Geo-information Science and Remote Sensing

Thesis Report GIRS-2020-41

Exploring the use of neural networks for object instance detection below the pixel resolution level

Finding Date Palm trees using Sentinel-2 imagery and FCNN's

Laurens Buddingh

07/10/2020

WAGENINGEN
UNIVERSITY & RESEARCH

Exploring the use of neural networks for object instance detection below the pixel resolution level

Finding Date Palm tree locations using Sentinel-2 imagery and FCNN's.

Laurens Buddingh

Registration number 960430143010

Supervisors:

Lammert Kooistra
Ioannis Athanasiadis

A thesis submitted in partial fulfilment of the degree of Master of Science
at Wageningen University and Research Centre,
The Netherlands.

7/10/2020
Wageningen, The Netherlands

Thesis code number: GRS-80436
Thesis Report: GIRS-2020-41
Wageningen University and Research Centre
Laboratory of Geo-Information Science and Remote Sensing

Abstract

Fully convolutional neural networks (FCNN's) are a promising technique to use spatial and spectral information within imagery for object detection and image segmentation. A well-built and functioning FCNN model that can successfully classify small objects using relatively coarse satellite imagery would open pathways for the creation of extensive datasets in both space and time.

This study seeks to implement and assess the use of FCNN models to detect Phoenix Canariensis date palm trees in the Canary Islands. Due to the use of Sentinel-2 imagery as a source of reflectance data, the size of the objects (date palms) is between 0.6 and 0.8 times the size of a reflectance pixel. Since the palm trees are smaller than the pixel size of the reflectance data, it is difficult to correctly identify the palm trees. However, an evaluation over similar data was done by a previous study with promising results. The starting point of this study was to see if the same could be done for the data and problem of this study.

This study first establishes the full pipeline of the data from its source to the finished model and then discusses the predictions. The ground truth data used for the study was a point-based dataset gathered through field surveys, covering the whole of the Canary island region, making the whole of the Canary Islands the sampling area for this study. The sampling area was cut into chunks and the ones containing no positive ground truths were discarded (no palm trees), before being loaded into the model in a 8/2 training validation split.

Building on previous works as a foundation, Deeplabv3 model architecture was implemented through python (Pytorch) code to create the finished model, and its fidelity was assessed using F1 and Au-ROC scores and its model convergence was assessed using the losses obtained during training and testing. With an optimal F1 of ***2.763%*** *the usability and fidelity of the model is at a point where it is rarely able to correctly assign the palm tree label on pixels correctly. The model is thus unsuitable for the accurate mapping of palm tree locations for further research. While the Au_ROC score scored significantly better at* ***0.74476*** *this was judged to be a side effect due to bias obvious within the ground truth data, greatly oversampling urban areas in comparison to other types of land use, likely as a result of the ease of access these locations provided to the field surveyors compared to the more dense forests and natural areas.*

Despite the fidelity of the model causing it to be unusable for palm tree mapping, results from previous studies and insights obtained through analysis of the source data do show that it is still possible to greatly improve and built upon the models results. This could be done by changing the source data and possibly applying different types of loss functions with a focus on improving model training. Data wise there is room for improvement as well, by integrating more spectral data and increasing the importance of spatial characteristics during training for example, increasing the amount of links between the data types for the model to come to a better classification.

Keywords: *Remote sensing, neural networks, deep learning, semantic segmentation*

Preface

The use of deep learning for object detection and classification within remote sensing data and Satellite data especially is a promising new field which could possibly greatly change and affect the way Satellite data is seen and used. I started this study as a combination of two different possible research topics, but during the course of the research the focus of the study drifted away from looking at ways to utilise Big Data for use in training Satellite deep learning models, and focused more on the application of those models themselves.

I want to give a large amount of thanks to Lammert Kooistra and Ioannis Athanasiadis, my two supervisors at Wageningen University. During critical stages of the thesis project, their guidance and advice has helped me get this thesis to the current stage.

I also want to give my thanks to the people at VITO. First I want to give thanks to Stephanie Delalieux who helped give me the idea of the final direction of the study, and I would like to thank Maria Culman Forero for helping me get access to the ground truth datasets and giving me some help with the background and some tips for the model architecture part of the study. From VITO I also want to thank Kristof van Tricht who helped me when I got massively stuck at implementing a part of the code for setting up the model architecture.

Furthermore, for similar issues with code I want to thank Benjamin Kellenberger and Sylvain Lobry from WUR for helping me with some implementation issues.

Finally, I would like to thank all my friends and my parents who supported me during the progress of this thesis and some of the complications that occurred during the Corona outbreak during this time.

Table of contents

Table of tables

Table of figures

Abbreviations

RPW – Red Palm Weevil (*Rhyncophorus Ferrugineus)*

CIPD – Canary Island Date Palm (*Phoenix Canariensis*)

FCNN – Fully convolutional neural network

NN – Neural network

NIR – Near infra-red

Uint8 – unsigned integer number stored with 8 bit

COCO – Common objects in context

ASPP – Atrous spatial pyramid pooling

RPN – Region proposal network

ROI – Region of interest

IoU – Intersect over union

MSE – Mean squared error

ROC – Receiver operating characteristic

Au_ROC – Area under receiver operating characteristic

R-CNN - Regions with convolutional neural networks

TP – True positive

TN – True negative

FP – False positive

FN – False negative

GPU – Graphics processing unit

RAM – random access memory

1 Introduction

1.1 Backdrop: the threat of the Red Palm Weevil

The start of this research project happened because more information had to be acquired on a major threat to Palm trees and Date Palm trees that has achieved global spread in recent decades. The Red Palm Weevil (*Rhyncophorus Ferrugineus) (*RPW*)* has been a persistent pest since they swept the Gulf states in the 1980's [1]. The beetle, originally native to Asia, swept through the Gulf states and now threatens palms worldwide. Even Isolated communities like the Canary Islands have been affected by the spread of the pest, and studies have shown that without intervention the spread of the pest could reach even further [1]

If a palm tree is not detected early, it is nearly certain that the palm tree will decease due to a lack of nutrient and water to the tree through a thorough blocking of the xylem and phloem by the RPW [1]. As the nymphs in the infected palm tree come to maturity, they fly out and plant their eggs in neighbouring palm trees, this way causing an ever-growing cascade of damaged palms. Commonly used methods for early detection of RPWS are the use of thermal imagery and acoustic sounding [2], [3], but lately hyperspectral data has seen useful in early detection as well [4].

One of the larger challenges in the testing of new techniques by which to detect such things as the RPW, is the gathering of enough data to provide a varied and large data lake on which you can employ statistical analysis. In the case of the RPW this is difficult, due to the often-multivariate (e.g. hyperspectral) nature of the data as well as due to data scarcity, especially on infected palms.

Data on larger clusters of palm trees can readily be obtained from palm tree plantations, but they will often cut down a tree at the first sign of RPW infestation to prevent further damage to the rest of the palms [1]. This makes it difficult to get enough data on infected palm trees in agricultural setting, since many techniques used for detection require specific fly-over planning in advance due to their generally drone based or high spatial resolution (and thus low swat width) satellite based platforms. By the time such a fly-over can be arranged the palm tree has already been felled.

To counteract this problem, it would be useful to attain a large sample size of palm trees both within and outside plantations from which useful information can be obtained periodically such as the thermal and hyperspectral imagery.

1.2 Palm tree characteristics and how they limit data platforms

With the aim of paving the way to new methods of data collection and thus increasing the amount of available data on palm trees, it is important to know something about the physiological characteristics of these palms and in what ways they are generally distributed over space. In general, this can differ greatly based on the genus and species of Palm tree in question. Due to their primacy within this study, it is important to know these characteristics for the Canary Island Date Palm (CIPD) (*Phoenix Canariensis*).

The CIPD is both a favored ornamental piece as well as a naturally occurring species on many of the Canary Islands. Due to this, the tree can be found in both urban environments for their ornamental value as well as in vegetated regions (most commonly in forests) outside of the urban sprawl in rural or natural areas [5].

Within urban environments, the trees are often planted and thus follow strict patterns such as lines along roads or walkways or checkboard patterns in parks. In rural or natural areas, the spread of palm trees is less organized, especially in areas where they do not serve as ornaments or as an agricultural product (plantations) and are naturally occurring instead. In forested regions, many types of vegetation other than the palm tree can be packed tightly together in a thicket.

Individual CIPD's vary in height and width based on their age and on climate conditions. On the Canary Islands, the CIPD's in the easternmost islands rarely grow above shrub height due to the drier climate caused by the proximity to the African continent [5], while in more optimal conditions on the center and westernmost Canary islands, adult specimens can grow up to 10-20 meters in height with a crown diameter of 6-8m, but adult specimens in sub-optimal environments will barely be bush height.

The size of the individual palm trees limits per pixel classification techniques to techniques that use data of higher spatial resolution than the crown size (the most important measure when looking from above), and this can be done with a moderate degree of precision [6], [7].

The limit of such techniques however is once again in the data source. The high accuracy multispectral and hyperspectral data requires specific flyovers, and it will be expensive and time consuming to produce a dataset with good temporal coverage. Meanwhile, satellite platforms such as Landsat 7+ and Sentinel-2 fly over most places in the globe often enough to get a dense temporal dataset, but they are limiting due to the coarseness of their spectral bands (10-30m for most bands). Should the data from either of these platforms prove usable using new techniques however, it would be possible to greatly increase the amount of data available on CIPD's. Increasing the amount of data would allow policy makers and other decision makers to accurately know the distribution of palm trees, allowing for more informed and thought out decisions [8].

1.3 Exploring new ways to utilize sentinel-2 data for obtaining palm tree data

The creation of a dataset of palm tree locations on a national or perhaps even the scale of their entire respective biome on a 10m resolution would be an asset for combating the spread of RPW infestations. The disadvantages of datasets like Quickbird or alternatives like Pleiades, or even drone flights are that by their nature their temporal resolution and possibly their spatial extent is unsuitable for temporal or real-time investigation of Palm tree locations, this lack of temporal data would make it unusable for a complex problem such as detecting the spread of RPW's [9]. These platforms are limited further when one takes into account the monetary costs of keeping an updated database of Quickbird or Pleiades data over a large swath of area, or the concessions made in spectral accuracy by focusing on ever higher spatial resolution [10].

The ideal dataset would combine high spatial resolution, spatial distribution and temporal resolution but due to trade-offs inherent in remote sensing such datasets currently doesn't exist [9]**.** A promising data source for creating a palm location dataset would be the Landsat or Sentinel projects. Their temporal resolution is high (~5 days revisit time by the 2 satellites, varying slightly depending on the latitude), Spatial extent is high as well (290km swath width) and the spectral accuracy might provide enough information for this case study [11], the trade-off is a lower spatial resolution (10-60m for Sentinel-2 bands)

To gain access on the large datasets that would become available, techniques that could obtain accurate data on palm trees from Sentinel-2 and Landsat imagery are very interesting. This study attempts to explore a new way in which identification of instances of CIPD might be possible within a pixel of Sentinel-2 imagery. This means that we are trying to assess the presence of an object that is of lower size than the native resolution of the most precise bands available through the sentinel-2 platform. Table 2 contains information on the spectral data available from the Sentinel-2 images.

Aside from it being very interesting on its own to see if such detection methods are achievable and accurate, opening up ways for its use for many other types of object classes, assessing the locations of palm trees in this way could be useful due to the role palm trees serve within their environment. Better assessments of palm tree density and situation should lead to better assessment of coastal erosion risks [12] **,** as well as provide information on their economic benefits in agriculture, where date palm trees are often used as a cash crop both historically and in the present day [13]. Palm trees also prove as a lure for tourists, the aesthetics and expectations of a palm surrounded white sandy beach providing a luxurious tourist fantasy [14]. The benefits of such a dataset are thus possibly wider scoped than the problem of RPW infestations that served as the impetus for this study.

1.4 The promising new technique of fully convolutional neural networks

To create a dataset on palm trees using Sentinel-2 data, it would require the use of a technique able to detect object smaller than the pixel size. While this seems impossible, through the use of Neural networks (NN) and the creation of a fully convolutional neural network model (FCNN) a study has proven this should be possible for other objects at different pixel to object ratios and object densities [15]. Where the detection of objects in the cluttered objects becomes hard, this is where deep learning comes in. Deep learning and neural networks can make use of high spectral resolution data to abstract object information invisible to human sight, learning complex relations between spectral bands [15]. This should allow FCNN's to identify object specific spectral signatures that allow for per pixel segmentation.

Exploring ways to adapt their approach to the datasets available in this study, instance segmentation of CIPD's might be possible within Sentinel-2 data, by finding relations between spectral signatures where human eyes would fail.

Should the utilization of FCNN's to detect Palm trees within sentinel-2 images be successful, this could be a great method to detect environmental damage occurring in Palm tree populations, allowing policy makers to make decisions to hamper further spread of the damage. It might also greatly increase the pool of data on infected palm trees due to the huge swaths of data available through the sentinel-2 platform in both time and space, allowing for more robust statistical analysis on methods of RPW infestation detection.

1.5 Research objective and hypothesis

The main research objective is to explore the use of FCNN's and their use in per pixel semantic segmentation of objects smaller than the pixel size. In this study, the object of study will be CIPD's in Sentinel-2 imagery. The highest resolutions available from any Sentinel-2

band are still lower than the largest size mature CIPD specimens achieve, so they fall within the research scope. The only way of detecting CIPD's in cluttered pixels would be to have the FCNN establish spectral relations for the CIPD's human eyes would normally miss, which was successfully done in the paper by Rodriguez et al [15]. FCNN's are a promising new technique in the classification of remote sensing imagery, and aside from the techniques applied by Rodriguez et al, other FCNN techniques that could have been applied will also be studied.

The hypothesis for this setup using CIPD's as the object of interest is "Instance segmentation using fully convolutional neural networks should provide adequate predictive power in finding Canary Island Date Palms within Sentinel-2 imagery".

The sentinel-2 imagery and the CIPD's chosen as subject of this thesis are only an example of use of the FCNN semantic segmentation for instance detection that is used. Should the algorithm be successful, it might have a wider range of application for many different types of objects. A similar study on which a significant part of the methodology was based has proven results for various objects already, with the object size/pixel area and the density of the objects used to train the FCNN seeming to be one of the most contributing factors as to the success of the FCNN model [15].

1.6 Research questions

To answer the hypothesis and resolve the research objective, the following research question needs to be answered.

1. How can FCNN's be used to create a map of CIPD locations within sentinel-2 data?

To answer this question, the following sub questions will need to be answered first:

a) Is the spectral and ground truth data used by this study of sufficient quality to create a good and unbiased per pixel image segmentation model through FCNN's?
b) How does this study differ from the study done by Rodriguez et al. [15], and how can their methodology be altered to work on this study's datasets?
c) Is the predictive power of the model adequate for use in creating Palm tree maps from Sentinel-2 data?
d) How consistently can the model detect Palm trees for different types of land use types?

Research question a and b are directed towards the setup and premise of this study, focusing on the usability of the data (a) and the comparison to the methodology that this study uses as its main inspiration (b). Research question c and d address the crux of the hypothesis, the model effectiveness (c) and its possible bias and other shortcomings (d).

2 Literature review

2.1 FCNN's as a new way to derive pixel level segmentation results from satellite data

The use of FCNN's for image segmentation tasks originally came to prominence as a different take on the use of R-CNN, incorporating this technique generally used for the classic computer vision task of *object detection* and changing the final layer to produce a feature map instead of a classification, turning the end result to a per pixel semantic segmentation [16].

R-CNN's were established as a way to address the problem of object detection within images by using a region proposal network to get relevant information on each image [17]. Later there were improvements on this technique, significantly improving the rate of RoI (region of interest) selection. The resulting techniques were called Fast R-CNN, which was followed by Faster R-CNN and YOLO [16]. The problem with these techniques however is that by making use of RoI-pooling, it gets a coarse quantization of the data, disallowing the use of accurate per pixel segmentation. By applying a segmentation mask on each region of interest prior to their pooling in parallel with the existing branch for classification and bounding box regression, MaskRCNN was the first algorithm to successfully apply R-CNN's for per pixel level segmentation tasks [18]. The architecture of mask-RCNN is visible in figure 1.

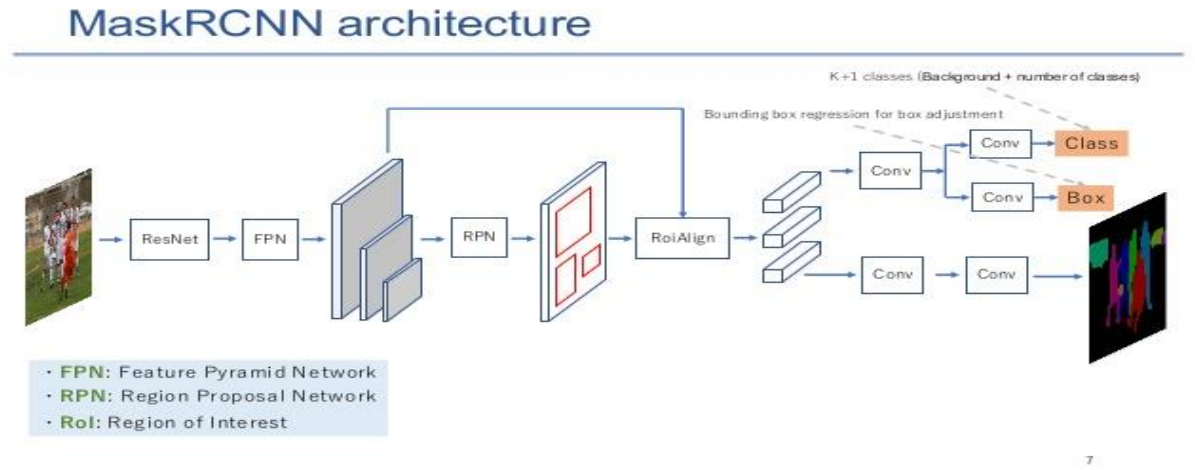

Figure 1: Visual representation of the Mask RCNN architecture, the first RCNN architecture for pixel level segmentation applications. (source: https://www.google.com/url?sa=i&url=https%3A%2F%2Fwww.slideshare.net%2Fhitheone%2Fmaskrcnn-for-instance-segmentation-117),

After the advent of Mask-RCNN the use of FCN was quickly superseded by U-net, a network that makes use of contracting path to capture the general context of images and uses a symmetric expanding path for precise localization of instances of interest [19]. The symmetric nature of the U-net architecture is clearly visible in figure 2.

Originally created for problems in the bio-medical field, U-net quickly showed promise in remote sensing problems as well [16], [20], achieving high fidelity scores in IBRIS and Deepglobe data challenges.

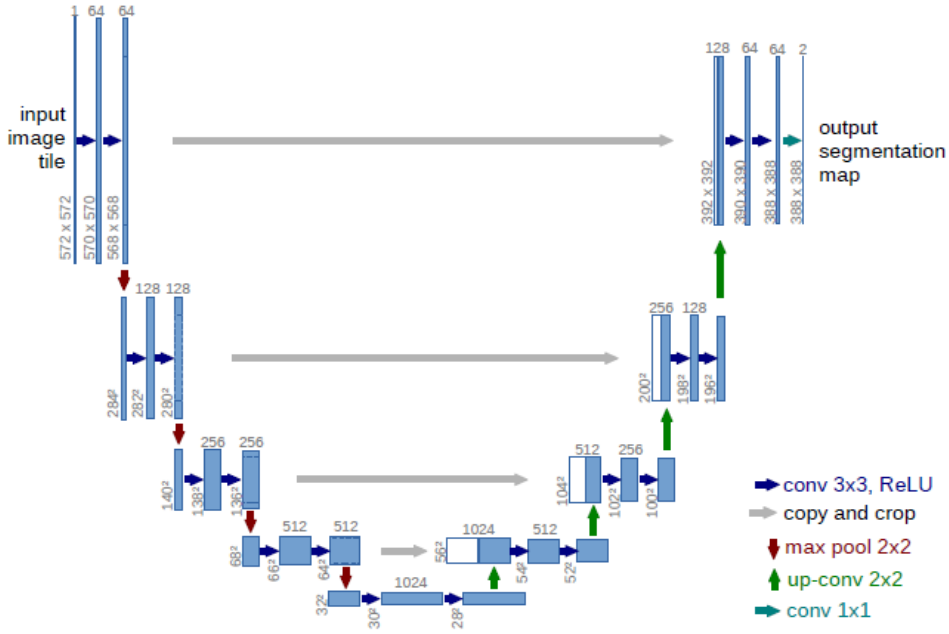

Figure 2:Visual representation of U-net architecture for semantic segmentation. Source: [19]

Another architecture built up on the foundation laid by MaskRCNN is the architecture known as Deeplab, as of this writing it is on its third iteration: DeeplabV3. The makers of this algorithm saw the same problem as the makers of U-net with RCNN's, the difficulty of gathering context during training and working with regions of interest on different scales [19], [21]. Deeplab makes use of parallel or cascade atrous convolutions to capture the multi-scale context of objects or clusters of objects of interest[21]. After proposing the network and application of atrous spatial pyramid pooling as a method to achieve this, Deeplabv3 attained comparable scores to state of the art models within the field [21]. Figure 3 shows how Deeplab makes use of atrous spatial pyramid building to create feature maps that better grab the image context.

