TÉCNICO LISBOA

Development of high-resolution maps of vegetation cover to support land planning and grazing management in fire prone landscapes

Bianka Trenčanová

Thesis to obtain the Master of Science Degree in

Energy Engineering and Management

Supervisors: Prof. Alexandre José Malheiro Bernardino,
Dr. Vânia Andreia Malheiro Proença

Examination Committee

Chairperson: Prof. Susana Isabel Carvalho Relvas

Supervisor: Prof. Alexandre José Malheiro Bernardino

Member of the Committee: Prof. Bruno Duarte Damas

January 2021

Abstract

The focus of this thesis is to develop a classifier of shrub vegetation cover. Shrubs are a key vegetation type in dry Mediterranean climates, that is associated with an increased risk of fire. The classifier will be further used for sustainable land planning and grazing management for fire prevention. Two main objectives are 1.) to design a new dataset from an unmanned aerial vehicle (UAV) imagery using ordinary RGB channels and 2.) to develop a method to increase the accuracy of a convolutional neural network (CNN) with a U-Net architecture to detect shrubs in a complex heterogeneous forest environment within a study farm in Portugal. The tested methods and their feasibility for this particular task are data augmentation, tiling, rescaling, dataset balancing and hyperparameter tuning (namely the number of filters, dropout rate and batch size). The biggest improvements were recorded with data augmentation, tiling and rescaling practices. The developed classification model achieves an average F1 score of 0.72 on three separate test sets even though it is trained on a relatively small dataset with some degree of inaccurate labels. It takes around four hours to train the model. The major challenges identified in this work were precise manual image annotation, small sample size, time and memory limits of used tools, and high intra-class and low inter-class variance of the target vegetation class. The main contributions of this study are evaluating the performance of the state-of-the-art CNN for mapping fine-grained land cover patterns from RGB remote sensing data and proposing a method to improve the model's performance.

Keywords: U-Net, convolutional neural network (CNN), shrub detection, heterogeneous land cover mapping, UAV imagery, Mediterranean forest

Resumo

O foco desta tese é desenvolver um classificador da cobertura arbustiva usando imagens obtidas por um veículo aéreo não tripulado (UAV). Em climas mediterrânicos, a expansão de biomassa arbustiva está frequentemente associada a um maior risco de incêndio. O classificador será usado posteriormente para apoiar o ordenamento e gestão sustentável de paisagens de pastoreio para prevenção de incêndios. Os dois objetivos principais são 1.) construir um novo conjunto de dados, a partir de imagens de um veículo aéreo não tripulado (UAV) usando canais RGB comuns e 2.) desenvolver um método para aumentar a precisão de uma rede neural convolucional (CNN) com uma arquitectura U-Ne t para detecção de arbustos num ambiente florestal, com cobertura do solo heterogénea e complexa, usando uma área de estudo em Portugal. Os métodos testados e sua viabilidade para esta tarefa são aumento de dados, *tiling*, reescalonamento, ponderação de conjunto de dados e ajuste de hiperparâmetros (ou seja, o número de filtros, taxa de dropout e tamanho do lote). As maiores melhorias foram registradas com as técnicasde aumento de dados, tiling e reescalonamento. O modelo de classificação desenvolvido atinge uma pontuação F1 média de 0,72 em três conjuntos de teste separados, embora o conjunto de dados de treino seja relativamente pequeno e contendo alguns rótulos imprecisos. O treino do modelo dura cerca de quatro horas. Os principais desafios identificados neste trabalho foram a anotação manual precisa da imagem, o pequeno tamanho da amostra, os limites de tempo e memória das ferramentas utilizadas e a alta variância intra-classe e baixa variância inter-classe da vegetação alvo. As principais contribuições deste estudo são a avaliação do desempenho do estado-da-arte da CNN para mapear a cobertura do solo em paisagem com uma textura fina (“fine-grained”), a partir de dados RGB de detecção remota, e propor um método para melhorar o desempenho do modelo.

Palavras-chave: U-Net, rede neural convolucional (CNN), detecção de arbustos, mapeamento de cobertura do solo de alta resolução, imagens de UAV, floresta mediterrânea

Acknowledgements

I am very grateful that I got the opportunity to contribute to a project whose objective is to bring sustainability into areas linked to a sector as critical to our society as the food production and to show that nature is a part of the solution. I would like to thank Dr. Vânia Proença for her guidance and willingness to share her knowledge. I hope my work will bring benefit to the project and move it closer to its goals.

I chose this thesis topic as an opportunity to learn something about a field that is gaining importance. Without any prior knowledge of machine learning, it was a challenging but enriching task and I would like to thank Prof. Alexandre José Malheiro Bernardino for his guidance and patience on my path.

I very much appreciate all the help and supervision provided to me even during the difficult times brought by the pandemics.

Last but not least, I would like to thank my friends for supporting me in good and bad times, especially during the quarantine. To Giacomo for his tireless helping hand, enlightened attitude and never stopping motivation, to Yasmine for her endless optimism, to Vasco for the scholarly talks; to Berto for sharing the journey and to Giorgia for her peculiar encouragements.

