/terms/artificial_intelligence.htm) (visited on 03/09/2022).
- [Sci22b] Scikit-learn. *MinMaxScaler*. 2022. URL: <https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.MinMaxScaler.html> (visited on 04/10/2022).
- [Sci22c] Scikit-learn. *StandardScaler*. 2022. URL: <https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html> (visited on 04/10/2022).
- [Sha18] Shachi Shah. *Do We Really Need GPU for Deep Learning? - CPU vs GPU*. 2018. URL: <https://medium.com/@shachishah.ce/do-we-really-need-gpu-for-deep-learning-47042c02efe2> (visited on 04/11/2022).

- [Siy19] Siyabona Africa. *Umbrella Thorn - Acacia Tortilis*. 2019. URL: [https://www.krugerpark.co.za/africa\\_umbrella\\_thorn.html](https://www.krugerpark.co.za/africa_umbrella_thorn.html) (visited on 03/12/2022).
- [Siy21] Siyabona Africa. *Highveld: South Africa Geographic Regions*. 2021. URL: <https://www.nature-reserve.co.za/south-africa-info-highveld.html> (visited on 03/04/2022).
- [SKS15] Sebastian Schnell, Christoph Kleinn, and Göran Ståhl. “Monitoring Trees Outside Forests: A Review”. In: *Environmental Monitoring and Assessment 2015 187:9* 187.9 (2015), pp. 1–17.
- [SLD15] Evan Shelhamer, Jonathan Long, and Trevor Darrell. *Fully Convolutional Network*. 2015. URL: <https://paperswithcode.com/method/fcn> (visited on 04/01/2022).
- [Sou14] South African Government. *Forestry*. 2014. URL: <https://www.gov.za/about-sa/forestry> (visited on 03/08/2022).
- [Str+12] L. C. Stringer et al. “Challenges and Opportunities in Linking Carbon Sequestration, Livelihoods and Ecosystem Service Provision in Drylands”. In: *Environmental Science and Policy* 19-20 (2012), pp. 121–135.
- [Swi22] Swift Geospatial Solutions. *Tree Count Application*. 2022. URL: <https://swiftgeospatial.solutions/2020/06/24/tree-count-application/> (visited on 03/07/2022).
- [Tem+15] Hailemariam Temesgen et al. “A Review of the Challenges and Opportunities in Estimating Above Ground Forest Biomass Using Tree-Level Models”. In: 30.4 (2015), pp. 326–335.
- [Ten22] TensorFlow. *GPU Support*. 2022. URL: <https://www.tensorflow.org/install/gpu> (visited on 04/11/2022).
- [The14] The Free Dictionary. *Definition of Highveld*. 2014. URL: <https://www.thefreedictionary.com/highveld> (visited on 03/03/2022).
- [The15] The Free Dictionary. *Definition of Bushveld*. 2015. URL: <https://www.thefreedictionary.com/bushveld> (visited on 03/03/2022).
- [Zaf+18] Iffat Zafar et al. *Max Unpooling - Hands-On Convolutional Neural Networks with TensorFlow*. 2018. URL: <https://www.oreilly.com/library/view/hands-on-convolutional-neural/9781789130331/6476c4d5-19f2-455f-8590-c6f99504b7a5.xhtml> (visited on 04/04/2022).
- [Zha+18] W. J. Zhang et al. “On Definition of Deep Learning”. In: *Proceedings of the World Automation Congress*. Stevenson, WA, USA: IEEE Computer Society, 2018, pp. 232–236.