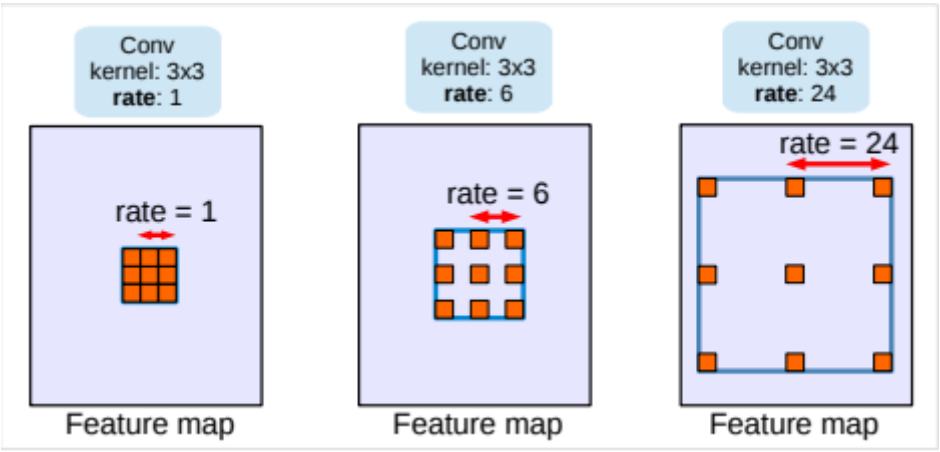

Figure 3:Visual representation of how different atrous rates can enlarge the field of view of the model, allowing for detection at multiple scales. Source [21]

2.2 Differences with the paper "Counting the uncountable: Deep semantic density estimation from space"

Since it has been referenced plenty already so far in the paper, it should not come as a surprise that the paper "Counting the uncountable: Deep semantic density estimation from space" by Rodriguez et al. [15] is one of the main inspirations and foundations for the methodology, background and analysis used within this study. This paper served as one of the catalysts that started this study.

The study by Rodriguez et al. proposed a new way to estimate object densities for objects of sub-pixel sizes through deep semantic segmentation, a combination of Deep learning and semantic segmentation. The study did this for four different object types, but the coconut and island palms can best be linked to this study. For these object types, their proposed methodology achieved a precision and recall of 0.756 and 0.821 respectively for their best performing model on Coconuts and a precision and recall of 0.795 and 0.796 respectively for their best performing model on island palms.

Since the end results differ and other choices have been made during the research process, it is necessary to explain differences in data, methodology and analysis between this paper and the "counting the uncountable".

Table 1:Differences and similarities between this study and the research performed by Rodriguez et al.

Research parameter/choice	*Rodriguez et al*	*This study*	*Differs?*
Spectral dataset for FCNN	Sentinel-2 imagery	Sentinel-2 Imagery	No
Band selection	All	2,4,8	Yes, though best performing bands where chosen due to implementation limitations
Spectral data acquisition	One set of spectral data closest to ground truth creation date	One set of spectral data close to ground truth creation date	No, though since the time of ground truth creation for our dataset is more vague, the median over a year was taken as 'closest'
Ground truth for palm trees	Polygons created through faster RCNN object detection on high resolution imagery	Points of palm tree locations with multiple additional attributes obtained through surveys of orthophotos by experts	Yes
Type of palm tree	Oil palm, Coconut	Canary island date	Yes

	palm	palm	
Model architecture	Deeplabv3+, atrous, 6 convolutional layers	Deeplabv3+, atrous, 6 convolutional layers	No
Ground truth for FCNN	Palm tree density map	Palm tree density map	No
Loss functions used	Binary cross entropy + object density	Mean squared loss	Yes
Scoring metrics used	IoU, precision, recall, MSE and MAE	F1, Au_roc, MSE loss	Yes. Though F1 includes precision and recall in how its calculated, IoU is missing.
Amount of palm trees	537500	555731	No, both data sets are about equal in size

The largest difference between this study and the study by Rodriguez et al. is in the data used to train the images, the ground truth data and the satellite data both. The satellite data used is Sentinel-2 data for both studies, but their algorithm supports the use of images with over 3 bands of data, while that was not possible to implement in the timeframe of this study due to lack of module support. From their results, vegetation objects require both infrared and RGB data to lead to good results, so band 2, 4 and 8 where selected for this studies 3 band limit where the red and blue band help identify the textures and the Infrared adds extra identifying information for the vegetation (Palm trees) specifically. Further elaboration on why only three bands where chosen instead of taking all the bands as data for the model training is done in section 3.3.1, and the consequences of the choice are discussed in section 5.1.

Another major difference is the type of ground truth used to train the algorithm. The ground truth used in their paper is a dataset obtained by running a classifier (Faster RCNN object detection) using manually detected palm trees over high resolution imagery (1m spatial resolution), where they score a precision and recall score of 0.77 on the closest matching vegetation in their study (Coconut Palm). This means that their dataset should be a relatively unbiased ground truth with 0.77 precision and recall for all Palm trees in their imagery. The dataset used in this study however is a dataset of palm trees manually annotated by experts through orthophotos made available by the Spanish national geoportal. While the precision and recall of this ground truth should be close to 1 (depending on the quality of work done by the experts in question), the dataset does seem to contain a heavy bias towards areas that are easy to assess through orthophotos (urban areas/areas near roads). The dataset also contains far fewer datapoints in natural conservation areas for example, where palm trees would be mixed in with other vegetation and difficult to distinguish by eye.

The model structure used for both this study and the study by Rodriguez is largely similar, both this study and theirs use a model of 6 Resnet blocks (with ASPP) and use an MSE loss function to do a density calculation that also helps score an y-label.

Both this study and theirs have set the stride of the Resnet blocks to 1 to avoid further spatial resolution loss (see figure 4 for model architecture.

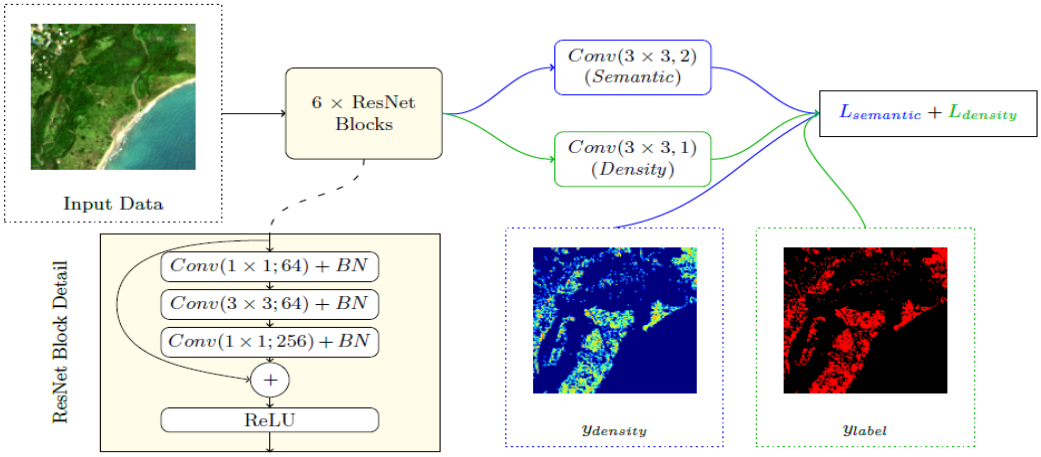

Figure 4: Visual representation of the model architecture used by both this study and the study of Rodriguez. For further information on the Conv blocks, see image 3. Source [15]

The physiological characteristics of the Palm trees in this ground truth should provide an Object area/pixel area ratio of between 0.6-0.8, which is in line with vegetation samples from their study [15]. A difference between the training of images for this study and theirs might be that this study included images of low instance density, sometimes even including images with a total object of interest ratio of lower than 1%, since all areas containing one or more palm trees where considered as valid training images.

2.3 A short review on other FCNN applications using satellite Imagery

For a technique inherently data hungry, generally requiring large amounts of data to produce accurate models, it should be no surprise that the growth of applications of FCNN's in the remote sensing world largely coincides with the release of new and extensive datasets that can be fed into these networks.

Remote sensing as a field has no lack of large pools of remote sensing data that could be used, just with the combined data from the Sentinel and Landsat Satellites on their own, but there is a clear lack of labelled satellite data required to train the FCNN's for satellite imagery since most popular datasets use on the ground labelled video or photo data [22]. This means that for most studies planning to train a new FCNN using satellite data, data will have to be trained from scratch on a self-created labelled dataset like this study and the study of Rodriguez et al. have done [15].

This however does not mean that the datasets currently available have few uses in the GIS sphere. The Cityscapes dataset, which focuses on urban environments has seen uses for urban scene detection, which could lead to a wide range of applications one of them being for use in self driving vehicles [23], but even there intersect over union detection tasks (67%) and instance level tasks (4%) still prove highly difficult. In this context, instance level tasks stands for simultaneously completing object detection and semantic segmentation in a single go [24].

For satellite's specifically, high resolution satellite imagery and FCNN's have seen use in tasks such as road extraction, with an intersect over union score of 0.64 for the Deepglobe – CVPR 2018 road extraction sub challenge.

The Deepglobe challenge is a very interesting data challenge in the remote sensing field that similarly to other challenge datasets such as COCO do in other computer vision fields it provides three datasets and corresponding evaluation methodologies *with labelled datasets* freely available online [25]. The existence of this dataset has already, and will no doubt further increase the competence and knowledge in the field of fully convolutional neural networks as tools for computer vision in Satellite Imagery, and would strongly urge to look at the top competitors of this challenge in the future to get a grasp of state of the art techniques within this field.

Figure 5:Three examples featuring subsequently the road extraction, building detection and land cover classification challenge datasets within the Deepglobe challenge. Source [25]

3 Materials and methods

3.1 Limiting factors and data restrictions

The experimental setup is both limited by the available ground truth data and the time constraints imposed by its nature as a master thesis. Despite these constraints it does intend achieve the best result with the time and the available resources. The goal is to assess the feasibility of an algorithm that can predict instances of sub-pixel level size within Satellite-imagery, trained from a ground truth mask of the objects in question. For this study, the objects to be examined will be Canary Island Date Palms**.**

3.2 Study area

The choice for the study area was mainly determined by the availability of the ground truth dataset that will be elaborated on in 3.3.2. The scope of the area is the entire Canary Islands, an island group on the west coast of the African continent and an autonomous municipality of Spain.

The climate on the Islands changes the closer you get to the (relatively compared to the sea) arid African continent, and this has an effect on the growth of the CIPD's in this study [5].

The human population on the Islands also differs, with Tenerife and Gran Canaria being far more populous and urbanised than some of the other islands in the archipelago. The isolated nature of the islands and the difference in climate thanks to proximity to the continent have led to the existence of a great variety of landscapes from arid sand dunes to tropical forest to be present on the archipelago [26]. Furthermore due to the high amount of geophysical and volcanic activity present on the archipelago, near mountainous and heavily sloped areas are abundant on the islands [27]. The study area is presented in figure 6.

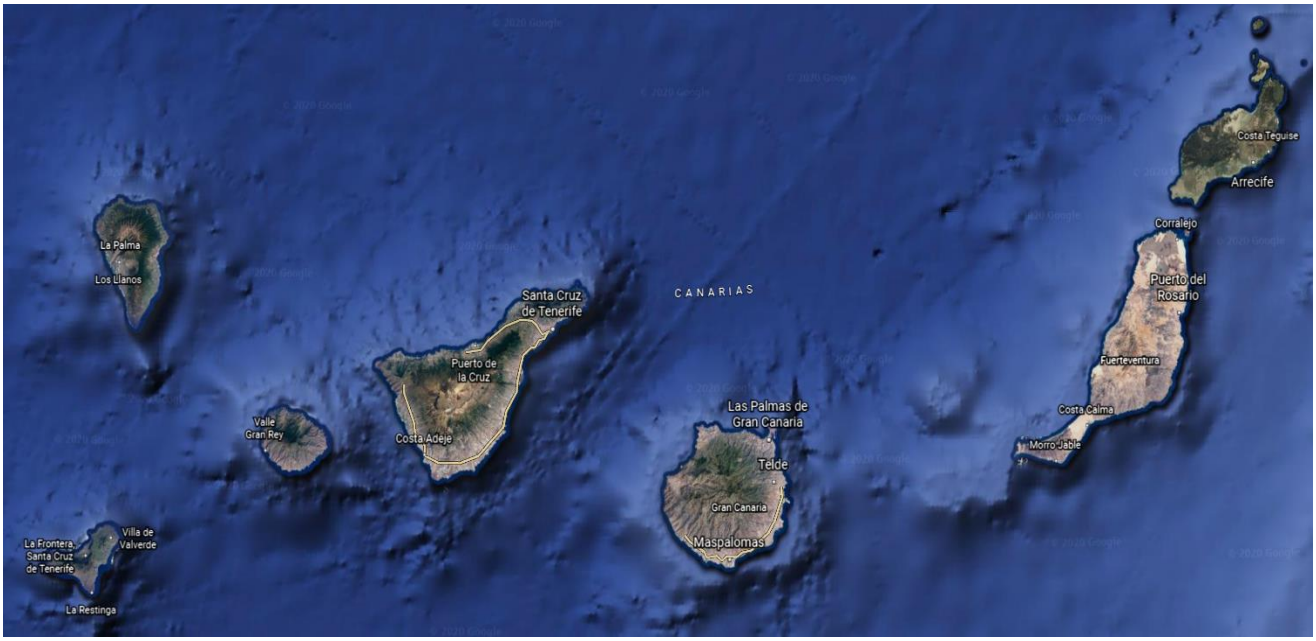

Figure 6: The study area, The Canary Islands (Canarias in Spanish). All the islands shown in this picture are part of the dataset. Image obtained through Google earth 7/25/2020

3.3 Data requirements

For the creation of the semantic segmentation algorithm two sources of data input are required. These datasets will be outlined in the paragraphs below

3.3.1 Spectral data

The first is an imagery source, in this case the imagery source will be sentinel-2 data obtained from the Copernicus hub through Google Earth engine. The data contains median cloud free images obtained a year before and a year after the acquisition of the ground truth data, to ensure that no holes are present within the imagery. Not all the bands of the sentinel-2 data have the same spatial resolution as can be seen from table 2, so the bands of lower resolution where converted to 10m bands using bilinear interpolation. The non-discrete nature of the data lends well to this manner of interpolation, and it is less potentially volatile than cubic convolution as an interpolation method [15], [28].

While all this pre-processing had finished, it was discovered that the modules intended to work with Deeplabv3 (Pillow, OpenCV) had no support for data with higher bands than 3-bands or data types above Uint8. While the second criteria could be easily fixed by scaling back all pixel data to a new range (Uint8), the first required for a selection of bands to be made that could best be used to identify palm trees.

To select 3 bands, the decision was made to look at the paper by Rodriguez et al and pick some the best performing bands from that study. These 3 bands would be used for the follow up algorithms used to pre-process the spectral data. For coconut palm trees, the best IoU scores were obtained for the use of all spectral data (both RGB and non RGB) [15]. The best density MSE and MAE were obtained using all data as well.

The choice thus needed to be a mix of RGB and non RGB bands. Preferably you want to use data without further steps applied to increase errors in the data, so the bands above 10m resolution were not picked due to the bilinear interpolation applied to them. The choice was made for a red and blue band and a NIR band, due to their prevalence in vegetation mapping while meeting the requirements while also including both RGB and non RGB data that way [11], [29]. The chosen bands were bands 2, 4 and 8, whose characteristics are shown in table 2:

Table 2: Sentinel-2 Bands: Central wavelengths of the spectral bands and their spatial resolutions. Source (https://www.satimagingcorp.com/satellite-sensors/other-satellite-sensors/sentinel-2a/)

Sentinel-2 Bands	Central Wavelength (µm)	Resolution (m)
Band 1 - Coastal aerosol	0.443	60
Band 2 - Blue	0.490	10
Band 3 - Green	0.560	10
Band 4 - Red	0.665	10
Band 5 - Vegetation Red Edge	0.705	20
Band 6 - Vegetation Red Edge	0.740	20
Band 7 - Vegetation Red Edge	0.783	20
Band 8 - NIR	0.842	10
Band 8A - Vegetation Red Edge	0.865	20
Band 9 - Water vapour	0.945	60
Band 10 - SWIR - Cirrus	1.375	60
Band 11 - SWIR	1.610	20
Band 12 - SWIR	2.190	20

3.3.2 Ground truth

The second required dataset is a ground truth dataset from which a truth mask could be generated for training and validation of the algorithm. The dataset in question used for this study is from an orthophoto survey by experts in 2018 in which as many Canary island Date Palm trees were georeferenced and their surroundings/species characterized as possible. This was done by the Spanish bureau for Geo-spatial data collection, funded by the European Commission and available through https://opendata.sitcan.es/dataset/mapa-de-palmeras-de-canarias. The dataset contains a total of 555731 palm trees identified through these orthophoto surveys into a set of vector points.

As a vector dataset, the dataset would require several pre-processing steps before it could be utilized in the FCNN model. The FCNN architecture cannot work with vector data inputs so the point data had to be rasterized and the rasterized ground truth and spectral data had to be perfectly aligned so that there was no mismatch between a ground truth pixel and a spectral pixel during model training.

The rasterization of the point dataset was done by using a kernel density function to create a density map in the same projection as the spectral data and ensuring the alignment matched by warping the raster. The ground truth in the bottom right of figure 7 has already gone through these steps.