This work was partially supported by projects GO SILVPAST - Implementação custo-eficiente de mosaicos silvo-pastoris de carvalho negral (PDR2020-101-031873), and FCT project FIREFRONT (PCIF/SSI/0096/2017).

Table of Contents

List of Figures

List of Tables

List of Acronyms

ANN	Artificial Neural Network
CNN	Convolutional Neural Network
CO_2 & CO_2-eq	Carbon dioxide & carbon dioxide equivalent
FN	False Negative
FP	False Positive
GHG	Greenhouse gas emissions
HR & VHR	High Resolution & Very High Resolution
IOU	Intersection Over Union
LR	Learning rate
LULC & LULUCF	Land-use-land-cover & Land Use, Land-Use Change, and Forestry
ML	Machine Learning
NIR	Near Infrared (band)
ReLU	Rectified Linear Unit
RGB	Red, Green, Blue (band)
ROI	Region of Interest
SDG	Sustainable Development Goals
TN	True Negative
TP	True Positive
UAV	Unmanned Aerial Vehicle
URI	Uniform Resource Identifier
VIS	Visible Range

1 Introduction

1.1 Background and motivation

Human activities have a tremendous impact on the environment. A dramatic population increase from around $1*10^9$ in 1800 to $7.8*10^9$ in 2020[1] and a society based on consumerism lead to ever growing needs and demands of humankind, which exerts unprecedented pressure on Earth systems. Notable consequences of Anthropocene-induced climate change have made us recognize our responsibility and defects in our structures. Numerous attempts have been made in order to redirect our civilization towards a more just and sustainable path. Well known examples are the United Nations' Sustainable Development Goals (SDGs, 2015)[2] and the Planetary Boundaries (2009)[3], a framework proposed by scientists to guide sustainable development within safe environmental limits. Issues addressed in SDG 2 (Zero hunger) and SDG 12 (Responsible consumption and production) and in Planetary Boundaries (mainly Biogeochemical flows, Biosphere integrity loss, and Land-system change) are directly linked to our food systems, with the two mentioned boundaries being already far beyond the Zone of high risks and the third one approaching it.

Food production is one of the major contributors to degradation of the environment. This sector accounts for approximately 26% (13.598 Gt CO_2-eq/yr) of global greenhouse gas emissions (GHG), out of which one half (6.93 Gt CO_2-eq/yr) comes from crop production and land use, linked to turning natural ecosystems such as forests and grasslands, that act as carbon "sinks", into cropland and pastures, that release additional carbon dioxide (CO_2)[4]. There is a tremendous opportunity cost to this as well, since insensible land use does not only release additional CO_2, but also significantly decreases the amount of CO_2 potentially sequestered by natural vegetation. Between years 2000 and 2011, there was a global increase of 6% (from 3.2 Gt CO_2/year to 3.4 Gt CO_2/year) in the amount of carbon sequestration loss due to agriculture and forestry, with forestry activities like logging contributing the biggest share (30%) to the total carbon sequestration loss (Marques et al., 2019). Complete decomposition of individual GHG emission contributors within the food production sector can be found in Figure 1. The food production sector is also responsible for nitrogen and phosphorus pollution, biodiversity loss, and water and land use, eroding the stability of the Earth system.

After the Kyoto Protocol, a worsening environmental situation had led to creation of yet another famous document trying to mitigate climate change in 2015, the Paris Agreement. Its target is to keep the global average temperature increase below 2°C by 2100, relative to 1861 – 1880 temperatures. To make this goal attainable a carbon budget for Land Use, Land-Use Change, and Forestry (LULUCF) translates to an allowance of emitting 5 Gt CO_2-eq/yr by 2050 and then transiting into a net carbon sink absorbing 10

[1] https://www.worldometers.info/world-population/
[2] https://sdgs.un.org/goals
[3] https://www.stockholmresilience.org/research/planetary-boundaries.html
[4] https://ourworldindata.org/food-ghg-emissions#:~:text=They%20are%20the%20direct%20emissions,for%2024%25%20of%20food%20emissions.

Gt CO_2-eq/yr by 2100 (Willett et al., 2019). Agricultural systems can become carbon sinks, but some emissions-producing biological processes that are intrinsic to agriculture, such as ruminant livestock's digestion producing methane (CH_4) or soil microbes emitting nitrous oxide (N_2O), imply that agricultural GHG emissions cannot be eliminated entirely, which makes this task even more challenging. That's why both bottom-up and top-down approaches are needed. While the former stands on a consumption side and is a responsibility of every individual, the latter requires structural change of the food production systems. Proper sustainable management practices that are aligned with Earth system processes urgently need to be developed and adapted on a large scale. Better harnessing of ecosystem services such as pest control, pollination, water regulation, and nutrient cycling, will lead to higher productivity and resilience, and at the same time will reduce harmful environmental impacts of this sector. Cattle farming, which is a big part of our current food production industry, is not only the biggest contributor of GHG emissions in the food production sector (31% along with fisheries, see Figure 1), but has also the highest impact on biodiversity, that contributed to approximately 28% of total impending bird species extinctions in 2011 (Marques et al., 2019).

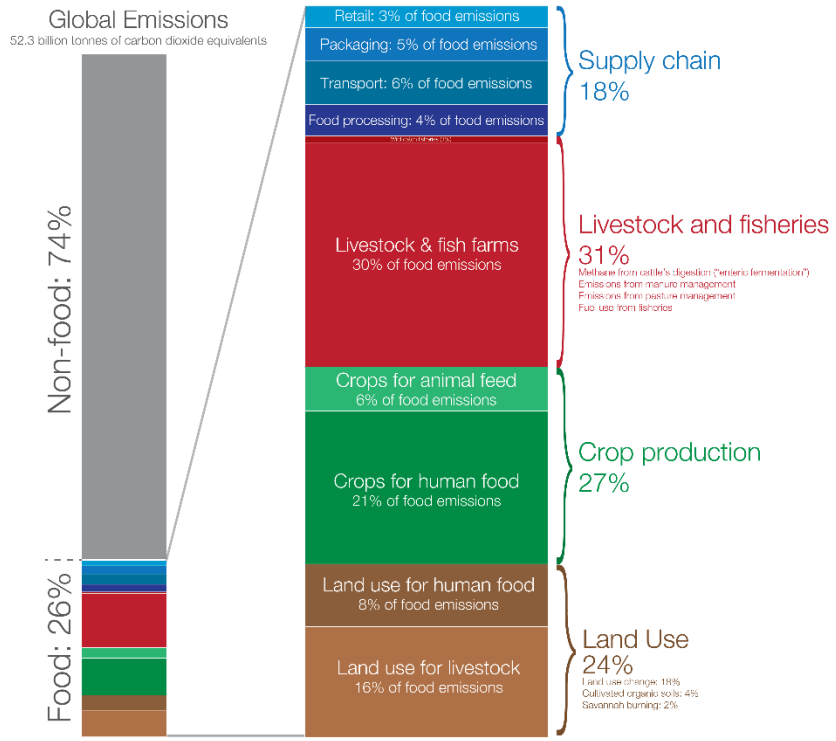


Figure 1 Global greenhouse gas emissions from food production (Source: Our World in Data, 2019)

However, traditional livestock systems also play a role in biodiversity conservation, climate adaptation, and socioecological resilience at regional and local scales (Proença & Teixeira, 2019).Ecological processes, such as nutrient cycling, soil fertilization, maintenance of genetic diversity and regulation of vegetation growth, once supported by wild large herbivores (Ripple et al., 2015), are now sustained by free-range livestock in areas where wild large herbivores are scarce or no longer present. However, strong socio-economic drivers stimulate rural-urban migration, leading to an extensive abandonment of

agricultural land. The absence of large herbivores and the withdrawal from human activities increase fuel loads and promotes homogenization of vegetation in the affected areas. Shrub growth, after the cessation of land use, increases the susceptibility of the landscape to fires, and is further s enhanced by wildfires due to its resprouting ability (Rey Benayas, 2007). This can establish a positive feedback loop, leading to vast declines in biodiversity and change in natural fire dynamics, especially dangerous in Mediterranean Basin, which has already high fire intensities and frequencies due to its dry climate and impacts of global warming in recent decades. Only in 2017, wildfires that destroyed areas in the European Natura 2000 network, incurred damage of 10 billion euros (Kinaneva et al., 2019). That is why active re-introduction of herbivores into fire prone regions could serve as an environmentally sustainable and time and cost-effective method for wildfire prevention. Prescribed (or targeted) grazing is a silvopastoral practice that promotes heterogeneous landscapes, controls shrub encroachment and is officially considered as a wildfire prevention tool (Lovreglio et al., 2014).

However, such interventions require thorough land planning, preventive management and regular monitoring for which a detailed land cover mapping is essential. Remote sensing is a primary source of data for vegetation mapping and thanks to continual developments in geo-information technologies this field is gradually becoming more universal. Limitations that satellite-based systems face, such as insufficient spatial, spectral and temporal resolutions, cloud cover or high cost of data acquisition, are resolved with the emergence of a new remote sensing aerial platform – unmanned aerial vehicles (UAVs). UAVs have very high spatial resolutions thanks to their low speed and flight altitude, they are cheaper, more flexible in obtaining data from target areas that are often difficult to reach, they minimize disturbances of inspected areas and provide real-time monitoring (Pérez-Rodríguez et al., 2020). Acquired data is often used in combination with Artificial Neural Networks (ANNs), that have the capacity to speed up evaluation process of the input information even over large datasets. That is why these methods are becoming a fundamental tool in numerous fields from wildlife conservation and management and various agricultural applications to fire detection.

1.2 Case study: Quinta da França

This thesis uses a case study farm, Quinta da França, that integrates agricultural and forest land uses. The farm's management is guided by sustainability principles. The farm is managed by Terraprima Agrícola. Terraprima is a business group formed by Terraprima - Serviços Ambientais (Environmental Services) and Terraprima - Sociedade Agrícola (Agricultural Society), promoting environmental services provided by agroforestry activities. Terraprima Ambiental is a spin-off of the Instituto Superior Técnico, dedicated to the design and implementation of integrated systems to compensate for environmental impacts resulting from human activities. They are involved in the management of projects for remuneration of farmers delivering environmental services through good soil management practices. Terraprima Agrícola was established in 1994 and since then has been managing a farm Quinta da França, whose sustainable forest management is a part of Terraprima's general long-term endeavors to demonstrate sustainable management of rural areas, while maximizing economic profitability,

ecosystem services and benefits for collaborators of projects related to the environment and sustainability[5].