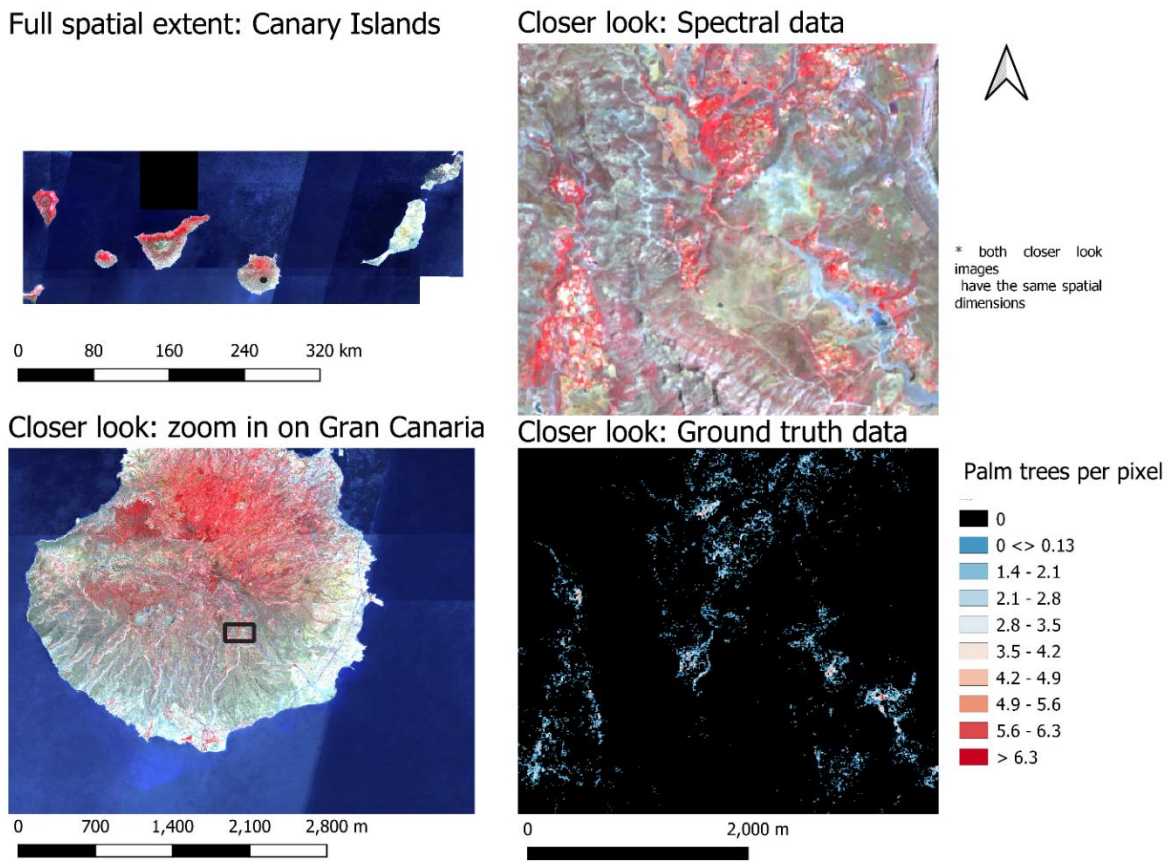

Figure 7: Spatial extent (top left) and spectral visualization of the multispectral (blue, red, NIR false colour) Sentinel-2 dataset, and showcase of the ground truth data (bottom right)

versus the spectral data, the spatial pattern of the ground truth data and the reddish colour in the false colour image seem to match closely

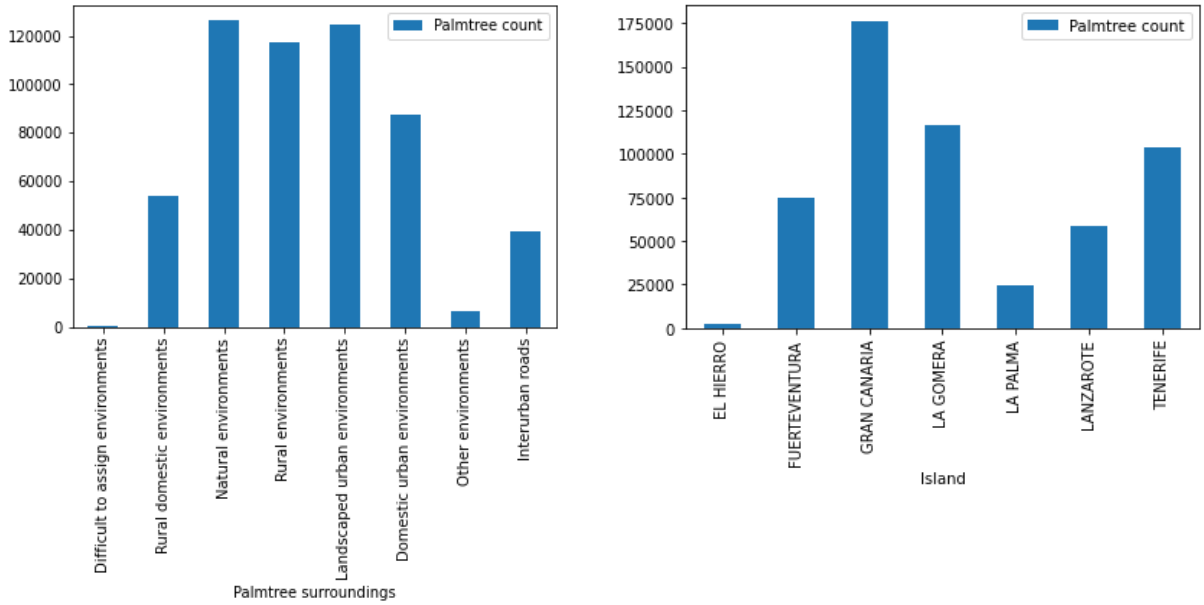

Figure 8: Distributions over type of surroundings as well as distributions over the islands, as obtained through the Spanish Palm tree dataset

3.3.3 Available but unused data

A third set of data was also available at the start of the project, a polygon-based dataset of palm trees located by an algorithm created by Maria Culman Forero from Vito. This algorithm was trained on Pleiades data, for the Palmwatch project. It has been proven already that Palm trees can be identified quite accurately in image sources of higher resolution such as the Pleiades dataset they have [15]**,** and they achieved a good accuracy for their algorithm according to their validation tests. In the end however, this dataset was not used for the training of the model for two reasons. One was that for band width and continuity reasons, it is best to train the model over one region at a time since my access to computing resources where somewhat limited. The second and more important reason was the fear that the number of false positives present within that dataset would negatively impact the predictive power of the final model.

If this dataset had been used the method of this study would have more closely followed the one utilized by Rodriguez et al. [15], the merits of this are discussed in section 5.2.

Figure 9: Zoom in of the area used to examine the data in more detail: See figure 7. Data is Google earth data collected in the summer of 2019

3.4 Methodology

3.4.1 Overview

To get from the raw input data to the final product, multiple scrips spanning several programming languages and computation methods had to be made in sequence. For a visual overview of the methods used and the sequence of scripts, see figure 10.

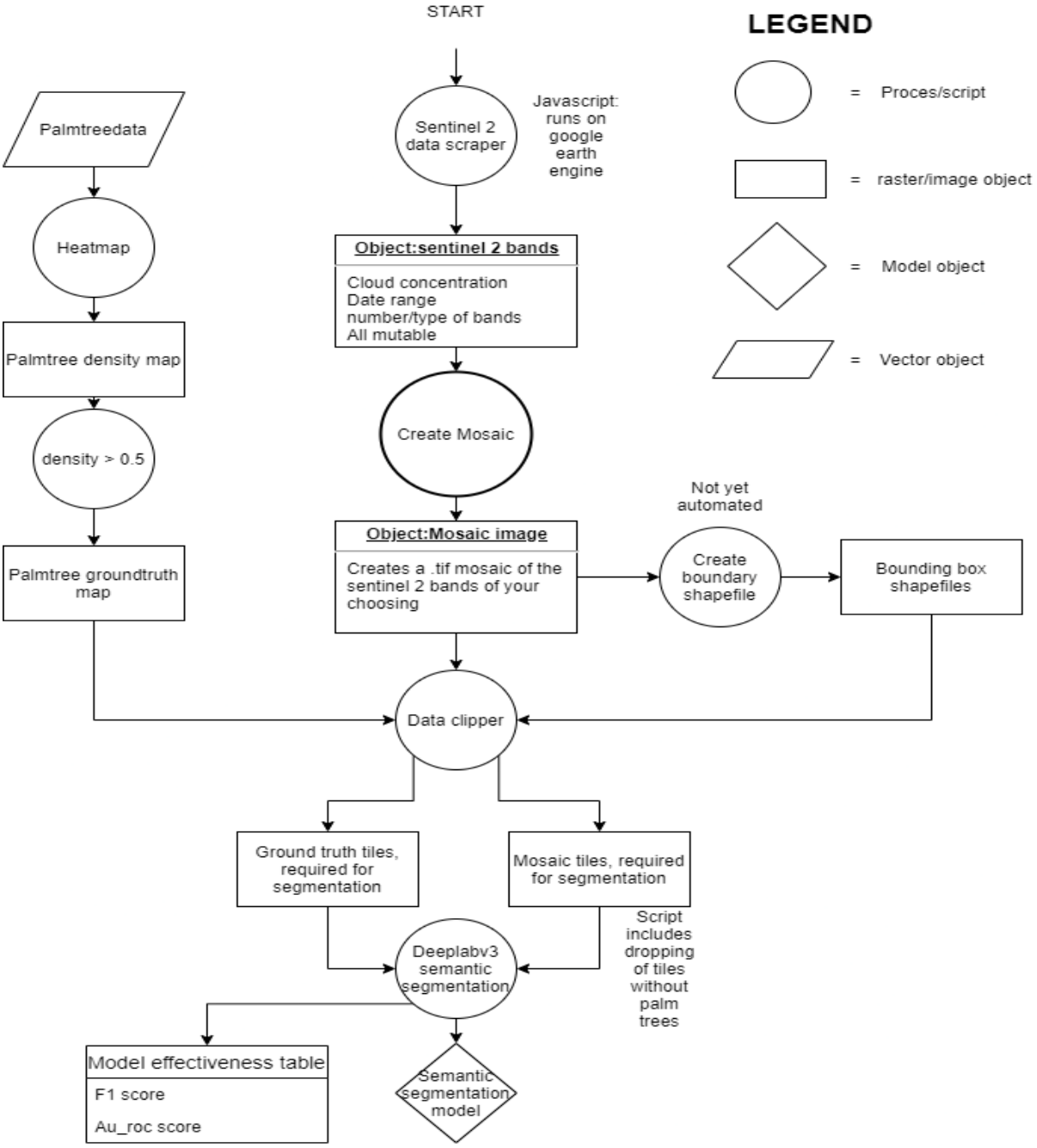

Figure 10: Overview of the full process to go from Sentinel-2 input data to the semantic segmentation model

3.4.2 Acquiring the satellite data and ground truth data

The first script is the one used to obtain the sentinel-2 data required as the imagery source for the algorithm. Obtaining the necessary data is always one of the first steps of any study. For the ground truth dataset, writing a similar script is not necessary as the point-based dataset is obtained through https://opendata.sitcan.es/dataset/mapa-de-palmeras-de-canarias.

The script to obtain the spectral data is a JavaScript implementation on Google earth engine that requires a vector of the area of interest for which imagery needs to be loaded (Figure 10, sentinel-2 data scraper). The start and end date over which a median of the available imagery will be drawn is alterable, as well as the maximum cloudiness of the imagery. For this study the data gathering started at 2017-01-01 and ended at 2018-12-31, as the gathering of orthophotos was done over the course of 2017-2018. The maximum cloudiness was set to less than 10 so that clouds would have low impact on the spectral data. Since the Canary Islands are both a tropical region and at sea, clouded images are commonplace.

The output of the script are several images of the specified area using an image driver of your choosing, a built-in function that forces the data into a specified file type and even with the specified image-bands of your choosing. For this study the chosen bands were the blue, red and NIR bands since they are used for NDVI and are often useful for identifying vegetation, though its lack of account for other environmental factors could hamper model accuracy [29].

3.4.3 Data mosaicking and preparing the ground truth mask

The produced images have overlapping areas that need to be sorted out, since the intent is to work with one spectral coverage image for the whole study area. Creating this mosaic has the side effect of producing an intermediate dataset over which the following processing steps can run a single file containing the full dataset, instead of iterating over loose bands and images.

The images are loaded into a python script that stitches together the images into a mosaic (figure 10, Create mosaic), using the Rasterio module which is a wrapper based around GDAL. While the Data Scraper script created a single set of images that provided full coverage of the study area, overlap between the images still existed. These images were merged after ordering them on acquisition through the reverse painter's algorithm (don't overwrite pixels that already contain data). The finished mosaic is used for the training of the final model is used in later intermediate steps.

The other dataset, the palm tree point based vector dataset is converted to a raster dataset for compatibility with the used modules and because working with raster data is generally less computationally heavy as working with vector data**.** This is done through a uniform kernel density estimation over zero distance, to transform the point data into a raster of matching resolution to the sentinel-2 imagery (figure 10).

The kernel used is a uniform (rectangular kernel), with points having full weight within their given pixel, and zero weight outside, though with a distance at zero the choice of kernel should be taken so that the value at distance zero is close or equal to one.

. The kernel was implemented through the heatmap plugin within QGIS. The kernel density was calculated using the quartic function shown below:

$$K(u) = \frac{15}{16}(1 - u^2)^{\,2} \qquad \textit{Equation 1:}$$

Where u is one time the used distance

The implementation of this function through the kernel density tool in qgis creates a raster from the vector dataset of a specified cellsize (set as same as spectral data) and counting the number of points from the vector datset overlapping with each cell of this new raster. By setting the distance to 0, it ensures vector points only contribute to the density of the very same pixel they would overlap in the new raster.

For the ground truth mask, since a kernel density label has been used each pixel can be said to contain either one or multiple palm trees, depending on the overlap between the points of the original datasets and the pixels of the density raster. Using this characterization, a simple raster calculation leads to the Boolean ground truth mask used for the algorithm where all pixels containing 1 or more palm trees have the y-label for palm trees. The kernel density map that has been created can be used as a density ground truth map to be used for regression.

From there, a choice can be made to create more specific shapefiles/vector features data of the area of interest before going to the next script, but this is an optional step that is mostly there to save on computation time.

3.4.4 Data splitting and Data augmentation

The spectral dataset created after mosaicking is unsuitable for training of the FCNN in its current format. If it were loaded, the total amount of images to train in would equal one times the amount of image augmentations, which is impossibly low. In this study such a methodology would not have been possible due to GPU limitations, since the amount of RAM needed to atrously convolve the entire mosaic over 6 convolutional layers would far exceed the amount available.

The dataset must be cut into chunks, ideally each containing some actual palm trees in the ground truth dataset.

The data clipper script (figure 10, data clipper) clips the ground truth mask and the mosaic of Sentinel-2 data into matching chunks with matching names, and saves them into 2 separate directories after ascertaining that at least one palm tree is present in the chunk. This readies the spectral data for training and validation of the final model.

The data clipper script works through a focal window operation of a size that can be specified within the script, cutting the large satellite imagery into 256x256 chunks and padding edges where necessary to achieve the 256 by 256-pixel window using nearest neighbour to extrapolate when padding is necessary. Before running this script, it is vital that the pixels of the ground truth and the pixels of the spectral data match 100%, this is achieved through a sub function available within the data clipper script. This function can reproject the data using a nearest neighbour algorithm, so that the original data is retained.

The dataset created this way can easily be subjected to data augmentation. The data can in theory also be kept as .TIF format if you intend to work with >3 banded data. this would require the use of GDAL instead of Pillow as an image handler, which would require major rewriting of later modules. The current data clipper saves the output images as .png with corresponding .xml world files.

Since the number of images within the image dataset might still prove small, data augmentation is used to artificially increase the pool of data to train on. By using random horizontal and vertical flips based on a random seed during the training phase of the algorithm a commonly used technique in data augmentation. The model will treat the flipped images as if they are new samples thus seemingly inflating the training data without the need of further

remote sensing imagery and subsequently improving the training as well [30]. This is done as part of the final script, and artificially doubled the amount of training data (figure 10, Deeplabv3 segmentation model).

3.4.5 Creating the Deeplabv3 FCNN architecture

The FCNN used for this study is based on Deeplabv3 architecture, the same architecture used by Rodriguez et al. [15]. Deeplabv3 makes use of atrous convolutions, a technique which adjusts the field of view of filters used as well as controlling the resolution of the filters used for semantic image segmentation to better handle object at multiple scales [21]. Specifically Atrous Spatial Pyramid Pooling (ASPP) is used as an atrous improvement over the known successful technique of spatial pyramid pooling [31], to try and identify both separate as well as cluster of Palm Trees effectively. This capability combined with the fact that this architecture scored well in the paper by Rodriguez et al. factored into the decision of its choice.

The backbone network build into the ASPP framework of Deeplabv3 is a Resnet-50 FCNN backbone that has been tailor made specifically to work well with Deeplabv3, available from the Torchvision GitHub repository, a showcase of how atrous convolutions within the backbone work can be seen in figure 3.

The pretrained versions of this backbone come trained on the Microsoft COCO dataset, a dataset that mainly consists of highly detailed imagery of everyday objects in their natural context [32]. The imagery used for this study, which contains spectral imagery of red, blue and near infrared bandwidths, all taken from space is highly mismatched with the COCO dataset, making the pretrained version of the model unusable as a material for transfer learning.

The FCNN architecture is build up as follows: First the input data goes through 4 convolutional layers plus batch normalization used for the Atrous Spatial Pyramid pooling, followed by the ASPP of these 4 layers. This is followed by two more convolutional layers and batch normalizations. Finally, the model is then activated using a sigmoid activation function and the loss is calculated. Since the resolution is already a limiting factor with the use of imagery from space, steps have been taken to reduce further loss of spatial resolution by setting the stride of all the convolutional layers to 1. All these steps are performed in the Deeplabv3 semantic segmentation script (Figure 10)

3.4.6 Used loss function

The Loss function used is a simple root mean squared error loss functions, used before to train semantic segmentation by forcing a minimal difference between the ground truth and the predictions using an optimization scheme that will be elaborated further in a later paragraph. The following mean square error loss function was used:

$$MSE = \frac{1}{n}\sum_{i=0}^{n}\left(Y_i - \hat{Y}_i\right)^2 \qquad \textit{Equation 2:}$$

Where n = sum of file pixels in training image, Y is the image pixel value, $\hat{Y}$ is the ground truth value. By punishing the model for having a large gap between the predicted pixel value and the ground truth value, the regression is fitted to minimize this gap as much as possible.

3.4.7 Scoring metrics

The predicted values are then used for multiple scores. This being the F1 value, a combination of precision and recall scores used as a general measure of model precision, the Area under the ROC curve score (AUROC) to tell if the model is able to rank regression scores correctly, as well as r1 scores to check the accuracy of the regression.

3.4.7.1 F1

The F1 score used in this study is the harmonic mean of the model precision and the model recall [33], and advocated as a single measure score to capture the effectiveness of systems. While the score ignores the effect of true negatives during its calculation it's an often-used metric to test model effectiveness. Due to its dependent relationship between recall and precision and the way it contains information on both those two metrics were left out of this study. The following equation was used to calculate F1 as a harmonic mean between model precision and recall.

$$F1 = \frac{tp}{(tp+(fn+fp))/2} = 2 * \frac{precision * recall}{precision + recall} \qquad \textit{Equation 3:}$$

Where tp = true positives, fn = false negatives and fp = false positives

3.4.7.2 Au_ROC score

The area under the ROC (receiver operating characteristics) curve score (Au_ROC) is a technique commonly used to check the performance of any classification model [34], [35]. Like the name suggests, its literally the area under the ROC curve, which is the false positive rate (False positive/(false positive + true negative)) versus the true positive rate (True positive/(True positive + False Positive)), and thus the integral of that curve. The Au_ROC score was calculated using the following integral function:

$$Au_{ROC} = \int_{x=0}^{1} TPR\left(FPR^{-1}(x)\right)dx \qquad \textit{Equation 4:}$$

with TPR as the true positive rate (recall) and FPR as the false positive rate. The results of the test for this study indicate the probability any pixel containing a true ground truth statement (correctly) classified with greater suspicion than a randomly chosen pixel with a false ground truth flag [26].

3.4.8 Optimization

Optimization of the training is done using ADAM optimization, a recent innovation that should combine both the positive traits of two other methods of stochastic gradient descent, namely Adaptive Gradient algorithm (good for sparse gradients, computer vision problems) and Root mean squares propagation (works well on noisy data) [37]. ADAM is currently a very good option to allow for quick convergence of algorithms and for efficiently solving Deep learning problems [37]**.**

The model training will go through 25 epochs, with the data augmentation of random vertical and horizontal flips being applied based on a random seed before each new epoch. Each epoch of training will result in a FCNN model with trained weights,

ADAM will adjust the adaptive learning rates for each parameter (band) and the step sizes taken for these parameters based on the first and second gradient moments within their gradient descent each epoch, attempting to reach convergence. [37]. Figure 11 shows that when compared to other stochastic optimizers adam produes very favorable results.

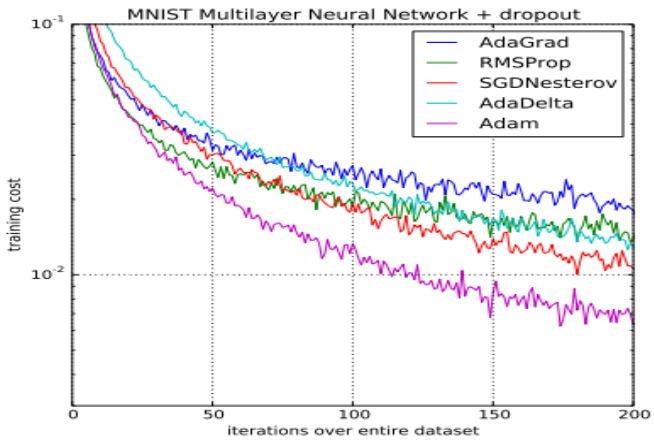

Figure 11:Comparison of ADAM versus other model optimizers using a training cost function spanning multiple iterations over the same dataset. Source [37]

4 Results

4.1 Suitability of the used ground truth and satellite data for FCNN semantic segmentation

To assess whether the pre-conditions are laid for a good training environment of the FCNN model, both the ground truth data as well as the satellite data need to be assessed for their suitability in model training. The suitability of the ground truth data and the Sentinel-2 data will be assessed in order.