On the farmland, Quinta da França exercises integrated production, which is an agricultural system for the food production that favors protection of the environment and the consumer, producing high quality products with rational management of natural resources, contributing to development of sustainable agriculture[6]. It maximizes synergies between forest production and agricultural production, that enhances multiple environmental services. A biodiverse pastures system, invented by Portuguese researcher Davis Crespo in the 70's, is implemented in non-forest areas of the farm. It mixes different plant species, which increases organic matter in the soil and better retains CO_2. Besides that, CO_2 is also captured by the forest area, thanks to what Quinta da França successfully demonstrated that the provision of environmental services (in this case natural agroforestry carbon sink) can be a competitive agricultural market product, alongside conventional food production, when in a big emission offset program in collaboration with EDP that ran from 2006 to 2012, its forests were used to sequester 7000 tons of CO_2/yr , aspiring to achieve the goals set by Portugal under the Kyoto Protocol[7]. Agricultural activities of the farm are a complement to raising livestock, which is the main part of the portfolio. Innovative management techniques such as monitoring of the movement, production and reproduction of the animals are implemented. The stress is put on animals' welfare and the animal feed is heavily based on grazing, with little supplementation of compound feed (less than 4% of the total feed) (Simões, 2019).

The forest area includes a semi-natural oak forest. Two big fire events on the farm's site, one in 80's and the other in 1996, wiped out around 200 ha of the original oak forest, that has been since then regenerating mostly through natural processes. The management of the forest is focused on the reduction of fire risk, increase of carbon sequestration, and biodiversity conservation. Vegetation cover and level of development is heterogenous, from areas with an already developed tree cover to open areas dominated by shrubs. Because the forest is relatively young (25-40 years after fires), young trees are dominant and often accompanied by dense understory, which increases their vulnerability to fire spread and requires management measures to reduce that risk, namely the regular removal of shrub cover. The use of livestock for biomass regulation is now being implemented as a part of an ongoing project by the SILVPAST Operational Group[8], which is aimed at the sustainable management and restoration of the oak forest. Grazing management, enhanced by monitoring of animals' real-time location with GPS collars, is expected to contribute to better soil fertilization, through nutrient inputs and recycling, further helping with soil restoration and CO_2 retention. Besides that, targeted grazing could help to regulate the vegetation structure, open trails, trample down shrubs and consequently reduce the risk of fire. The potential impact of the presence of animals is now being investigated also within the forest site. A possibility to install biodiverse pastures in forest clearings, in order to maintain open

[5] https://www.terraprima.pt/en/sobre-nos/
[6] https://dre.pt/web/guest/pesquisa/-/search/259760/details/normal?q=DecretoLei+n%C2%BA%2037%2F2013
[7] https://www.terraprima.pt/en/projecto/13
[8] https://www.terraprima.pt/pt/projecto/23

(firebreak) areas through grazing (by both, wild herbivores and cattle), has been brought up but has not yet been implemented.

Furthermore, the Forest Fire Protection Program and Fuel Management is an important part of the Quinta da França's agenda. Along with the periodic shrub control, the other important interventions are a good road network and strategic distribution of low fuel content plots (e.g. fuel management strips, agricultural plots or rocky pastures and outcrops). That is why grazing management and land planning, such as implementing forage spaces for cattle in the forest area, in conjunction with the forest's fire protection structure is one of the exploration objectives for the fuel management, that has a potential to reduce the costs of vegetation control and fire prevention (Terraprima -Sociedade Agrícola Lda., 2012).

1.3 Challenges

To implement grazing and vegetation management in the most efficient way, regularly updated land cover maps are necessary. Mapping the vegetation cover is, however, a challenging task, especially so in case of shrubs. Shrubs, or bushes, are a very broad and heterogeneous group of perennial woody plants. They come in various shapes and sizes and form complex clusters of individuals, which makes it difficult to map and monitor their growth. Moreover, unless they clearly stand out from their environment, such as in a desert (Guirado et al., 2017), they are difficult to delimit from the surroundings (Ayhan & Kwan, 2020; Hellesen & Matikainen, 2013). This becomes especially an issue in an intricate diverse forest environment containing a lot of mixed classes and unclear boundaries among them. Big intra-class variance and at the same time low inter-class overlapping of shrubs' spectral signatures make the detection harder even for machine learning models and often lead to misclassification of vegetation types. The research on land cover mapping of complex ecosystems containing mixed vegetation classes is scarce, which leads to an absence of labelled datasets in this domain and thus the need to create them for a specific task by oneself. Due to the features of shrub vegetation and of the land cover patterns, this task is a time-consuming process, but also very strenuous in terms of visual recognition from remotely sensed imagery. On the top of that, depending on the time and cost constraints, it may lead to an insufficient amount of labelled data and thus the need to artificially increase their volume by heavy data augmentation.

1.4 Objectives

The aim of this thesis is to develop a method for high resolution mapping of land cover in a forest area with heterogenous land cover composition, with a focus on fire prone shrub vegetation. A classifier of the target vegetation type (i.e. shrubs) in the areas susceptible to fire will be created based on exemplary data from UAV imagery, that can recognize the corresponding patterns in new images. Maps of vegetation cover will then serve as a foundation for better informed landscape planning and grazing

management and for research of innovative ways of integrating livestock production, biodiversity conservation and fire prevention in fire prone landscapes in the Mediterranean regions.

The main objectives and contributions of this work are:

I. Classification of fire-prone vegetation type (shrubs) from natural color UAV images – creating manually labeled dataset for training, validation and testing, using semantic segmentation;
II. Using supervised learning approach to train a CNN (U-net) to automatically detect the key vegetation type in new images;
III. Developing a method to increase the detection accuracy of shrubs in the specific type of ecosystem;
IV. Evaluating the feasibility and performance of the detection of an irregular shrub cover in a complex heterogeneous landscape.

1.5 Structure of the thesis

This work is organized in seven sections as follows: Section 1 consists of a general introduction, motivation and presents the case study. A review of main concepts and related works in the field is provided in section 2. A description of applied methods, used materials and their development can be found in section 3. Section 4 explains further in detail the performed experiments, their purpose, underlying assumptions and the used data. Results of the experiments are then presented in section 0 and discussed more in depth in section 6. Lastly, conclusions and recommendations for a future work are presented in section 7.

2 Related work

Technological advancements have significantly improved our understanding of the planet. Developments in fields such as remote sensing, computer vision, deep learning, and computer hardware bringing low-cost and high-performance GPUs are interrelated and gave rise to new possibilities of monitoring the earth surface. The image classification that recognizes the content of aerial images plays an important role in applications as diverse as updating maps, improving urban planning, assessment of land use changes, environment monitoring and even disaster relief. Rapid developments in remote sensing are bringing along ever-growing volumes of unlabelled high-resolution data that are unfeasible to process by humans and still pose a challenge for the computer vision to accurately interpretate them. Other challenges include small labelled datasets of interest, the character of data and the challenges resulting from the used machine learning algorithm (unsupervised and supervised). While unsupervised learning methods cluster scenes of interest, with supervised methods the model in trained on specifically hand-crafted features that describe the image content locally (Volpi & Tuia, 2017). Supervised classification is the state-of-the-art method of land cover mapping (Stoian et al., 2019) and is used in this thesis.

2.1 Land cover mapping

Satellite remote sensing is an effective way of acquiring data for various land cover mapping applications (Ahmed & Noman, 2015; Fröhlich et al., 2013; Kussul et al., 2017; Vanjare et al., 2014). Different satellites have different qualities, for example Sentinel-1 offers high spatial resolution, while Sentinel-2 offers high revisit time (Gbodjo et al., 2020). However, satellites are generally continuously improving in these terms and some studies (Gbodjo et al., 2020) are also trying to exploit the fusion of multi-source data to benefit from the different features and improve the performance. Satellites have an advantage of the capacity to map large areas at the same time, but their biggest drawback is resolution that is still coarse for some applications, they suffer from cloud cover and they are limited by fixed-timing and costly data acquisition (Matese et al., 2015). These issues are resolved by a newer platform of unmanned aerial vehicles (UAVs).

Although originally developed for military purposes, drones, or UAVs, have become an important commercial tool for monitoring, revolutionizing the acquisition of fine-grained data thanks to their high spatial resolution. It is a low-cost, low-impact solution that is highly flexible and enables data collection also in difficult to access areas. This versatile technology can be used for monitoring and analysis of small ecosystems to large areas and even a climate change. Therefore, UAVs found their place in various fields including ecology and conservation of wildlife (Getzin et al., 2012; Mangewa et al., 2019), agriculture and forestry (Csillik et al., 2018), firefighting (Kinaneva et al., 2019) and also disaster zone mapping (Kerle et al., 2019). Unlike satellites, UAV-based mapping is often conducted at a local scale (Brandt et al., 2020) and faces several other technical challenges. Some of their biggest disadvantages

are power limitation, low flight time, small payload, low spectral resolution and sensitivity to atmospheric conditions (Mangewa et al., 2019; Paneque-Gálvez et al., 2014). Important are also accurate positions and flight heights, that influence the sensor's accuracy. E.g. flying at higher altitudes makes observations more sensitive to the vehicle's motion and can cause motion blur (Hung et al., 2014). Flight mission planning and execution are crucial and the photogrammetric processing of the imagery is, due to variations in levels of image overlap and relief displacement, challenging as well. Last but not least drones pose serious ethical, security and safety issues.

The applications of remote sensing imagery solely in vegetation assessments are very diverse, including the monitoring of species after fire events (Pérez-Rodríguez et al., 2020; Sankey et al., 2017) and the health condition of vegetation (Baena et al., 2017; Malenovský et al., 2017), mapping of ecosystem structure and function (Langford et al., 2019), plant communities (Lopatin et al., 2017) and the species at individual level (Cao et al., 2018), assessing biodiversity (Getzin et al., 2012), plant diseases (Sladojevic et al., 2016) and many more.