4.1.1 Ground truth data

Determining whether the ground truth data is suitable for training the model can be done by assessing if there is bias prevalent within the datasets to certain areas. To train the model well on the spectral data, as many palm trees as possible have to be correctly annotated within the ground truth data, to reduce the amount of time the model is punished for detecting a palm tree in pixel where in reality a palm tree is present but for which the ground truth data has inaccurate/missing annotations.

In the figures 12-14 below several regions on multiple islands have been selected to check for possible bias within the ground truth datasets, by comparing ground truth data for pixels with at least one palm tree with OpenStreetMap data to look for spatial similarities in distribution. From the left most part of each of the three image, it is possible to grasp the layout of urban centres and natural parks, and on the right side these can be compared to the density and distribution of palm trees within the ground truth dataset.

In figure 12 a location on Gran Canaria is inspected that has a spread of both natural reserves (bottom right, centre of both images) and some urbanisation (Moya and Firgas, top right of both images). As can be deduced from the ground truth, the areas with high ground truth density of Palm trees are in the urban areas of Moya and especially in Firgas, while the natural areas that are in reality densely vegetated and should contain Palm trees as well are largely black in the ground truth images.

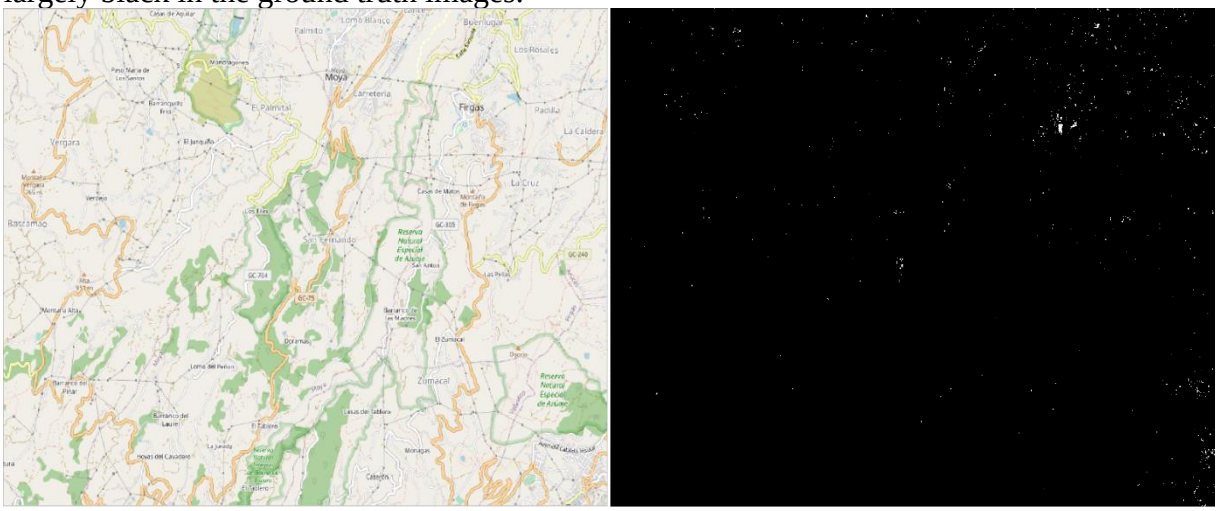

Figure 12:The region around Las Garzas on the mainland of Gran Canaria. This spot was chosen for its prevalence of both natural parks and reservations as well as urban sprawl (though of a more rural area). The image on the left is the OpenStreetMap data, the image on the right is the ground truth. Pixels containing 1 or more palm trees are indicated as white in the ground truth

In figure 13, There once again seems to be a clear spatial link between the existence of urbanized zones and the presence of palm trees within the ground truth images. In the slightly more zoomed in image around the vicinity of Los Christianos on Santa Cruz areas of high palm tree density in the ground truth seem to be nearly strictly relegated to urban areas, with off grid coastal areas or natural areas showing as near black voids on the ground truth map. CIPD's should nearly certainly be present within these areas as these are its natural habitats but looking at the natural areas indicated in green on the left part of figure 13 these seem to have no ground truth data in the right sided image.

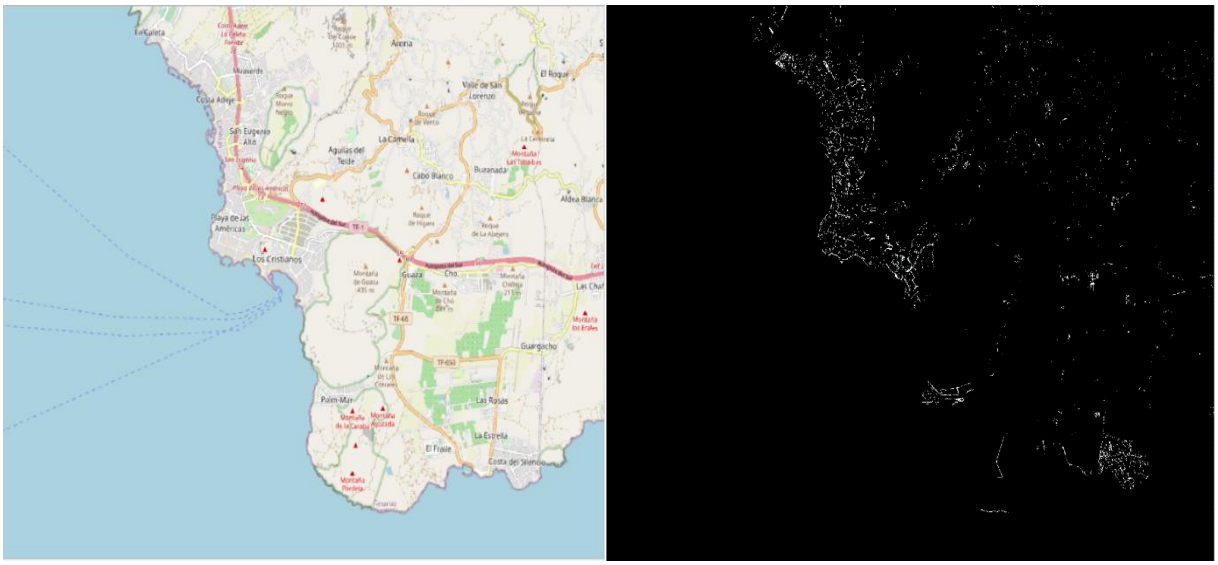

Figure 13: Comparison of OpenStreetMap data with ground truth data in the vicinity of Los Christianos, Santa cruz. The left image contains OpenStreetMap data, while the right one indicates the presence of at least 1 palm tree in a pixel as white within the ground map.

There is also a clear difference in ground truth density between different Islands. While Figure 12 and 13 compare regions on the islands Gran Canaria and Santa Cruz respectively (Some of the more populated islands among the Canary Islands) figure 14 shows a region on the island El Hierro, a far smaller and less populated island. Compared to the previous two figures, it can be clearly seen that the amount of annotated Palm trees is lower than on the other two islands. El Hierro contains a larger proportion of unpopulated or sparsely populated regions and once again these areas show up as mostly black within the ground truth images. This is odd, considering the westernmost Island El Hierro has a very temperate tropical climate, ideal for the growth of CIPD's, a larger density of CIPD's are thus expected in figure 14 but the ground truth image shows a very low palm tree density [5].

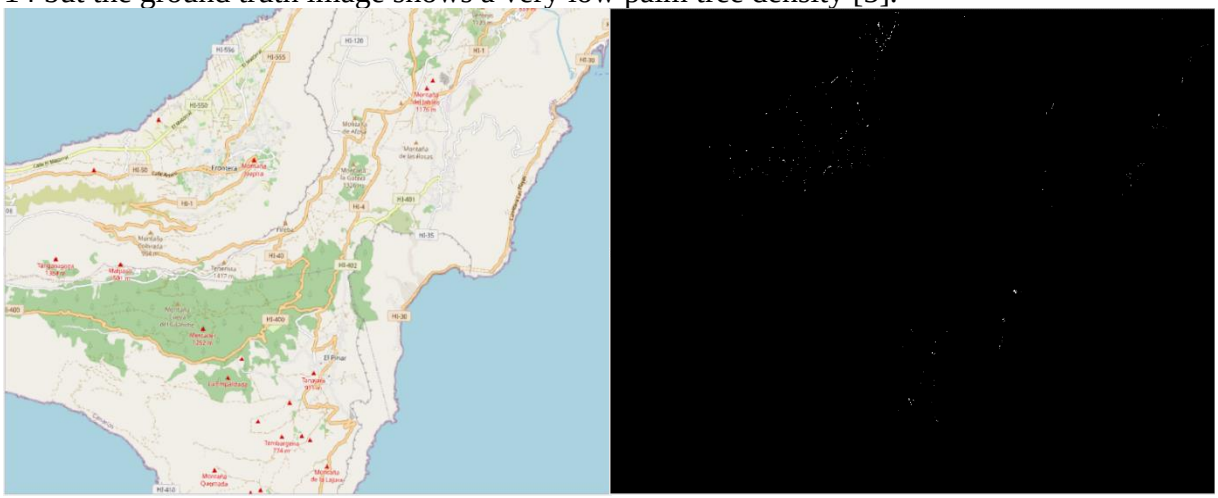

Figure 14: Comparison of OpenStreetMap and ground truth data on the island of El Hierro. OSM data is on the left, ground truth data on the right. Pixels containing at least 1 palm tree show as white in the ground truth data.

4.1.2 Sentinel-2 data

The suitability of the sentinel-2 data for the model training can be approached from two main directions. The first one is the predictive power of reflectance as a means to detect Palm trees. For this it is important to know the spectral range and centre of the used multispectral data as well as their relevance in palm tree detection.

Due to image handling constraints encountered while working on the model architecture, image handling had to be done using the PIL python module, which in its current iteration has no support for the handling of images and tensors above 3 spectral bands. Due to this a selection had to be made during the mosaicking script (figure 10) to reduce the spectral data from the 12 bands present (table 2) to 3 bands. Based on results of the model by Rodriguez et al. and the ability of combined NIR and blue information to distinguish vegetation from other types of land use, band two three and four where chosen for the reduced dataset [15], [38].

The second factor of suitability would be the size of the dataset. The large number of unknowns within FCNN's require an equally large number of predictors. After application of the data-clipping script the size of the sentinel dataset was 980 images 256 by 256 pixels in size, this amount would be doubled due to the horizontal and vertical flips applied to augment the data to a total size of 1960 images. This dataset would then be divided into a ratio of 0.8 training data and 0.2 test data.

The use of a dataset for which the locations of *Phoenix Canariensis* are only known in a fixed point in time (2018) and limitations imposed due to memory constraints only allow for a single full coverage of spectral data taken as a median of cloud free Sentinel-2 data around the gathering date of the ground truth data.

4.2 Pre-processing the data and building the FCNN model

A very large part of the work on this study in the end went to the creation of the pre-processing steps and scripts visible in figure 10. While the work of Rodriguez et al. and the implementation comments for the used modules on GitHub (Deeplabv3) could be relied upon to broadly get a picture on how to implement the process to get from the input data to the FCNN model, no immediate tool or all-encompassing guide was available so a large part went into writing new code, debugging and trial and error to get the scripts working.

To make it easier for the next person that intends to create a FCNN for Sentinel-2 imagery, commented versions of all the scripts alluded to in figure 10 will be available in appendix C, including a step by step guide on how to work from the input to the final data. Since some of the pre-processing tasks where done in GIS environments (QGIS for this study), how to implement these steps will be explained there as well.

4.3 Predictive power of the FCNN model

To get a grasp on the predictive power of the created model, the results for the metrics described in the methodology will be evaluated one by one, aided by a visualization (plot) of each metric over the epochs the model was training. The resulting graphs are thus a comparison of the finished models on the previously mentioned scoring metrics for each epoch.

An extra evaluation of the Training loss versus the Test loss will also be done to see if anything of note can be discovered on the training process of the model. This data can be used for later evaluations on possible over or underfitting of the data by the FCNN models in the discussion.

An example of how the ground truth compares versus the prediction is shown in figure 15 below. A large amount of erroneously predicted palm trees are concentrated in the northern part of the Gran Canarias. This area contains a large amount of urbanisation compared to the south, the model is clearly correlating the presence of built up area and the presence of Palm trees a lot. In the south, where there are fewer but quite dense clusters of urban area there is more overlap between the model and the ground truth.

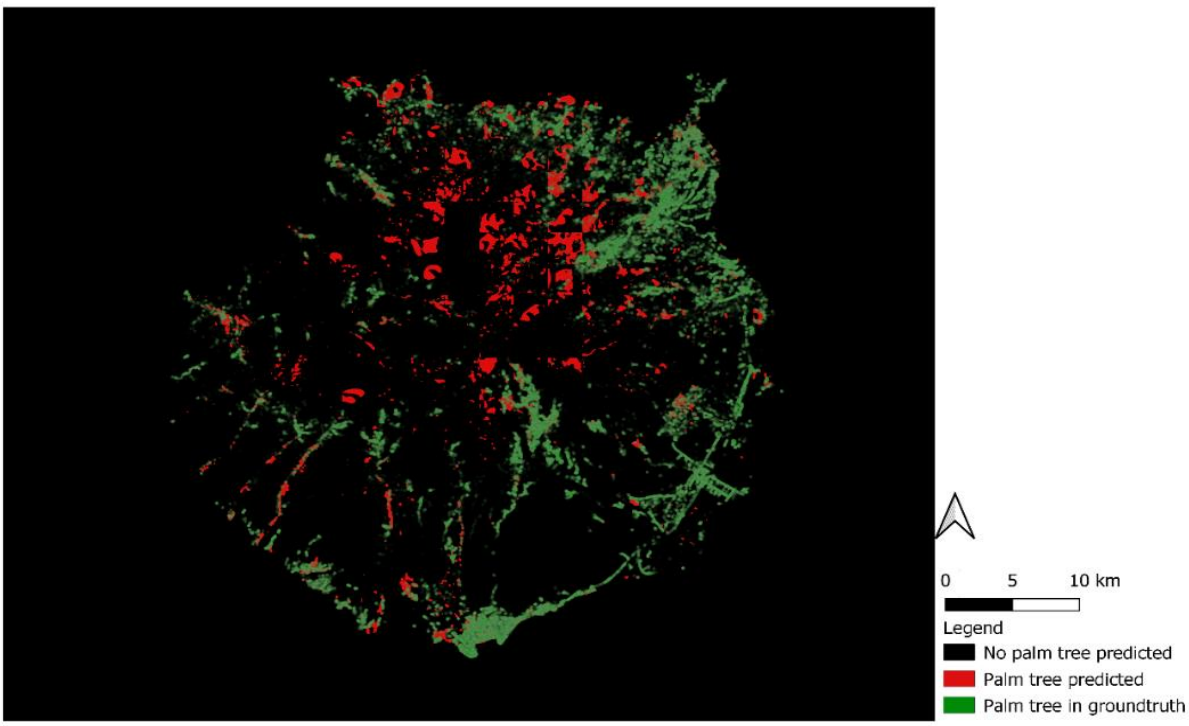

Figure 15:Model predictions and ground truth palm tree locations on the island of Gran Canaria

4.3.1 F1 score

From figure 16. containing the F1 scores of the models from each epoch, several things can be seen. A high F1 score (0.6+ taking other studies results on precision and recall as a benchmark [7], [15]) means high model accuracy and recall (see equation 3) and serves as a positive indicator of model fidelity.

Especially from the F1 scores on the training data in figure 16 over the largest part of the upwards an upward trend in F1 score can be deduced, with a spike downwards after the model from the 23rd epoch. This upward trend does not seem to be shared by the F1 scores on the test dataset, while the train F1 score rises between epoch 5 and 15 the test F1 score lowers. There does seem to be some overlap between the peak F1 scores for both models.

The biggest takeaway from the scores is the overall range. The F1 scores for the models range between 0.014 and 0.018 for the training dataset and between 0.011 and 0.028 for the test dataset, meaning the model at its best has around 2% recall and precision.

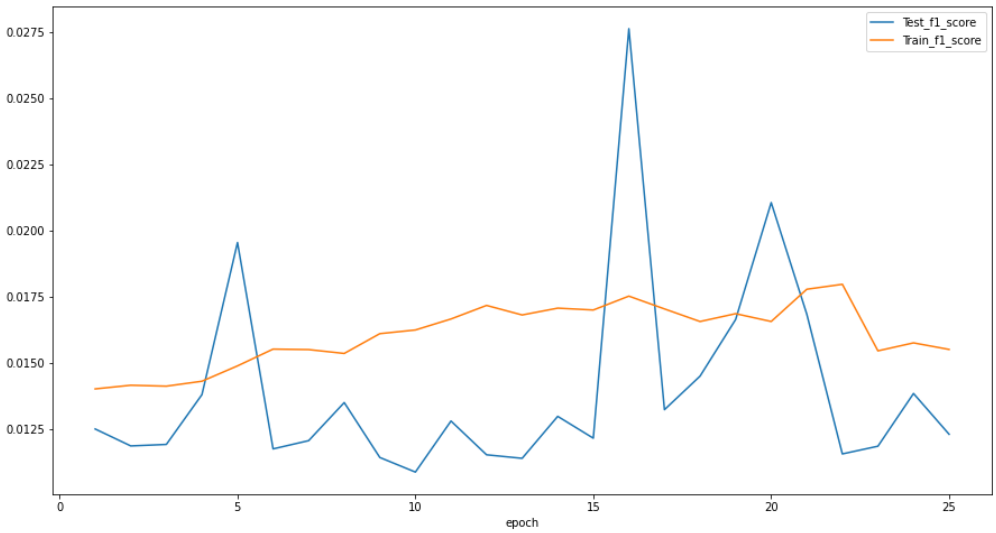

Figure 16: F1 scores for the model during each epoch. X-axis contains the epoch number and the y-axis contains the F1 score. To better illustrate F1 trends, the origin of the y-axis has been set to 0.0125. The blue line portrays the F1 score on the test dataset while the orange line contains the F1 score for the training dataset.

4.3.2 Au_roc score

The Au_roc score highlights whether pixels containing a true statement (at least 1 palm tree within the ground truth dataset) are correctly classified with greater probability than a random pixel containing a false statement within the ground truth (no palm trees). As such since the mode classifies two classes (palm tree, no palm tree), a completely random model would have an Au_roc score of 0.5, and higher Au_roc scores indicate improvements in classification effectiveness.

Over the course of the epochs, an upward trend is visible for both the test as well as the training datasets, until once again the scores drop in around the 23rd epoch. In contrast to the F1 scores from figure 17, the Au_roc scores for the test dataset largely follow the same trends as the Au_roc scores for the training dataset.

The overall range of the Au_roc scores for the test dataset range between 0.55 and 0.79 for the train dataset and between 0.53 and 0.74 for the test dataset.

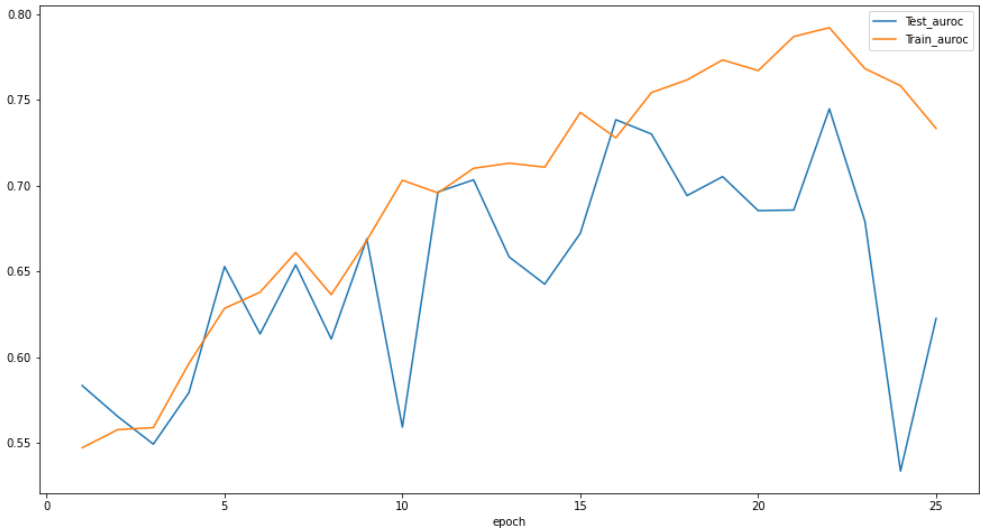

Figure 17: Au_roc scores for the model during each epoch. X-axis contains the epoch number and the y-axis contains the Au_roc score. To better illustrate Au_roc trends, the origin of the y-axis has been set to 0.55. The blue line portrays the Au_roc score on the test dataset while the orange line contains the Au_roc score for the training dataset.

4.3.3 Train versus test loss reduction

The model architecture used uses a mean squared error loss to get an accurate regression for the presence of palm trees within the Sentinel-2 imagery (for more information see equation 2). The model gets punished for high differences between the ground truth density and the predicted density, and these would show as a high loss in the graph. Good regressions thus lead to lower losses, and if the model progresses towards higher accuracy over the epochs it would show as having a downward trend in figure 18.

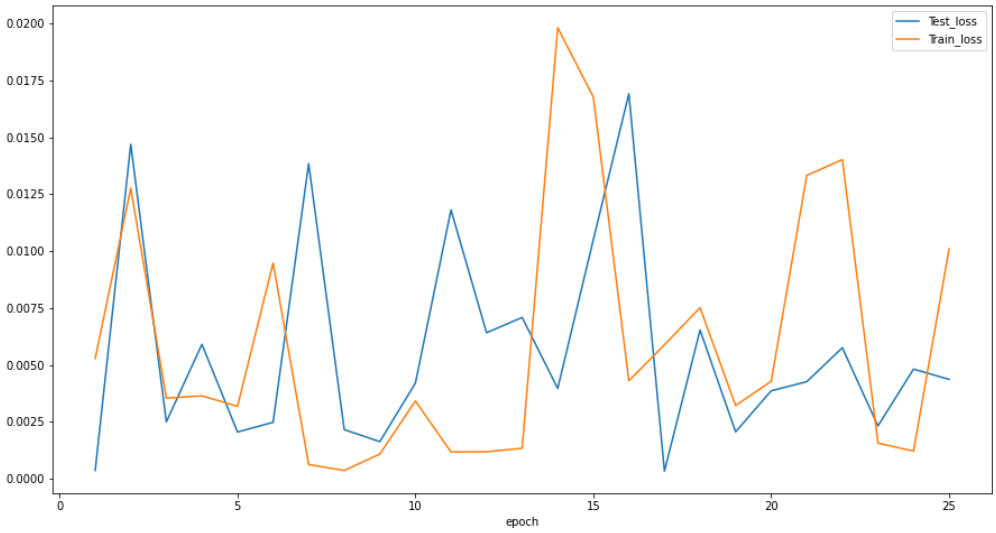

Figure 18: Train versus test loss reduction over each model epoch. Train loss is in orange, while test loss is in blue. The y-axis contains the mean square error losses for the models, the x-axis shows the course of the training epochs

4.4 Land use as a possible factor of bias in model predictions

While this paragraph at first might seem like paragraph 4.1, the contents discussed will be markedly different. While during paragraph 4.1 an overview of the data was made as a check for bias in the data gathered for the ground truth masks, this section instead looks for bias within the model predictions. To do this, in figure 20 comparisons are made between the Sentinel-2 data, the ground truth masks and the model predictions for an urban area, a rural/natural area and a coastal area. The locations of these sampling areas are shown below in figure 19.

The sampling areas were chosen by first looking for three possible sampling areas for the three different land use types discussed (coastal, urban, natural area) based on the natural habit of the CIPD [5]. Going from island to island a total of 10 regions (4 urban, 3 coastal, 3 natural) were chosen by searching for promising areas through google earth and assessing whether these locations had enough ground truth data. The chosen urban, vegetated and coastal area where randomly drawn from these 10 based on their assignation.

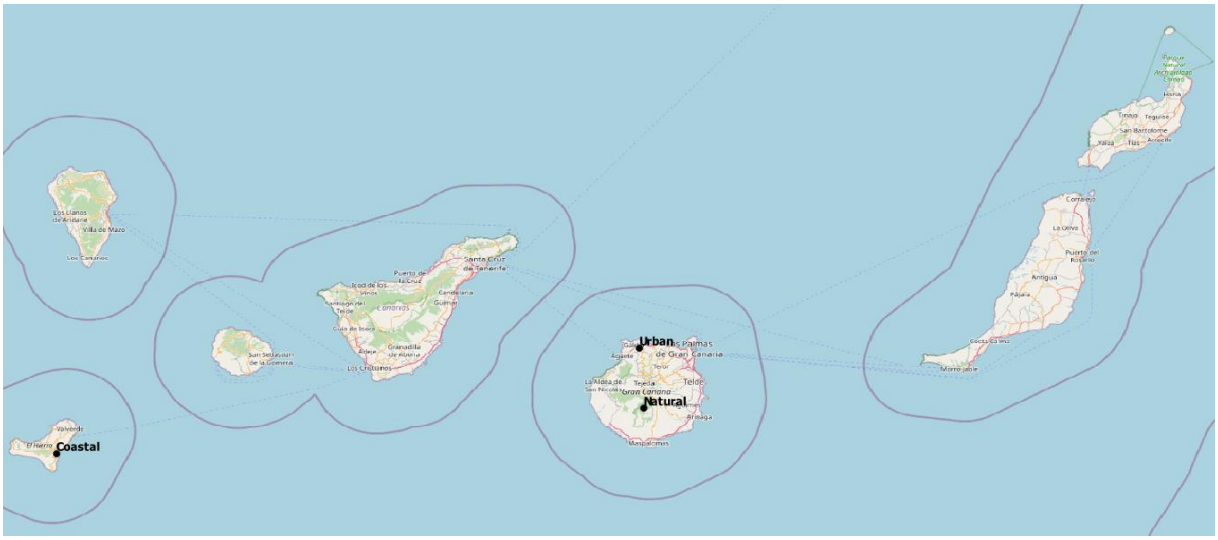

Figure 19: Sampling areas for an urban, natural and Coastal region. The Coastal region is in Las Playas de la arena (El Hierro), the urban region is the area around Gáldar (Gran Canaria) and the Natural area is at the edge of Las Tederas (Gran Canaria).

The threshold for detections (when the purple turns to yellow in the right column of figure 20**,** has been set to 0.03 to allow for at least some analysis of spatial patterns since higher thresholds lead to the predictions in figure 20 being entirely purple.

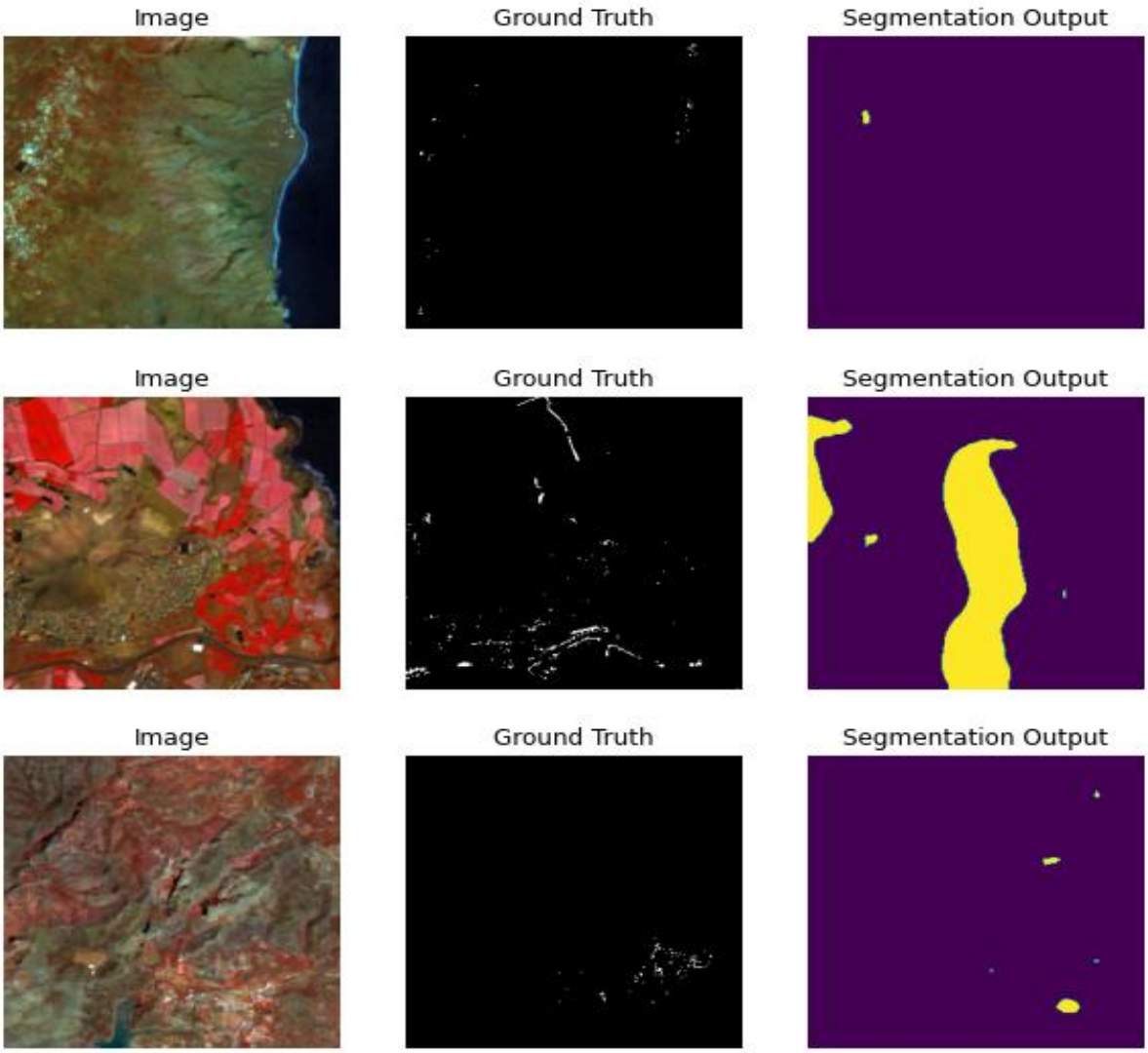

Figure 20: Sentinel-2 (left column, false colour (Blue, red, NIR)) and ground truth data (middle column) compared to the model predictions (Right column, yellow = palm tree presence predicted, purple means no palm trees, detection threshold set as 0.03 (higher thresholds leads to no detections in most areas)). The locations are a selection of Natural coastal area, urban area and mountainous rural area respectively (top, middle bottom row respectively).

From the nine images visible in figure 20, three images that included other types of variation in landscape as well (built up beaches, mountains, rivers) were chosen to include some of that variation and how the model reacts to it as well. The left and centre columns in figure 16 are the outputs of the data clipper script referred to in the methodology section (see figure 4 and section 3.4.4), while the right column contains output from evaluating best working FCNN model over the images in the left column.

Does the model behave similarly for these different land use types? This section will compare results for various land use types visible within figure 20 as a visual aid. Furthermore, the bar

charts of figure 21 below that have calculated prediction statistics over the whole of the dataset will be used to better answer the question

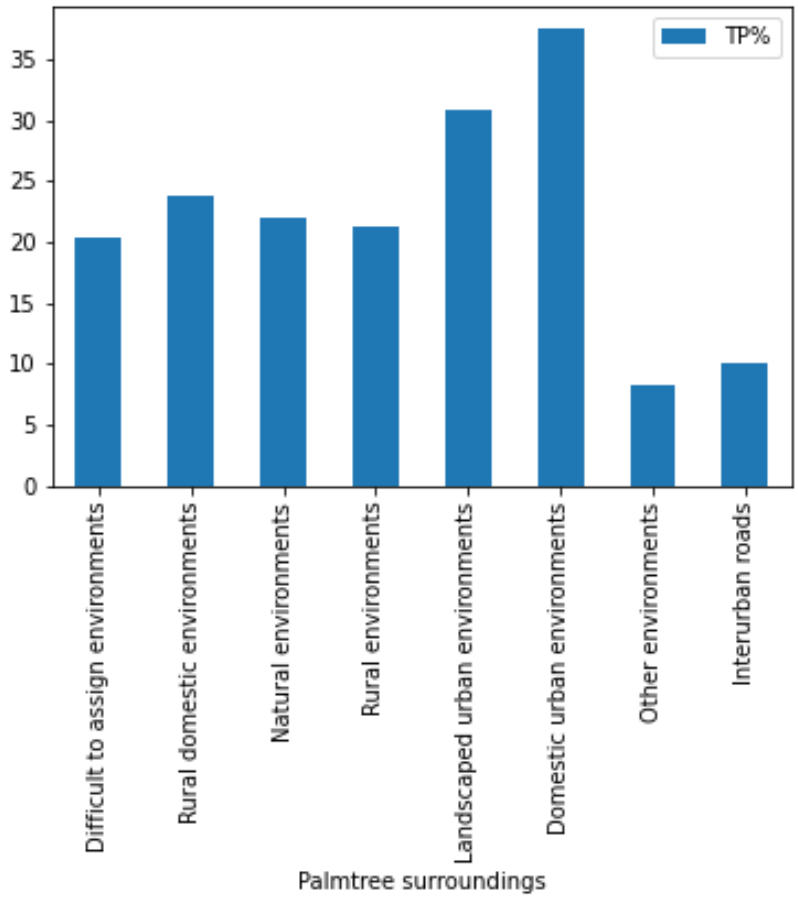

Figure 21: Percentage of true positive predictions of palm trees for detection threshold = 0.03 over the whole dataset

4.4.1 Urban areas

The middle row of figure 20 shows a comparison between the 3 aforementioned data streams of an urban area. The urban sprawl is mostly located around the river that's visible in the bottom part of the images in this central row of images. Another urban centre can be seen in the bottom row of figure 20 in the bottom right, though the rest of this image is largely natural/rural areas.

In both cases the model has at least some positive predictions within these urban areas, though the predictions come in the form of a large uniform blob instead of the more spatially distinct ground truth. In the images of the middle row the size of the area supposedly containing palm trees is incredibly overproportioned, while in the bottom row the predicted area is smaller than the area containing palm trees in the ground truth. It is also slightly off centre in this one.

From figure 21 it is visible that in comparison to any other surroundings for the palm trees, the amount of true positive predictions for palm trees in urban landscaped and urban domestic environments is higher.

4.4.2 Rural/natural areas

Rural and natural areas can be best seen in the top and bottom rows of figure 20**.** Finding rural and especially natural areas with a high degree of annotated Palm trees proved very difficult within this dataset, which should come as no surprise following the results detailed within paragraph 4.1 The top row contains some positive ground truth along a road through a mountainous natural area, while the bottom row contains large parts of natural area aside from the bottom right section. Within these areas, the model does seem to have some overlap with the ground truths of these areas, though most ground truth in natural and rural areas are sparse within the dataset and the model seems to predict negative in these areas most of the time.

From figure 21 it can be seen that in rural or natural areas the model scores a lower amount of true positive predictions than within urban environments.

4.4.3 Coastal zones

Similarly to the rural and natural areas, finding positive Palm tree data that was reasonably densely populated was difficult for most coastal zones within the dataset, as data here was very sparse even though it's a favoured habitat of the *Phoenix Canariensis* [5]. Within figure 20 coastal zones are present on the right side of each image in the top row and on the top right side of each image in the middle row.

In both locations the model predicts no presence of Palm trees, though it should have done so in the images of the top row of figure 20 according to the ground truth data.

Since there is no distinction made for coastal zones within the original dataset, it is not really possible to build a conclusion on the predictions on these regions from figure 21.

5 Discussion

In this section there will be a further evaluation and assessment of the data and information first discussed in the results section. The research questions will be assessed on the basis of the obtained research results and on literature review of prior works in the field, obtained prior and during the progression of said research. This section is built up in a way to address each research question in order.

5.1 The suitability of used data for model training

The results from the assessment of section 4.1 (suitability of the used ground truth and satellite data for model training) as well as the difficulty in finding areas of comparison for section 4.4 (Land use as a factor of bias in model predictions) paint a picture of heavy bias in labelling density in favour of urban areas and other areas that are easier to assess such as palm trees lining interurban roads. The notable lack of labels in coastal areas and natural areas (especially forested regions) despite these being some of the most common habitats of the *Phoenix Canariensis* [5] imply a large amount of mislabelled (false negative) pixels within these regions. For natural areas that have been validated through the ground truth it can be seen in figure 20 that they are some of the least correctly classified areas. The only specifically named area scoring worse are interurban roads, this lack of valid classification for interurban roads matches that of the one by Rodriguez et al. They noted that high density areas surrounded by low density areas (this is the case for roads in this dataset) tend to be underestimated [15].

Since the region proposal networks and the dilation present within Deeplabv3 are designed to take instance context into account [21], the spectral characteristics of the surroundings of palm trees will attribute to the classification results. Even for single pixel classifications, due to the object/pixel ratio of the *Phoenix Canariensis* trees being between 0.6-0.8 for full grown specimens more than just canopy reflectance will be present in the pixels. If the context or area around the labelled palm trees is thus skewed towards a certain type of land use such as built up area within this dataset, overfitting for such regions in favour of other land use types is likely to happen [39].

The dataset could furthermore be called imbalanced due to the large difference in sample size between the majority class (no palm tree) versus the minority class (palm tree present) [40]. While this imbalance is partially addressed through region proposals applied by the region proposal network built into the Deeplabv3 architecture [41], this imbalance will not be addressed completely and might still lead to overfitting in favour of no presence of palm trees since the skewed distribution of classes will train the classification model in favour of the majority class [39].

While the suitability of the Sentinel-2 data is likely sufficient as proven by its successful use in the paper of Rodriguez et al. who worked with a similar amount of objects (537500 palms, versus this studies 555731) [15], the size and amount of spectral information of the data set used in this research might contribute negatively to the model results. Both this study and the study by Rodriguez only picked one set of spectral data close to the ground truth creation date. With the amount of palm trees in both studies being in similar orders of magnitude, due to the lack of information on the pre-processing steps taken by Rodriguez et al. for loading in this data it is unclear what steps he took to alleviate this data size problem [15]. For this study, the dataset had been cut into 980 image clips, but it is unknown how many where in the dataset by Rodriguez et al, or whether he took a similar step at all [15].

Due to the constraints imposed by the used image handling modules, and the resulting spectral data consisting of only Sentinel-2 band two three and four (table 2) some reflectance data that could improve the detection of *Phoenix Canariensis* is missing [7], [11], [15]. This would make it harder for the classification model to distinguish the palm trees from other objects in the background compared to model created by Rodriguez et al. who used all available spectral data [15]. The amount of data this studies model has access to allowed it to draw from fewer spectral correlations and linkages between the ground truth and spectral data than the model by Rodriguez et al.

The size of the dataset is also a major problem. The used Deeplabv3 model architecture contains 2048 feature channels. Even with the data augmentation present within this study used to increase the size of the training data set to improve results [30], the size of the inflated dataset amounts to 1960, which is lower than the amount of feature channels even before splitting the data into a test and training dataset, which makes the model underdetermined.

5.2 Translatability of the approach used by "Counting the uncountable"

As was described in the literature review section before, the methodology applied by Rodriguez et al. [15] could largely be applied to this dataset. The largest difference is however in the ground truth dataset. For their ground truth dataset, high resolution satellite imagery is manually labelled to train a R-CNN object detection model, resulting in polygons/ boundary boxes that contain a palm tree. For use in a ground truth mask, such a dataset would likely need to be converted using polygon to raster conversions before loading it into the final model for training.

The dataset within this study however has a point based annotated dataset of Palm tree locations. Since the exact geometries of the palm trees in question is not known, the best way to convert from is to count the number of points that overlap with each pixel in the Satellite Imagery to create a ground truth dataset. This conversion was achieved by using kernel density equations where the distance for the function was set to be equal to 0 and the cell size equal to the spatial resolution of the Sentinel-2 data.

Once the ground truth is created, the data available to both this study and the study done in the "Counting the uncountable" paper is of similar shape, and thus translating their semantic segmentation methodology and model architecture was possible, a near identical model architecture was achieved.

Had the dataset used in section 3.3.3 been used instead, the ground truth used would have nearly identically matched the one used in this study. Whether this would have improved the overall model fidelity is uncertain. It might have less bias in palm tree classifications but would have many falsely identified Palm trees as well.