2.2 Bands, indices and data fusion

When it comes to land cover mapping, utilizing different bands and their combinations, so called indices, can be a powerful tool for identification of many classes. Specifically, for the classification of natural land cover types useful bands are blue (448-510 nm), that differentiates soil and rock surfaces from vegetation; green (518-586 nm), that separates vegetation (such as forest or croplands with standing crops) from soil; red (640-670 nm), that senses chlorophyll absorption, discriminates vegetation and soil and highlights barren lands; yellow (590-630 nm), that separates vegetation and soil, highlights barren lands and separates croplands with standing crops from bare croplands with stubble. Healthy plants reflect NIR (772-954nm), thus it is a great band to use in ecology purposes or for estimating the burn severity (Pérez-Rodríguez et al., 2020). Information from this band is essential for important indexes like normalized difference vegetation index (NDVI), that is widely used to assess the presence and health state of a vegetation. SWIR (1195-2365nm) is another band, typically present in satellite imagery, that is suitable for distinguishing wet from dry earth and rocks from soils (Iglovikov et al., 2017).

More bands naturally contain more spectral information, that is why multispectral data are popular in land cover classification tasks (Ashapure et al., 2019; Mahdianpari et al., 2018; Pérez-Rodríguez et al., 2020). However, other types of data offer different qualities, such as high-resolution information present in panchromatic images, which makes data fusion an interesting approach to exploit all the useful properties of the available data. Gaetano et al. (2018) fuses high-resolution panchromatic and multispectral data in their two-branch neural architecture MultiResoLCC, which shows better performance on land cover classes such as orchards, meadows, herbaceous savannah and different types of crops. Iglovikov et al. (2017) went even further and in their work fused not only panchromatic and multispectral images, but also RGB channels and reflectance indices.

2.3 Convolutional neural networks

Advancements in remote sensing and the growing availability of remote sensing data require automated solutions to process such large volumes of information. The answer comes in the form of convolutional neural networks.

Convolutional neural networks, CNNs or ConvNets, are state-of-the-art deep learning algorithms mimicking biological neural structures developed for image processing and computer vision tasks (Kattenborn et al., 2020). While the history of CNNs has started decades ago, they were not practical due to their memory requirements and the lack of available training data in the past. Recent developments in the computer vision, graphical processor units (GPUs) and growing amounts of data that are publicly available resulted in their revival in 2012 (Reina et al., 2020). Their capacity to handle growing quantities of earth observation data in an automated way makes them a promising solution in many fields. However, their drawback of requiring huge amounts of training data remains and poses serious problems especially in cases of image segmentation, where training data acquisition is still expensive and scarce. Creating pixel level segmentation masks is complicated, labour and time intensive process (Ulmas & Liiv, 2020) and faulty labelling can become a great limiting factor (Rakhlin et al., 2018). Even if it is still possible for the model to correctly identify a class even though it was labelled incorrectly, as e.g. in the case of (Rakhlin et al., 2018), it will worsen the performance results after comparing with the ground truth. Moreover, faulty labelling can become a serious danger in fields like medical imagery (Ibtehaz & Rahman, 2020; Litjens et al., 2017) or autonomous driving (Treml et al., 2016). Some of the ways to overcome the lack of training data are using a weakly supervised learning method (Nivaggioli & Randrianarivo, 2019; Wang et al., 2020), transfer learning or a data augmentation (Scott et al., 2017), that will be explained more in section 2.9. Other challenges of CNNs include complicated tuning process, tendency to overfitting and still high computational requirements.

CNNs are an end-to-end solution that automatically learns local feature extractors over many examples at different spatial scales (Flood et al., 2019), without the need for a feature engineering, improves generalization and decreases the number of parameters due to weight sharing (Flood et al., 2019; Nogueira et al., 2015). Their ability to encode spectral as well as spatial information makes them superior to standard classifiers, such as random forests, that work solely with the spectral information (Diakogiannis et al., 2020). CNNs have been successfully applied for image and scene classification, segmentation and object detection (Hu et al., 2015; Stoian et al., 2019; P. Zhang et al., 2018; W. Zhang et al., 2019).

A typical CNN consists of a stack of convolutions, activation functions and pooling layers. Convolutions are an essential step in which the features are learnt in a hierarchical manner – first layers extract low-level features, such as edges, lines and corners, while deeper layers learn increasingly more complex features such as shapes, structures and entire objects (Nogueira et al., 2015). A non-linearity is introduced by an activation function, the most popular one being ReLU (*max(0, x)*). Pooling layer then reduces the dimension of the extracted features, fostering translation invariance (P. Zhang et al., 2018). The most popular technique is MaxPooling that acts as a noise suppressant. It selects only dominant

features, making the training more robust and decreasing the required computational power. Follows the fully connected layers and a classifier layer, softmax, that outputs a vector of scores or probabilities for each class. The training of CNNs is generally based on the prediction loss minimization, a loss function that measures the difference between the output of the final layer and the ground truth (Guirado et al., 2017). The most commonly used performance metric is accuracy, that is defined as:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

where TP are true positives, TN true negatives, FP false positives and FN false negatives.