5.3 The fidelity and usability of the trained model

With the metrics obtained and visualized in figures 16-18 it is now possible to assess the fidelity and usability of the trained model for its designated task of providing per pixel assessments of the presence of palm trees, and to compare these results to similar semantic segmentation tasks applied in other studies.

The F1 scores for the testing results within this study is 0.02763, or 2.763% for the model with optimal F1 results. This indicates a precision and accuracy below 5%, indicating the

model has extremely poor predictive power [33, p. 1]. Since the model tends to have difficulties of assigning palm trees in areas that do not have high CIPD concentrations or are not urban centres, large amounts of palm trees are incorrectly assigned. The 0.5 threshold derived from the study by Rodriguez et al as a palm tree has rarely ever been predicted on the location a palm tree existed in the ground truth dataset, likely as a result of the far lower prediction values created by the FCNN model of this study (see figure 20, which uses a threshold of 0.03 instead).

F1 score is also often denoted as dice score, which means the obtained F1 scores can be compared to dice scores obtained in other studies. The first study to compare the results to is the study on which most of the model architecture was based, the study by Rodriguez et al. [15]. Their study obtained a precision and recall of 0.795 and 0.796 which would mean an F1 score of 0.796 for their highest scoring model. In another study using Satellite data for the training of FCNN models, dice (F1) scores of 0.45 were achieved for SegNet architecture, 0.68 for TLinkNet and 0.76 for U-net [16]. Comparatively, the results of the model from this study fall way short compared to those hallmarks.

The Au_roc scores of the trained model achieves a score of 0.74 on the test set. For a random selection for a binary classification, the Au_roc score would have been 0.5, and an increase towards 1 would indicate improved tendency to correctly classify pixels containing a palm tree versus pixels containing no palm trees in the ground truth data [36]. This tendency might be due to the way data is skewed towards urban centres, and the models tendency to broadly predict presence of Palm trees in urban centres visible from figure 20 and the data in figure 21, so while the score looks good on paper it might mean the model has simply accurately responded to the bias present in the ground truth dataset, as expected [39].

From the graph of train versus test loss in figure 18, it can be seen that the ADAM optimization has not yet reached a point where it could be said to have reached optimal hyperparameter settings and weights yet, as the behaviour of the test loss and training loss is still extremely noisy, this might be possible by training it for many more epochs but with the underdetermined and biased dataset of this study it might never happen as well [37]. The test loss is higher than the training loss for most of the epochs, indicating overfitting on the training dataset. The predicted regressions in most pixels where very low as well, which is most likely due to the severe over representation of the class with 0 palm trees versus the class with any amount of palm trees, which would tend to punish the model for having higher regression results [42].

5.4 Land use bias within model predictions

Likely as a result of bias within the ground truth data (see figure 20,21), the trained model appears to over predict in built-up and urban areas, while it appears to have close to zero predictions in natural areas and coastal zones without infrastructure.

From these results its highly likely that if the model's results are taken at face value, it would imply that the natural habitat of the *Phoenix Canariensis* would be in the vicinity of urban sprawl and roads. While *Phoenix Canariensis* are commonly used as ornamental and decorative vegetation and produced on plantation in the area, the tree should naturally occur regularly in coastal zones and forest within the Canary Islands as well [5]. The result of this bias is likely the ease at which Palm trees in urban centres can be assessed in comparison to palm trees in more rural and natural areas where they will be surrounded by other vegetation.

6 Conclusion and recommendations

6.1 Conclusion

To assess whether it was possible, this study has attempted to create a robust FCNN model to determine the locations of *Phoenix Canariensis* palm trees within Sentinel-2 satellite imagery. Pre-processing the available data and getting the model architecture and training done right took up very large proportions of the time available for this research project but at the very least a final model was successfully created.

Based primarily on previous work by Rodriguez et al. [15] and with the help of online guides in Pytorch coding, a workflow was successfully made all the way from the raw data to the finished model (figure 10).

Based on previous forays into this field of study [15], [22], [23], there was hope for positive results and a usable model for Palm tree detection, but the fidelity of the created model unusable for further mapping operations, having an F1/dice score of only 2.763%.

While the Au_roc score showed some promise at 0.74476., the apparent discrepancy between this metric and the F1 score could be explained in the bias towards urban and built-up regions clearly visible through even visible assessment present in both the ground truth dataset used as the basis and therefore subsequently in the model predictions.

Looking at how the model had been training over the epochs based on its loss graph (figure 18), it could be clearly seen that stabilization by the optimizer had not yet occurred within 25 epochs, and no real trend was visible either, so it is not known if convergence would occur at some point if the model was trained for many multitudes of epochs more (which was difficult due to computational limits such as the limited amount of GPU RAM involved in this study).

With all this combined, though it is a promising avenue of making use of low resolution data to get “high resolution” data, with the level of data leveraged within this study it is not possible to create a high fidelity model to detect *Phoenix Canariensis* trees within Sentinel-2 data.

6.2 Future recommendations

In hindsight for this study, and maybe as a recommendation for similar studies to follow I would like to give some recommendations to possibly improve model fidelity for a similar classification problem in the future, as well as give some possible ways in which further research may be possible for semantic segmentation using deep learning in remote sensing.

To increase the amount of information the model could use from the spectral data, it would be prudent to configure the modules in such a way to allow compatibility with arrays made through GDAL if working with python or R code, which is compatible with data with multitudes of spectral bands. This is a way in which eventually, even hyperspectral data might be incorporated into a deep learning FCNN like the one used in this study, hyperspectral data performs very well in detecting Palm trees and other types of vegetation and could even give information about plant health among other things [4], [7].

Another thing to incorporate to better punish/reward the model for spatial patterns in the data and model predictions would be inclusion of IoU (intersect over union) or otherwise known as Jaccard loss [15], [22], [43]. The model currently works with a per pixel loss function only, making the spectral information far more important than the spatial dimension during training. The F1 score could possibly still have been improved by lowering the threshold for Palm tree detection, but this would have no theoretical backing and could possibly be a solution for this dataset only.

To increase the fidelity and reduce bias in the Ground truth data, I recommend creating a separate model and use object detection algorithms such as Faster R-CNN or U-net to get a less biased and more complete dataset, even with the added risks of obtaining more false positives within the ground truth data. This was the approach used in the “counting the uncountable” study [15]. Bias within the ground truth dataset was one of the major obstacles within this study.

A side benefit from creating your own dataset through object-detection on higher resolution imagery would be an increase in the amount of spectral and ground truth masks available for training as spectral data would no longer be tied around a fixed sampling date. This would allow for the use of a far larger dataset for training should the computational limits imposed by the study permit it.

Finally, if the dataset for a new study in the field can still be expected to be imbalanced, (too small or biased), instead of MSE as a loss function for regression a loss function designed to work better with imbalanced data such as MFE (mean false error) could be tried instead [39].

7 References

[1] Ó. Dembilio and J. A. Jaques, “Biology and management of red palm weevil,” in *Sustainable Pest Management in Date Palm: Current Status and Emerging Challenges*, Springer, 2015, pp. 13–36.

[2] O. Golomb, V. Alchanatis, Y. Cohen, N. Levin, Y. Cohen, and V. Soroker, “Detection of red palm weevil infected trees using thermal imaging,” in *Precision agriculture ’15*, J. V. Stafford, Ed. The Netherlands: Wageningen Academic Publishers, 2015, pp. 643–650.

[3] J. Pinhas, V. Soroker, A. Hetzroni, A. Mizrach, M. Teicher, and J. Goldberger, “Automatic acoustic detection of the red palm weevil,” *computers and electronics in agriculture*, vol. 63, no. 2, pp. 131–139, 2008.

[4] M. S. Yones, M. A. Aboelghar, M. A. El-Shirbeny, G. A. Khdry, A. M. Ali, and N. S. Saleh, “Hyperspectral indices for assessing damage by the red palm weevil Rhynchophorus ferrugineus (coleoptera: curculionidae) in date palms,” *International Journal Geosciences and Geomatics*, vol. 2, pp. 16–23, 2014.

[5] C. Morici, “Phoenix canariensis in the wild,” *Principes*, vol. 42, no. 2, pp. 85–89, 92–93, 1998.

[6] P. Srestasathiern and P. Rakwatin, “Oil Palm Tree Detection with High Resolution Multi-Spectral Satellite Imagery,” *Remote Sensing*, vol. 6, no. 10, Art. no. 10, Oct. 2014, doi: 10.3390/rs6109749.

[7] K. Jusoff and M. Pathan, “Mapping of Individual Oil Palm Trees Using Airborne Hyperspectral Sensing: An Overview,” *APR*, vol. 1, no. 1, Art. no. 1, Apr. 2009, doi: 10.5539/apr.v1n1p15.

[8] M. MacDonell, K. Morgan, and L. Newland, “Integrating information for better environmental decisions,” *Environ Sci & Pollut Res*, vol. 9, no. 6, pp. 359–368, Nov. 2002, doi: 10.1007/BF02987582.

[9] C. Pohl and J. L. V. Genderen, “Review article Multisensor image fusion in remote sensing: Concepts, methods and applications,” *International Journal of Remote Sensing*, vol. 19, no. 5, pp. 823–854, Jan. 1998, doi: 10.1080/014311698215748.

[10] H. Ghassemian, “A review of remote sensing image fusion methods,” *Information Fusion*, vol. 32, pp. 75–89, Nov. 2016, doi: 10.1016/j.inffus.2016.03.003.

[11] R. Petersen *et al.*, “Mapping tree plantations with multispectral imagery: preliminary results for seven tropical countries,” p. 18.

[12] J. Matocha, G. Schroth, T. Hills, and D. Hole, “Integrating Climate Change Adaptation and Mitigation Through Agroforestry and Ecosystem Conservation,” in *Agroforestry - The Future of Global Land Use*, P. K. R. Nair and D. Garrity, Eds. Dordrecht: Springer Netherlands, 2012, pp. 105–126.

[13] A. E. Hadrami and J. M. Al-Khayri, “Socioeconomic and traditional importance of date palm,” p. 16.

[14] D. Martin and A. G. Woodside, “Grounded Theory of International Tourism Behavior,” *Journal of Travel & Tourism Marketing*, vol. 24, no. 4, pp. 245–258, Jul. 2008, doi: 10.1080/10548400802156695.

[15] A. C. Rodriguez and J. D. Wegner, “Counting the Uncountable: Deep Semantic Density Estimation from Space,” in *Pattern Recognition*, Cham, 2019, pp. 351–362, doi: 10.1007/978-3-030-12939-2_24.

[16] V. Khryashchev, L. Ivanovsky, V. Pavlov, A. Ostrovskaya, and A. Rubtsov, “Comparison of Different Convolutional Neural Network Architectures for Satellite Image Segmentation,” in *Proceedings of the 23rd Conference of Open Innovations Association FRUCT*, Bologna, Italy, Nov. 2018, pp. 172–179, Accessed: Jul. 07, 2020. [Online].

[17] R. Girshick, J. Donahue, T. Darrell, and J. Malik, “Rich feature hierarchies for accurate object detection and semantic segmentation,” *arXiv:1311.2524 [cs]*, Oct. 2014, Accessed: Jul. 07, 2020. [Online]. Available: http://arxiv.org/abs/1311.2524.

[18] K. He, G. Gkioxari, P. Dollar, and R. Girshick, “Mask R-CNN,” 2017, pp. 2961–2969, Accessed: Jul. 07, 2020. [Online]. Available:

https://openaccess.thecvf.com/content_iccv_2017/html/He_Mask_R-CNN_ICCV_2017_paper.html.

[19] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional Networks for Biomedical Image Segmentation," in *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, Cham, 2015, pp. 234–241, doi: 10.1007/978-3-319-24574-4_28.

[20] A. Rakhlin, A. Davydow, and S. Nikolenko, "Land Cover Classification from Satellite Imagery with U-Net and Lovász-Softmax Loss," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, Salt Lake City, UT, USA, Jun. 2018, pp. 257–2574, doi: 10.1109/CVPRW.2018.00048.

[21] L.-C. Chen, G. Papandreou, F. Schroff, and H. Adam, "Rethinking Atrous Convolution for Semantic Image Segmentation," *arXiv:1706.05587 [cs]*, Dec. 2017, Accessed: Jun. 25, 2020. [Online]. Available: http://arxiv.org/abs/1706.05587.

[22] A. Garcia-Garcia, S. Orts-Escolano, S. Oprea, V. Villena-Martinez, and J. Garcia-Rodriguez, "A Review on Deep Learning Techniques Applied to Semantic Segmentation," *arXiv:1704.06857 [cs]*, Apr. 2017, Accessed: Jul. 07, 2020. [Online]. Available: http://arxiv.org/abs/1704.06857.

[23] M. Cordts *et al.*, "The cityscapes dataset for semantic urban scene understanding," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 3213–3223.

[24] L.-C. Chen, A. Hermans, G. Papandreou, F. Schroff, P. Wang, and H. Adam, "MaskLab: Instance Segmentation by Refining Object Detection With Semantic and Direction Features," 2018, pp. 4013–4022, Accessed: Jul. 07, 2020. [Online]. Available: https://openaccess.thecvf.com/content_cvpr_2018/html/Chen_MaskLab_Instance_Segmentation_CVPR_2018_paper.html.

[25] I. Demir *et al.*, "DeepGlobe 2018: A Challenge to Parse the Earth through Satellite Images," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, Salt Lake City, UT, Jun. 2018, pp. 172–17209, doi: 10.1109/CVPRW.2018.00031.

[26] G. Kunkel, *Biogeography and Ecology in the Canary Islands*. Springer Science & Business Media, 2012.

[27] H.-U. Schmincke, "The Geology of the Canary Islands," in *Biogeography and Ecology in the Canary Islands*, G. Kunkel, Ed. Dordrecht: Springer Netherlands, 1976, pp. 67–184.

[28] S. D. Jawak and A. J. Luis, "A Comprehensive Evaluation of PAN-Sharpening Algorithms Coupled with Resampling Methods for Image Synthesis of Very High Resolution Remotely Sensed Satellite Data," *Advances in Remote Sensing*, vol. 2013, Dec. 2013, doi: 10.4236/ars.2013.24036.

[29] A. J. Elmore, J. F. Mustard, S. J. Manning, and D. B. Lobell, "Quantifying Vegetation Change in Semiarid Environments: Precision and Accuracy of Spectral Mixture Analysis and the Normalized Difference Vegetation Index," *Remote Sensing of Environment*, vol. 73, no. 1, pp. 87–102, Jul. 2000, doi: 10.1016/S0034-4257(00)00100-0.

[30] L. Taylor and G. Nitschke, "Improving Deep Learning using Generic Data Augmentation," *arXiv:1708.06020 [cs, stat]*, Aug. 2017, Accessed: Jul. 07, 2020. [Online]. Available: http://arxiv.org/abs/1708.06020.

[31] K. Grauman and T. Darrell, "The pyramid match kernel: discriminative classification with sets of image features," in *Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1*, Beijing, China, 2005, pp. 1458-1465 Vol. 2, doi: 10.1109/ICCV.2005.239.

[32] T.-Y. Lin *et al.*, "Microsoft COCO: Common Objects in Context," in *Computer Vision – ECCV 2014*, vol. 8693, D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, Eds. Cham: Springer International Publishing, 2014, pp. 740–755.

[33] D. M. Powers, "Evaluation: from Precision, Recall and F-measure to ROC, Informedness, Markedness and Correlation," Dec. 2011, Accessed: Jul. 07, 2020. [Online]. Available: https://dspace.flinders.edu.au/xmlui/handle/2328/27165.

[34] L. C. Alatorre, R. Sánchez-Andrés, S. Cirujano, S. Beguería, and S. Sánchez-Carrillo, "Identification of Mangrove Areas by Remote Sensing: The ROC Curve Technique Applied to the Northwestern Mexico Coastal Zone Using Landsat Imagery," *Remote Sensing*, vol. 3, no. 8, Art. no. 8, Aug. 2011, doi: 10.3390/rs3081568.

[35] K. Woods and K. W. Bowyer, "Generating ROC curves for artificial neural networks," *IEEE Transactions on Medical Imaging*, vol. 16, no. 3, pp. 329–337, Jun. 1997, doi: 10.1109/42.585767.

[36] J. A. Hanley and B. J. McNeil, "The meaning and use of the area under a receiver operating characteristic (ROC) curve.," *Radiology*, vol. 143, no. 1, pp. 29–36, 1982.

[37] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[38] J. W. Rouse Jr, R. H. Haas, J. A. Schell, and D. W. Deering, "Paper A 20," in *Third Earth Resources Technology Satellite-1 Symposium: The Proceedings of a Symposium Held by Goddard Space Flight Center at Washington, DC on December 10-14, 1973: Prepared at Goddard Space Flight Center*, 1974, vol. 351, p. 309.

[39] S. Wang, W. Liu, J. Wu, L. Cao, Q. Meng, and P. J. Kennedy, "Training deep neural networks on imbalanced data sets," in *2016 International Joint Conference on Neural Networks (IJCNN)*, Jul. 2016, pp. 4368–4374, doi: 10.1109/IJCNN.2016.7727770.

[40] H. He and X. Shen, "A Ranked Subspace Learning Method for Gene Expression Data Classification.," in *IC-AI*, 2007, pp. 358–364.

[41] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," in *Advances in Neural Information Processing Systems 28*, C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, Eds. Curran Associates, Inc., 2015, pp. 91–99.

[42] M. A. Mazurowski, P. A. Habas, J. M. Zurada, J. Y. Lo, J. A. Baker, and G. D. Tourassi, "Training neural network classifiers for medical decision making: The effects of imbalanced datasets on classification performance," *Neural Networks*, vol. 21, no. 2, pp. 427–436, Mar. 2008, doi: 10.1016/j.neunet.2007.12.031.

[43] E. Maggiori, Y. Tarabalka, G. Charpiat, and P. Alliez, "Convolutional Neural Networks for Large-Scale Remote-Sensing Image Classification," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 55, no. 2, pp. 645–657, Feb. 2017, doi: 10.1109/TGRS.2016.2612821.

Appendix A: Table of contents of accompanying zip

Below a content scheme for the accompanying zip of the report will be shown. The directories and subdirectories will be shown using bullet levels, with a higher level indicating the shown directory is a sub-directory/file of the directory one bullet level lower. Directories will be indicated as bold and files as normal text. Explanations will be given in cursive.

- **Master thesis Laurens Buddingh:**
  - **Report**
    - Final report.pdf
  - **Presentations**
    - Midterm presentation.pdf
    - Final presentation.pdf
  - **Data**
    - **Groundtruth**
      - Point dataset.shp (*as acquired from* https://opendata.sitcan.es)
      - Density raster.tif *(Kernel density product)*
    - **Spectral data**
      - 3 banded mosaic.tif *(The mosaic used for this study)*
    - **Trained model**
      - Log.csv *(metrics for each epoch)*
      - Trained model *(Best scoring model)*
    - **Predictions**
      - Merged predictions.tif *(Merged predictions raster obtained using the trained model)*
  - **Figures**
    - **Images**
      - Title image
      - Image 1-21 *(separate files for all 21 images used in this report)*
    - **Tables**
      - All tables within this paper
  - **Scripts**
    - Sentinel-2 Data scraper (*Java script for earth engine*)
    - Create mosaic (*Python notebook, used in Google Colab environment)*
    - Data clipper (*Python notebook, used in Google Colab environment*)
    - DeeplabV3 Semantic segmentation (*Python notebook, used in Google Colab environment*)
    - Modelevals (*Python notebook, used in Google Colab environment*)
    - Datavisualization (*Python notebook, used in Google Colab environment*)
  - **Literature**
    - Reference library (*Zotero rdf containing all the references and pdfs of these papers, should be endnote compatible*)
  - **Other**
    - **Additional sources**
      - GitHub sources.txt *(Text file containing links to used GitHub libraries)*

Appendix B: Uncertain choices

Especially in the creating of the scripts for the FCNN model and the pre-processing, documentation on best practices where not always available.

The dataset I used as a ground truth data in hindsight might have been a poor choice. To avoid some of the bias at the risk of having a ground truth that is less accurate for the palm tree locations might have worked out better.

The first choice was made during sampling of the spectral data. The Canary Islands are a large sampling area and requires multiple Sentinel-2 images to fully cover. The sampling of the palm trees was done during 2017-2018 so in accordance with the research by Rodriguez et al. [15] the data was collected close to this sampling campaign (see appendix B sentinel-2 data scraper script). Median raster bands (based on their data of acquisition) were taken but it might still be possible that reflectance data on some parts of the sampling area are taken at different seasons than others, which would definitely affect the reflectance data.

The ground truth was rasterized using a quartic kernel, at distance zero, the weight of each palm tree is 15/16, which does not equal 0. This means the density over the whole map is 15/16 of what it should be. Since this has no impact on training or validation it is not important for this study, but if you want to have the most realistic density result it is better to use a triangular kernel instead since that has a weight of 1 at distance 0. The ground truth also did not immediately exactly match the spectral data pixel by pixel, and a warp using the NN algorithm was required. While for most pixels in denser areas this would not have been a large problem, this could have affected the accuracy of the ground truth in areas with lower ground truth densities through sometimes assigning a palm tree to the wrong neighboring pixel.

Implementation of the Deeplabv3 model had not been covered in any of my previous study material or in the paper by Rodriguez et al, so I mainly used other GitHub repos of deeplabv3 implementations and the GitHub page of Deeplabv3 itself as a guideline for implementation. While I am certain that with more time, I could have gotten an infrastructure ready that supports high bit depth multi layered .TIF files as a data source, doing so would require rewriting many of the libraries and modules Deeplabv3 standardly uses. Should unexpected artifacts or errors occur during such a progress, it would take far more time than I had available to troubleshoot these problems.

The choice for model optimization fell on ADAM based on its performance over other model optimizers, but its automated (and somewhat untransparent) nature makes it hard to tell what kind of influence it had on model improvements, even if it was likely better than other stochastic optimizers I could have chosen.

Appendix C: Scripts

The scripts shown in overview figure 10 will be listed in order here. There are also two extra scripts used for the creation of some of the figures in the results included as well. Aside from the first script which was made to run in Google Earth Engine, the other scripts are Jupyter Notebooks made through Google Colab.

These Jupyter Notebooks should still work on other python parsers if the correct dependencies are installed and the file paths are changed accordingly, in their current state they still contain several hardcoded model paths.

Some parts of the scripts might need rerunning several times to get all the required data. This will be mentioned through comments where applicable.

Script: Sentinel-2 data scraper:

Made for use in Google earth engine.

----start script

```
//Script for obtaining necessary Sentinel-2 data for the masterthesis
//Link to Sentinel-2 repository and dates
var sent2images = ee.ImageCollection("COPERNICUS/S2_SR");
var startdate = '2017-01-01';
var enddate = '2018-12-31';
//Link to canary islands geometry
var geometry = /* color: #d63000 */ee.Geometry.Polygon([ [ [ -
17.983562111252255, 27.640364565653599 ], [ -
18.103908886189071, 27.748368200036687 ], [ -
18.108295667807063, 27.753839972168592 ], [ -
18.001964609729917, 28.78439863645454 ], [ -
17.955776714967996, 28.825027802955692 ], [ -
17.945734346543624, 28.831876473093224 ], [ -
17.928676580192011, 28.841150884711901 ], [ -
17.907759614989637, 28.849241828627328 ], [ -
13.481834357996847, 29.26231878836197 ], [ -
13.480018386452018, 29.262143998115818 ], [ -
13.479003681730651, 29.261728861440272 ], [ -
```

```
13.451080547206391, 29.221574282347117 ], [ -
13.427207952149834, 29.17386375234231 ], [ -
13.449987152362565, 29.085114928593324 ], [ -
13.450314308595859, 29.083960408476266 ], [ -
13.481742053004165, 29.002494615214747 ], [ -
13.486577867935562, 28.992716606342654 ], [ -
13.895762552847867, 28.323124385362561 ], [ -
13.896345476678382, 28.322185316479612 ], [ -
13.983551964167813, 28.228084347331908 ], [ -
14.014103650289043, 28.211389324127005 ], [ -
14.329013305400638, 28.049773041707393 ], [ -
15.598315551824479, 27.735120044725196 ], [ -
17.983562111252255, 27.640364565653599 ] ] ]);
//filter data
var sent2subset = sent2images
  .filterBounds(geometry)
  .filterDate(startdate,enddate)
  .filterMetadata("CLOUD_COVERAGE_ASSESSMENT","less_than",
10);

// Select median stitched data (cloudfree) per pixel in the region
var bandselect =
sent2subset.select("B1","B2","B3","B4","B5","B6","B7","B8","B8A","B
9","B11","B12").median();

//visualize
Map.addLayer(bandselect);

Export.image.toDrive({
  image: bandselect,
  region: geometry,
```

```
  description: 'canary islands sentinel-2 2018',
  scale: 10,
  fileFormat: 'GeoTIFF',
  formatOptions: {cloudOptimized: true},
  maxPixels: 3500000000000
});
```

---end script

Script: Create mosaic

.ipynb used in google colab environment

----start script

```
#install missing dependencies
!pip install rasterio
!pip install glob
!pip install os



#import modules
import rasterio
from rasterio.merge import merge
import glob
import os
import gc
from google.colab import drive
drive.mount('/content/drive')


# File and folder paths
In [6]: dirpath = "/content/drive/My Drive/MASTERTHESE/sentinel 2"

In [7]: out_fp = "/content/drive/My Drive/MASTERTHESE/mosaic"

# Make a search criteria to select the DEM files
In [8]: search_criteria = "canary*.tif"

In [9]: q = os.path.join(dirpath, search_criteria)

In [10]: print(q)


#check if mounting works
!ls "/content/drive/My Drive/MASTERTHESE/sentinel 2"
```

```
#create a list of the sentinel-2 images based on the chosen wildcard
sent2_fps = glob.glob(q)
sent2_fps


#create an empty list for later
src_files_to_mosaic = []


#open the sentinel 2 images and add them to the empty list
for fp in sent2_fps:
  src = rasterio.open(fp)
  src_files_to_mosaic.append(src)


# Merge function returns a single mosaic array and the transformation in
fo, can only do single banded mosaics, change index from 1 to 12 manuall
y and rerun
mosaic, out_trans = merge(src_files_to_mosaic, indexes = [1])


#update the metadata to work with single banded images for now due to me
mory restrictions
out_meta = src.meta.copy()

out_meta.update({
 'count': 1,
 'height': mosaic.shape[1],
 'width': mosaic.shape[2],
 'transform': out_trans})

out_meta


#write the mosaiced raster to a file
out_fp = "/content/drive/My Drive/MASTERTHESE/mosaic/mosaic001.tif"
with rasterio.open(out_fp, "w", **out_meta) as dest:
    dest.write(mosaic)

#get the path to the mosaic directory and put them in a list with a wild
card
mosaicpath = "/content/drive/My Drive/MASTERTHESE/mosaic/"
mosaiccriteria = "mosaic*"
p = os.path.join(mosaicpath, mosaiccriteria)

#Create the wildcard list
mosaic_fps = sorted(glob.glob(p))
mosaic3band = [mosaic_fps[1],mosaic_fps[3],mosaic_fps[7]]
```

```
# Read metadata of first file
with rasterio.open(mosaic3band[0]) as src0:
    meta = src0.meta

# Update meta to reflect the number of layers
meta.update(count = len(mosaic3band))

# Read each layer and write it to stack geotiff file
CHECK_DISK_FREE_SPACE = False
stackedmosaic_fp = "/content/drive/My Drive/MASTERTHESE/mosaic/palmtreem
osaic3bd.tif"
with rasterio.Env(CHECK_DISK_FREE_SPACE=False):
  with rasterio.open(stackedmosaic_fp, 'w', **meta) as dst:
    for id, layer in enumerate(mosaic3band, start=1):
        with rasterio.open(layer) as src1:
            dst.write_band(id, src1.read(1))
```

---Script end

Script: Data clipper

.ipynb used in google colab environment

Kd image required data made in QGIS using the kernel density tool

---Script start

```
install required libraries
!pip install rasterio
!pip install pycrs
!pip install geopandas

import required functions/tools
import rasterio
from rasterio.mask import mask
import geopandas as gpd
import shapely
from fiona.crs import from_epsg
import pycrs
import glob
import os
from itertools import product
from rasterio import windows
from osgeo import gdal, gdalconst
import numpy as np
from PIL import Image
from scipy import ndimage
```

```
#match ground truth raster and input data raster extents and resolution
through resampling

# add input sources
src_filename = '/content/drive/My Drive/MASTERTHESE/fulldatarun/groundtr
uth/gtboolean.tif'
src = gdal.Open(src_filename, gdalconst.GA_ReadOnly)
src_proj = src.GetProjection()
src_geotrans = src.GetGeoTransform()

# We want a section of source that matches this:
match_filename = '/content/drive/My Drive/MASTERTHESE/mosaic/qgis_reproj
bd3.tif'
match_ds = gdal.Open(match_filename, gdalconst.GA_ReadOnly)
match_proj = match_ds.GetProjection()
match_geotrans = match_ds.GetGeoTransform()
wide = match_ds.RasterXSize
high = match_ds.RasterYSize

# establish correct output files, destinations
dst_filename = '/content/drive/My Drive/MASTERTHESE/fulldatarun/groundtr
uth/gtboolean_reproj.tif'
dst = gdal.GetDriverByName('GTiff').Create(dst_filename, wide, high, 1,
gdalconst.GDT_Int16)
dst.SetGeoTransform( match_geotrans )
dst.SetProjection( match_proj)

# Reproject
gdal.ReprojectImage(src, dst, src_proj, match_proj, gdalconst.GRA_Neares
tNeighbour)

del dst # Flush to clear ram


#Reclassify ground truth data to change all values other than 0 or 1 to
0 (nodata is -3.26E38 for this dataset)
file = '/content/drive/My Drive/MASTERTHESE/fulldatarun/groundtruth/gtde
nsitymap.tif'
ds = gdal.Open(file)
band = ds.GetRasterBand(1)
arr = band.ReadAsArray()
[cols, rows] = arr.shape
arr_out = np.where((arr >= 0 ), arr, 0)
driver = gdal.GetDriverByName("GTiff")
outdata = driver.Create('/content/drive/My Drive/MASTERTHESE/fulldatarun
/groundtruth/gtdensitymapcorrect.tif', rows, cols, 1, gdal.GDT_UInt16)
outdata.SetGeoTransform(ds.GetGeoTransform())##sets same geotransform as
 input
```

```
outdata.SetProjection(ds.GetProjection())##sets same projection as input
outdata.GetRasterBand(1).WriteArray(arr_out)
outdata.FlushCache() ##saves to disk!!
outdata = None
band=None
ds=None

#STEP 3: import required data
#raster to clip, shapefile to clip with and output clipped raster
raster = "/content/drive/My Drive/MASTERTHESE/fulldatarun/groundtruth/de
nsitycorrect_reproj.tif"
gtraster = "/content/drive/My Drive/MASTERTHESE/fulldatarun/groundtruth/
gtboolean_reproj.tif"
gtdata = rasterio.open(gtraster)
data = rasterio.open(raster)
shp = "/content/drive/My Drive/MASTERTHESE/fulldatarun/shapefile/alldata
shp.shp"
output = "/content/drive/My Drive/MASTERTHESE/fulldatarun/clips/full3bdc
lip.tif"

#Convert the clip shape file to a geopanda
geo = gpd.read_file(shp)

def getFeatures(gdf):
    """Function to parse features from GeoDataFrame in such a manner tha
t rasterio wants them"""
    import json
    return [json.loads(gdf.to_json())['features'][0]['geometry']]

#Get the geometry of the geopanda
coords = getFeatures(geo)
# Clip the raster with Polygon
out_img, out_transform = mask(dataset=data, shapes=coords, crop=True)


# Copy the metadata
out_meta = data.meta.copy()

# Parse EPSG code
epsg_code = int(data.crs.data['init'][5:])
print(epsg_code)

#Update metadata and write clipped raster to file
out_meta.update({"driver": "GTiff",
                 "height": out_img.shape[1],
                 "width": out_img.shape[2],
                 "transform": out_transform,
```

```
                "crs": pycrs.parse.from_epsg_code(epsg_code).to_proj4()
}
                          )
with rasterio.open(output, "w", **out_meta) as dest:
        dest.write(out_img)

convert raster datasets into clips for usage as trainingset
###Reminder###, for followup steps, make sure groundtruth data is in the
 format (0= not instance of interest, 1 to n= instances of interest)
in_path_gt = '/content/drive/My Drive/MASTERTHESE/fulldatarun/groundtrut
h'
input_filename_gt = 'gtboolean_reproj.tif'

out_path_gt = '/content/drive/My Drive/MASTERTHESE/fulldatarun/gttiles'
output_filename_gt = 'gt_tile_{}-{}.tif'

in_path_3b = '/content/drive/My Drive/MASTERTHESE/mosaic'
input_filename_3b = 'qgis_reprojbd3.tif'

out_path_3b = '/content/drive/My Drive/MASTERTHESE/fulldatarun/mbtiles'
output_filename_3b = 'b3_tile_{}-{}.tif'

in_path_kd = '/content/drive/My Drive/MASTERTHESE/fulldatarun/groundtrut
h'
input_filename_kd = 'densitycorrect_reproj.tif'

out_path_kd = '/content/drive/My Drive/MASTERTHESE/fulldatarun/kdtiles'
output_filename_kd = 'kd_tile_{}-{}.tif'

def get_tiles(ds, width=256, height=256):
  """Function used to cut the dataset streams into clips for use in trai
ning the fcnn"""
    nols, nrows = ds.meta['width'], ds.meta['height']
    offsets = product(range(0, nols, width), range(0, nrows, height))
    big_window = windows.Window(col_off=0, row_off=0, width=nols, height
=nrows)
    for col_off, row_off in  offsets:
        window =windows.Window(col_off=col_off, row_off=row_off, width=w
idth, height=height).intersection(big_window)
        transform = windows.transform(window, ds.transform)
        yield window, transform

#use the get tiles function to write the current datasets into clips of
the desired height and width
with rasterio.open(os.path.join(in_path_gt, input_filename_gt)) as inds:
    tile_width, tile_height = 256, 256

    meta = inds.meta.copy()
```

```
    for window, transform in get_tiles(inds):
        meta['transform'] = transform
        meta['width'], meta['height'] = window.width, window.height
        outpath = os.path.join(out_path_kd,output_filename_kd.format(int
(window.col_off), int(window.row_off)))
        with rasterio.open(outpath, 'w', **meta) as outds:
            outds.write(inds.read(window=window))

#Use the following function to remove superfluous data
def groundtruthimages3(gtdirectory='/content/drive/My Drive/MASTERTHESE/
fulldatarun/gttiles/*.tif'):
  """Function that searches through images for objects of interest, crea
ting a list containing only those images that
  specifications."""
  imgs = os.listdir(gtdirectory)
  containsgt = []
  for i in tqdm(imgs):
    inds = Image.open(os.path.join(gtdirectory,i))
    arr = np.array(inds)
    sumgt = np.count_nonzero(arr > 0)
    if (sumgt > 1):
      containsgt.append(os.path.join(gtdirectory,i))
  return containsgt


#Create matching lists of groundtruthimages and png images in the direct
ories
groundtruthimagesnew = groundtruthimages3("/content/drive/My Drive/MASTE
RTHESE/fulldatarun/kdtiles")
#Replace directory path of ground truth files with directory name of raw
 images
#Use replace function to get matching spectral tiles
pngimages = [x.replace('kdtiles/kd', 'mbtiles/b3') for x in groundtruthi
magesnew]

#Check if the same amount of windows for groundtruth vs input were made
print(len(os.listdir('/content/drive/My Drive/MASTERTHESE/fulldatarun/mb
tiles')), 'vs',
len(os.listdir('/content/drive/My Drive/MASTERTHESE/fulldatarun/kdtiles'
)))


#Write the remaining clipped files to png images in a new directory, onc
e for the ground truth and once for the spectral data

options_list = [
    '-ot Byte',
```

```
    '-of PNG',
    '-scale'
]
options_string = " ".join(options_list)

outputdir = '/content/drive/My Drive/MASTERTHESE/fulldatarun/multipng_3b
'
for tif in tqdm(pngimages):
      base = os.path.splitext(tif)[0]
      tifname = base.split("/")[-1]
      pngname = tifname + ".png"
      output_path = os.path.join(outputdir,pngname)
      gdal.Translate(output_path, tif, options=options_string)
#Repeat for ground truth if necessary
```