Another common metric is F1 score, a class-specific measure of segmentation accuracy, suitable for unbalanced datasets. It is the geometric mean between precision (user's accuracy) and recall (producer's accuracy) (Volpi & Tuia, 2017), defined as follows:

$$F1\ score = 2\ x\ \frac{Precision\ x\ Recall}{Precision + Recall}$$

where:

$$Precision = \frac{TP}{TP + FP}$$

and:

$$Recall = \frac{TP}{TP + FN}$$

Some of the well-known architectures include LeNet, AlexNet, VGGNet, GoogLeNet, ResNet.

A common way of boosting the CNNs' performance in land-cover classification is to combine them with the height information (Maltezos et al., 2017) and a digital surface model (DSM) (Längkvist et al., 2016), but there is an emerging group of neural networks that focuses specifically on semantic segmentation tasks, i.e. pixel level segmentation, which is especially efficient in land-cover classification.

2.4 Segmentation networks

There are five types of image analysis based on the granularity of understanding the images [9]. The most coarse-grained is classification (Krizhevsky et al., 2017) of an entire image, outputting a discrete label. Follows classification with localization (Sermanet et al., 2014), which outputs a discrete label and a localization information usually in form of parameters of a bounding box (B-box). Object detection (Girshick, 2015), that unlike the previous case can classify and localize more than one object in an image. Semantic segmentation (Q. Zhou et al., 2019), that labels each pixel of an image with a corresponding class and the resulting HR map is typically of the same size as the input image – a so

[9] https://nanonets.com/blog/semantic-image-segmentation-2020/

called dense prediction. Finally, instance segmentation (Graham et al., 2019) is also a pixel level classification but unlike the previous type it classifies each instance of a class separately.

Semantic segmentation is the most interesting one for the land cover classification tasks, able to learn also spatial configuration of labels and class-specific structures (Volpi & Tuia, 2017). The detection can be either of one specific class (Wen et al., 2017) or multiple classes at the same time (Paisitkriangkrai et al., 2016). Two big remaining challenges of the existing methods are intra-class inconsistency and inter-class indistinction (Yu et al., 2018).

One of the main research topics nowadays is how to provide pixel-level high-resolution segmentation. Two approaches try to address this problem – 1.) using dilated (atrous) convolution and 2.) connecting pooling and un-pooling layers, e.g. DeconvNet, SegNet or U-Net (Li et al., 2018). Among the first networks focusing on semantic segmentation was a fully convolutional network (FCN) (Long et al., 2015). It uses traditional CNN as a feature extractor but replaces the fully connected layers with up-convolutions, producing spatial feature maps instead of classification scores, that are further up-sampled to a dense pixel-wise output. Improvement of the FCN is already mentioned SegNet (Badrinarayanan et al., 2016), that consists of an encoder part, extracting spatial features, and a decoder part, up-sampling the feature maps. Similar to FCN and SegNet is a fully convolutional semantic segmentation network U-Net (Ronneberger et al., 2015), that will be discussed further in the next section. SegNet and U-Net are able to densely label every pixel at the original resolution of the image thanks to their down-sample-up-sample architecture. High-level representations are learnt via convolutions and then up-sampled back to the original resolution via deconvolution. These nets are computationally efficient and able to learn spatial dependencies among classes. Their drawback is low geometric accuracy (Stoian et al., 2019). Other approaches are presented by Audebert et al. (2018) and their multi-scale FCN or L.-C. Chen et al. (2017) DeepLab with atrous convolutions for the semantic segmentation.

2.5 U-Net

Building on so called skip connections first introduced by Long et al. (2015), Ronneberger et al. (2015) created U-Net (hereinafter the original U-Net), an improved FCN that works with very few images from a biomedical field. The combination of low level features with detailed spatial information and high level features with semantic information improving segmentation accuracy, makes it a good choice for one-class segmentation tasks (P. Zhang et al., 2018). The architecture consists of two symmetric paths – contracting (left side) and expansive (right side), which give the network its characteristic U-shape. The contracting path is a typical CNN architecture – it is a stack of two consecutive convolutions followed by rectified linear unit (ReLU) and max pooling operations. This is the down-sampling part of the network, where at each step the number of feature maps (kernels) doubles so that the network can learn more complex features in the image, however at the cost of losing localization information. By increasing the receptive field information of multiple scales (local and global) is gained and fused together (Zheng et al., 2016). The expansive path, on the other hand, applies a sequence of skip connections that, unlike

FCN that sums the features (Z. Zhou et al., 2020), concatenate the output of transposed convolutions and corresponding feature maps from the contracting path; followed by two consecutive regular convolutions with ReLU. This is the up-sampling part of the network, that acts as a compensation for the previous max pooling layers. The localization information is reconstructed here and more precise output is yielded. A final layer is (1 x 1) convolution that outputs densely labelled segmentation map with a size equal to size of the input image. The architecture is depicted in Figure 2. Because max pooling uses (2 x 2) sized filters, the input image has to always have an even height and width size. Authors used large input tiles and reduced the batch size to a single image. Data augmentation, especially elastic deformations that simulates a common tissue variation, was applied to the dataset. The model was used with training datasets of 30 (512 x 512) px fully annotated images, and 35 and 20 partially annotated images. The intersection over union (IOU) for partially annotated datasets was 92% and 77.5%, respectively.