---script end

Script: Deeplabv3 semantic segmentation

.ipynb used in google colab environment

---Script start

```
%%shell
#install prerequisites
pip install rasterio
pip install cython
# Install pycocotools, the version by default in Colab
# has a bug fixed in https://github.com/cocodataset/cocoapi/pull/354
pip install -
U 'git+https://github.com/cocodataset/cocoapi.git#subdirectory=PythonAPI
'
#Import required modules
from torch.utils.data import Dataset, DataLoader
import glob
import os
import numpy as np
import cv2
import torch
from torchvision import transforms, utils
from PIL import Image
import numpy as np
from scipy import ndimage



#obtain matching lists of groundtruthimages and png images in the direct
ories created in dataclipper script
```

```
groundtruthimages = sorted(glob.glob('/content/drive/My Drive/MASTERTHES
E/fulldatarun/multipng_kd/*.png'))
#Replace directory path of ground truth files with directory name of raw
 images and assert both directories contain same amount of images
pngimages = sorted(glob.glob('/content/drive/My Drive/MASTERTHESE/fullda
tarun/multipng_3b/*.png'))

assert len(groundtruthimages) == len(pngimages)


#Code for loading the data and all the required classes
class SegDataset(Dataset):
    """Segmentation Dataset"""

    def __init__(self, root_dir, images, masks, transform=None, seed=Non
e, fraction=None, subset=None, imagecolormode='rgb', maskcolormode='gray
scale'):
        """
        Args:
            root_dir (string): Directory with all the images and should
have the following structure.
            root
            --Images
            -----Img 1
            -----Img N
            --Mask
            -----Mask 1
            -----Mask N
            images (string) = 'Images' : Sorted list of training images.
            masks (string)  = 'Masks : Sorted list of masks.
            transform (callable, optional): Optional transform to be app
lied on a sample.
            seed: Specify a seed for the train and test split
            fraction: A float value from 0 to 1 which specifies the vali
dation split fraction
            subset: 'Train' or 'Test' to select the appropriate set.
            imagecolormode: 'rgb' or 'grayscale'
            maskcolormode: 'rgb' or 'grayscale'
        """
        self.color_dict = {'rgb': 1, 'grayscale': 0}
        assert(imagecolormode in ['rgb', 'grayscale'])
        assert(maskcolormode in ['rgb', 'grayscale'])

        self.imagecolorflag = self.color_dict[imagecolormode]
        self.maskcolorflag = self.color_dict[maskcolormode]
        self.root_dir = root_dir
        self.transform = transform
        if not fraction:
```

```
            self.image_names = sorted(images)
            self.mask_names = sorted(masks)
        else:
            assert(subset in ['Train', 'Test'])
            self.fraction = fraction
            self.image_list = np.array(
                sorted(images))
            self.mask_list = np.array(
                sorted(masks))
            if seed:
                np.random.seed(seed)
                indices = np.arange(len(self.image_list))
                np.random.shuffle(indices)
                self.image_list = self.image_list[indices]
                self.mask_list = self.mask_list[indices]
            if subset == 'Train':
                self.image_names = self.image_list[:int(
                    np.ceil(len(self.image_list)*(1-self.fraction)))]
                self.mask_names = self.mask_list[:int(
                    np.ceil(len(self.mask_list)*(1-self.fraction)))]
            else:
                self.image_names = self.image_list[int(
                    np.ceil(len(self.image_list)*(1-self.fraction))):]
                self.mask_names = self.mask_list[int(
                    np.ceil(len(self.mask_list)*(1-self.fraction))):]

    def __len__(self):
        return len(self.image_names)

    def __getitem__(self, idx):
        img_name = self.image_names[idx]
        if self.imagecolorflag:
            image = cv2.imread(
                img_name, self.imagecolorflag).transpose(2, 0, 1)
        else:
            image = cv2.imread(img_name, self.imagecolorflag)
        msk_name = self.mask_names[idx]
        if self.maskcolorflag:
            mask = cv2.imread(msk_name, self.maskcolorflag).transpose(2,
0, 1)
        else:
            mask = cv2.imread(msk_name, self.maskcolorflag)
        sample = {'image': image, 'mask': mask}

        if self.transform:
            sample = self.transform(sample)

        return sample
```

```
# Define few transformations for the Segmentation Dataloader


class Resize(object):
    """Resize image and/or masks."""

    def __init__(self, imageresize, maskresize):
        self.imageresize = imageresize
        self.maskresize = maskresize

    def __call__(self, sample):
        image, mask = sample['image'], sample['mask']
        if len(image.shape) == 3:
            image = image.transpose(1, 2, 0)
        if len(mask.shape) == 3:
            mask = mask.transpose(1, 2, 0)
        mask = cv2.resize(mask, self.maskresize, cv2.INTER_AREA)
        image = cv2.resize(image, self.imageresize, cv2.INTER_AREA)
        if len(image.shape) == 3:
            image = image.transpose(2, 0, 1)
        if len(mask.shape) == 3:
            masknew = mask.transpose(2, 0, 1)

        return {'image': image,
                'mask': mask}


class ToTensor(object):
    """Convert ndarrays in sample to Tensors."""

    def __call__(self, sample, maskresize=None, imageresize=None):
        image, mask = sample['image'], sample['mask']
        if len(mask.shape) == 2:
            mask = mask.reshape((1,)+mask.shape)
        if len(image.shape) == 2:
            image = image.reshape((1,)+image.shape)
        return {'image': torch.from_numpy(image),
                'mask': torch.from_numpy(mask)}


class Normalize(object):
    '''Normalize image'''

    def __call__(self, sample):
        image, mask = sample['image'], sample['mask']
        return {'image': image.type(torch.FloatTensor)/255,
                'mask': mask.type(torch.FloatTensor)/255}
```

```
class RandomHorizontalFlip(object):
  """randomly flip an image and its ground truth"""
  def __call__(self, sample, maskresize=None, imageresize=None):
    image, mask = sample['image'], sample['mask']
    if len(image.shape) == 3:
            image = image.transpose(1, 2, 0)
    if len(mask.shape) == 3:
            mask = mask.transpose(1, 2, 0)
    if torch.rand(1) < 0.5:
            image = cv2.flip(image, 1)
            mask = cv2.flip(mask, 1)
    if len(image.shape) == 3:
            image = image.transpose(2, 0, 1)
    if len(mask.shape) == 3:
            masknew = mask.transpose(2, 0, 1)
    return {'image': image,
                'mask': mask}

class RandomVerticalFlip(object):
  """randomly flip an image and its ground truth"""
  def __call__(self, sample, maskresize=None, imageresize=None):
    image, mask = sample['image'], sample['mask']
    if len(image.shape) == 3:
            image = image.transpose(1, 2, 0)
    if len(mask.shape) == 3:
            mask = mask.transpose(1, 2, 0)
    if torch.rand(1) < 0.5:
            image = cv2.flip(image, 0)
            mask = cv2.flip(mask, 0)
    if len(image.shape) == 3:
            image = image.transpose(2, 0, 1)
    if len(mask.shape) == 3:
            masknew = mask.transpose(2, 0, 1)
    return {'image': image,
                'mask': mask}




def get_dataloader_sep_folder(data_dir, imageFolder='Image', maskFolder=
'Mask', batch_size=4):
    """
        Create Train and Test dataloaders from two separate Train and Te
st folders.
        The directory structure should be as follows.
        data_dir
        --Train
        ------Image
```

```
        ---------Image1
        ---------ImageN
        ------Mask
        ---------Mask1
        ---------MaskN
        --Train
        ------Image
        ---------Image1
        ---------ImageN
        ------Mask
        ---------Mask1
        ---------MaskN
    """
    data_transforms = {
        'Train': transforms.Compose([transforms.ToTensor(), transforms.N
ormalize(), transforms.RandomHorizontalflip(), transforms.RandomVertical
flip]),
        'Test': transforms.Compose([transforms.ToTensor(), transforms.No
rmalize()]),
    }

    image_datasets = {x: SegDataset(root_dir=os.path.join(data_dir, x),
                                     transform=data_transforms[x], maskFo
lder=maskFolder, imageFolder=imageFolder)
                      for x in ['Train', 'Test']}
    dataloaders = {x: DataLoader(image_datasets[x], batch_size=batch_siz
e,
                                 shuffle=True, num_workers=8)
                   for x in ['Train', 'Test']}
    return dataloaders


def get_dataloader_single_folder(data_dir, images, masks, fraction=0.2,
batch_size=4):
    """
        Create training and testing dataloaders from a single folder,
this version is primarily used in the study
    """
    data_transforms = {
        'Train': transforms.Compose([RandomHorizontalFlip(), RandomVerti
calFlip(), ToTensor(), Normalize()]),
        'Test': transforms.Compose([ToTensor(), Normalize()]),
    }

    image_datasets = {x: SegDataset(data_dir, images, masks, seed=100, f
raction=fraction, subset=x, transform=data_transforms[x])
                      for x in ['Train', 'Test']}
```

```
    dataloaders = {x: DataLoader(image_datasets[x], batch_size=batch_siz
e,
                                 shuffle=True, num_workers=8)
                   for x in ['Train', 'Test']}
    return dataloaders


%%shell
#download pytorch vision repo
git clone https://github.com/pytorch/vision.git
cd vision
git checkout v0.3.0



""" DeepLabv3 Model download and change the head for your prediction"""
import torchvision
from torchvision import models
from torchvision.models.segmentation.deeplabv3 import DeepLabHead

def createDeepLabv3(outputchannels=1):
    model = models.segmentation.deeplabv3_resnet50(
        pretrained=False, progress=True, num_classes=1)
    # Added a Sigmoid activation after the last convolution layer
    model.classifier = DeepLabHead(2048,1)
    model.classifier = torch.nn.Sequential(*list(model.classifier) + [to
rch.nn.Sigmoid()])
    # Set the model in training mode
    model.train()
    return model


def train_model(model, dataloaders, optimizer, metrics, bpath, num_epoch
s=3):
  """Train the model using the previously build data loader and metrics
  ARGS:
  model = the model infrastructure to be trained
  dataloaders = the dataloader used to supply data to the training model
  optimizer = the choice of optimizer used for training hyperparameters
  metrics = the statistical metrics calculated for each epoch
  bpath = destination path of the output model and log
  num_epochs = the number of epochs the model iterates over"""
    since = time.time()
    best_model_wts = copy.deepcopy(model.state_dict())
    best_loss = 1e10
    # Use gpu if available
    device = torch.device("cuda:0" if torch.cuda.is_available() else "cp
u")
```

```
    model.to(device)
    # Initialize the log file for training and testing loss and metrics
    fieldnames = ['epoch', 'Train_loss', 'Test_loss'] + \
        [f'Train_{m}' for m in metrics.keys()] + \
        [f'Test_{m}' for m in metrics.keys()]
    with open(os.path.join(bpath, 'log.csv'), 'w', newline='') as csvfil
e:
        writer = csv.DictWriter(csvfile, fieldnames=fieldnames)
        writer.writeheader()

    for epoch in range(1, num_epochs+1):
        print('Epoch {}/{}'.format(epoch, num_epochs))
        print('-' * 10)
        # Each epoch has a training and validation phase
        # Initialize batch summary
        batchsummary = {a: [0] for a in fieldnames}

        for phase in ['Train', 'Test']:
            if phase == 'Train':
                model.train()  # Set model to training mode
            else:
                model.eval()   # Set model to evaluate mode

            # Iterate over data.
            for sample in tqdm(iter(dataloaders[phase])):
                inputs = sample['image'].to(device)
                masks = sample['mask'].to(device)
                # zero the parameter gradients
                optimizer.zero_grad()

                # track history if only in train
                with torch.set_grad_enabled(phase == 'Train'):
                    outputs = model(inputs)
                    loss = criterion(outputs['out'],masks)
                    y_pred = outputs['out'].data.cpu().numpy().ravel()
                    y_true = masks.data.cpu().numpy().ravel()
                    for name, metric in metrics.items():
                        if name == 'f1_score':
                            # Use a classification threshold of 0.1
                            batchsummary[f'{phase}_{name}'].append(
                                metric(y_true > 0, y_pred > 0.05))
                        else:
                            batchsummary[f'{phase}_{name}'].append(
                                metric(y_true.astype('uint8'), y_pred))

                    # backward + optimize only if in training phase
                    if phase == 'Train':
                        loss.backward()
```

```
                        optimizer.step()
                batchsummary['epoch'] = epoch
                epoch_loss = loss
                batchsummary[f'{phase}_loss'] = epoch_loss.item()
                print('{} Loss: {:.4f}'.format(
                    phase, loss))
            for field in fieldnames[3:]:
                batchsummary[field] = np.mean(batchsummary[field])
            print(batchsummary)
            with open(os.path.join(bpath, 'log.csv'), 'a', newline='') as cs
vfile:
                writer = csv.DictWriter(csvfile, fieldnames=fieldnames)
                writer.writerow(batchsummary)
                # deep copy the model
                if phase == 'Test' and loss < best_loss:
                    best_loss = loss
                    best_model_wts = copy.deepcopy(model.state_dict())

        time_elapsed = time.time() - since
        print('Training complete in {:.0f}m {:.0f}s'.format(
            time_elapsed // 60, time_elapsed % 60))
        print('Lowest Loss: {:4f}'.format(best_loss))

        # load best model weights
        model.load_state_dict(best_model_wts)
        return model

# Create the deeplabv3 resnet50 model which is pretrained on a subset of
 COCO train2017, on the 20 categories that are present in the Pascal VOC
 dataset.
model = createDeepLabv3()
#added a softmax layer at the end of the classifier
model.train()
# Create the experiment directory if not present
if not os.path.isdir("/content/drive/My Drive/MASTERTHESE/fulldatarun/mo
delruns"):
    os.mkdir("/content/drive/My Drive/MASTERTHESE/fulldatarun/modelruns"
)


# Specify the loss function
criterion = torch.nn.MSELoss(reduction='mean')
# Specify the optimizer with a lower learning rate
optimizer = torch.optim.Adam(model.parameters(), lr=1e-3)

# Specify the evalutation metrics
metrics = {'f1_score': f1_score, 'auroc': roc_auc_score, "r2": r2_score,
 'RMSE': mean_squared_error}
```

```
# Create the dataloader
torch.manual_seed(10)
dataloaders = get_dataloader_single_folder(
    '/content/drive/My Drive/MASTERTHESE/fulldatarun', pngimages,groundt
ruthimages , fraction= 0.2, batch_size=2)
trained_model = train_model(model, dataloaders,
                            optimizer=optimizer, bpath="/content/drive/M
y Drive/MASTERTHESE/fulldatarun/modelruns/reproj_sigmoid_regres_r50_more
metrics_randomflips", metrics=metrics, num_epochs=25)


# Save the trained model
# torch.save({'model_state_dict':trained_model.state_dict()},os.path.joi
n(bpath,'weights'))
torch.save(model, os.path.join("/content/drive/My Drive/MASTERTHESE/full
datarun/modelruns/reproj_sigmoid_regres_r50_moremetrics_randomflips", 'w
eights.pt'))
```

---Script end

Evaluation scripts:
Two scripts were made to provide some of the imagery used in the results section of this paper.

Script: Modelevals:
Was used to make figures 20 and helped create the dataset that was necessary for figure 21 that has been visualized through QGIS in figure 15.

.ipynb used in google colab environment
---Script start

```
%%shell
#install prerequisites
pip install rasterio
pip install cython
# Install pycocotools, the version by default in Colab
# has a bug fixed in https://github.com/cocodataset/cocoapi/pull/354
pip install -
U 'git+https://github.com/cocodataset/cocoapi.git#subdirectory=PythonAPI
'
#download pytorch vision repo
git clone https://github.com/pytorch/vision.git
cd vision
git checkout v0.3.0

#import required modules
```

```
from torch.utils.data import Dataset, DataLoader
import glob
import os
import numpy as np
import cv2
import torch
from torchvision import transforms, utils
from PIL import Image
import numpy as np
from scipy import ndimage
import csv
import copy
import time
from tqdm import tqdm


# Load the trained model
device = torch.device("cuda:0" if torch.cuda.is_available() else "cpu")
model = torch.load('/content/drive/My Drive/MASTERTHESE/fulldatarun/mode
lruns/reproj_regres_r50_moremetrics_randomflips/weights.pt')
# Set the model to evaluate mode
model.eval()

# Read the log file using pandas into a dataframe
df = pd.read_csv('/content/drive/My Drive/MASTERTHESE/fulldatarun/modelr
uns/reproj_regres_r50_moremetrics_randomflips/log.csv')
y = df[df.columns.difference(['Train_r2', 'Test_r2'])]
aurochs = df[["Test_auroc", "Train_auroc", "epoch"]]
f1 = df[["Test_f1_score", "Train_f1_score", "epoch"]]
loss = df[["Test_loss", "Train_loss", "epoch"]]
r2 = df[["Test_r2", "Train_r2", "epoch"]]

#Create list of sampling areas
kddir = '/content/drive/My Drive/MASTERTHESE/fulldatarun/multipng_kd'
b3dir = '/content/drive/My Drive/MASTERTHESE/fulldatarun/multipng_3b'
def getsampleimages(dir, imagelist):
  samples = []
  for i in imagelist:
    sample = os.path.join(dir, i)
    samples.append(sample)
  return samples

evalimagelistb3 = ['b3_tile_16896-10496.png','b3_tile_16384-
11520.png','b3_tile_13568-13312.png','b3_tile_13312-
12544.png','b3_tile_13568-9984.png','b3_tile_24320-
15104.png','b3_tile_24064-12544.png','b3_tile_24320-
12544.png','b3_tile_24576-12800.png','b3_tile_1280-17152.png' ]
evalimagelistkd = [x.replace('b3','kd') for x in evalimagelistb3]
```

```
kdlist =getsampleimages(kddir, evalimagelistkd)
b3list = getsampleimages(b3dir, evalimagelistb3)
evalimagelistkd[0]

# Plot all the values of the metrics with respect to the epochs (can
repeat for each metric by changing the panda being plotted)
plotgraph = r2.plot(x='epoch',figsize=(15,8))
graph = plotgraph.get_figure()
graph.savefig('/content/drive/My Drive/MASTERTHESE/visualizations/r2.png
')

ino = 2

# Read  a sample image and mask from the data-set
img = cv2.imread(str(b3list[8])).transpose(2,0,1).reshape(1,3,256,256)
mask = cv2.imread(kdlist[8])
with torch.no_grad():
    a = model(torch.from_numpy(img).type(torch.cuda.FloatTensor)/255)

# Plot histogram of the prediction to find a suitable threshold. From th
e histogram a 0.03 looks like a good choice.
plt.hist(a['out'].data.cpu().numpy().flatten())


# Plot the input image, ground truth and the predicted output in a row o
f 3 and save the figure
plt.figure(figsize=(10,10));
plt.subplot(131);
plt.imshow(img[0,...].transpose(1,2,0));
plt.title('Image')
plt.axis('off');
plt.subplot(132);
plt.imshow(mask);
plt.title('Ground Truth')
plt.axis('off');
plt.subplot(133);
plt.imshow(a['out'].cpu().detach().numpy()[0][0]>0.03);
plt.title('Segmentation Output')
plt.axis('off');
plt.savefig('/content/drive/My Drive/MASTERTHESE/visualizations/Segmenta
tionOutput9.png',bbox_inches='tight')

#Run modeleval over whole dataset

outfile = '/content/drive/My Drive/MASTERTHESE/fulldatarun/modelevals/pr
ediction_{}_{}.png'
inputdir = '/content/drive/My Drive/MASTERTHESE/fulldatarun/mbtiles'
```

```
for i in tqdm(images):
  split1 = i.split("/")[-1]
  split2 = split1.split("_")[-1]
  split3 = split2.split(".")[0]
  split4 = split3.split("-")
  colnum = split4[0]
  rownum = split4[1]
  img = cv2.imread(i).transpose(2,0,1).reshape(1,3,256,256)
  with torch.no_grad():
    a = model(torch.from_numpy(img).type(torch.cuda.FloatTensor)/255)
    predict = a['out'].cpu().detach().numpy()[0][0]>0.03
    predictimg = Image.fromarray(predict)
    predictimg.save(outfile.format(int(colnum),int(rownum)))
#Move worldfiles from the 3banded clips made in the data clipper
script to the folder of predicions (same spatial projection)
srcfolder = '/content/drive/My Drive/MASTERTHESE/fulldatarun/multipng_3b
'
destinationfolder = '/content/drive/My Drive/MASTERTHESE/fulldatarun/mod
elevals/'
for i in tqdm(worldfiledir):
  replace1 = i.replace('b3_tile', 'prediction')
  replace2 = replace1.replace('-',"_")
  replace3 = replace2.replace(srcfolder + "/", "")
  newfilename = os.path.join(destinationfolder, replace3)
  shutil.copy(i, newfilename)

destinationfolder = '/content/drive/My Drive/MASTERTHESE/predtifs'
sourcefolder = '/content/drive/My Drive/MASTERTHESE/fulldatarun/modeleva
ls'
sourcefiles = glob.glob(os.path.join(sourcefolder + "/*.png"))

#Translate projectionless pngs to projected tifs using the moved worldfi
les
for png in tqdm(sourcefiles):
  dirreplace = png.replace(sourcefolder,destinationfolder)
  newfilename = dirreplace.replace(".png", ".tif")
  gdal.Translate(newfilename, png)

#Zip just the tifs to a file (aux xml files ruin the merge later on)
!zip -
r /content/drive/My\ Drive/MASTERTHESE/predictions2.zip /content/drive/M
y\ Drive/MASTERTHESE/fulldatarun/modelevaltifs/*.tif

##After zipping, unzip and apply a merge on the tif files, followed by s
ampling raster values using the palm tree dataset in this study this was
done through QGIS with the merge and sample raster values tools
```

---Script end

Script: Datavisualization

Was used to create figures 8 and 21.

.ipynb used in google colab environment

---Script start

```
#install prerequisites
!pip install geopandas googletrans

#import required modules
import geopandas
import pandas as pd
import tqdm


#Set filepath to the point based vector dataset
datapath = "/content/drive/My Drive/MASTERTHESE/palmbomen/Palmtreedata_reprojected.geojson"

#Read vector dataset into geopanda
df = geopandas.read_file(datapath)

#Activate translator if data needs to be translated
from googletrans import Translator
translator = Translator()

#Group the palm trees on a to be researched variable and remove duplicate data
uniquevalues = df.groupby("ISLA").agg(pd.Series.nunique)

#Index requires resetting because it the grouped column as an index which we still need
u = uniquevalues.reset_index()

#Translate the required spanish data to english and rename Spanish columns to english (if grouped by a column containing spanish info)
u['Palmtree_surroundings'] = u['Tip_amb_de'].map(lambda x: translator.translate(x,src='es', dest="en").text)

#Rename unintuitive columns
unew = u.rename(columns={u.columns[0]:"Island", u.columns[1]:"Palmtree count"})

#Plot a bar chart (your column of choice)
plot = unew.plot(x = unew.columns[0], y = unew.columns[1], kind= "bar")

#save created figure
```

```
fig = plot.get_figure()
fig.savefig("/content/drive/My Drive/MASTERTHESE/visualizations/Island_e
ng.png", bbox_inches = "tight")
#####end of first part

#visualisation for modelpredictions
datapath2 = "/content/drive/My Drive/MASTERTHESE/fulldatarun/Palmtreespr
edictionsamples/palmtreepredictionssampled.geojson"
df2 = geopandas.read_file(datapath2)
#Get true positive prediction visualized for the whole dataset
#Group the palm trees on a to be researched variable and aggregate based
 on mean (0 is 0% correct, 1 = 100% correct)
uniquevalues2 = df2.groupby("Palmtree surroundings", as_index=False).agg
({'rvalue_1':['mean']})

#Translate if spanish data is present, create columns for TP, FN, TN
uniquevalues2['Palmtree surroundings'] = uniquevalues2['Palmtree surroun
dings'].map(lambda x: translator.translate(x,src='es', dest="en").text)
uniquevalues2['TP%'] = uniquevalues2['rvalue_1'] *100
uniquevalues2['FP%'] = 100 - uniquevalues2['TP%']
uniquevalues2

#plot TP values
plot = uniquevalues2.plot(x = 'Palmtree surroundings', y = 'TP%', kind=
"bar")

#save created plot
fig = plot.get_figure()
fig.savefig("/content/drive/My Drive/MASTERTHESE/visualizations/TPpredic
tions_environments_eng.png", bbox_inches = "tight")
```

---Script end