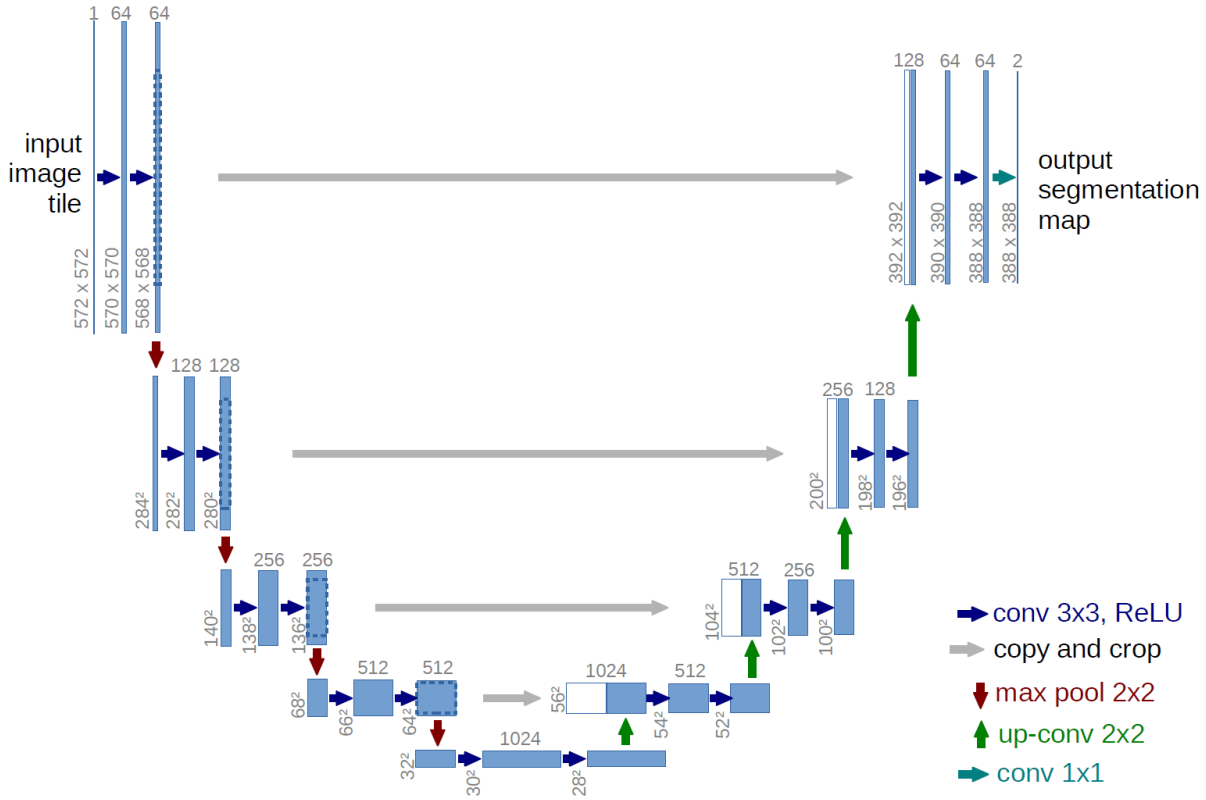


Figure 2 The original U-Net architecture. Blue boxes – multi-channel feature maps, white boxes – copied feature maps, the no. of the channels is denoted at the top of the box, height and width at lower left edge of the box, the arrows denote different operations. (Source: Ronneberger et al. (2015))

The encoder-decoder networks are widely used for semantic and instance segmentation (Volpi & Tuia, 2017; Z. Zhou et al., 2020). There are other examples of feature fusion methods than skip connections, include wiring feature maps into a sort of grid in GridNet (Fourure et al., 2017); employing two streams in the network – pooling, that carries the context information and residual, that carries full-resolution information (Pohlen et al., 2016); and variations of this network (Jiang et al., 2019). U-Net has become the state-of-the-art model for biomedical image segmentation tasks, but because of its ability to exploit

both texture and spatial structure in high resolution imagery, it is used in other applications like land cover classification, too (Garg et al., 2019).

There are many variations of its architecture. Zhou et al. (2020) presented UNet++, a built-in ensemble of U-Nets of varying depths that partially share an encoder and have intertwined decoders. The architecture overcomes the problem of unknown optimal depth of the network for different applications and restrictive design of skip connections. The feature aggregation in decoders is more flexible and the outputs are formed gradually, which helps Unet++ to outperform other compared U-Net architectures. Chen et al. (2019) in their Channel-UNet proposed a spatial channel-wise convolution along the channel of feature maps to extract the spatial information. By converging this information and feature maps from the original U-Net that serves as a backbone, the network effectively mitigates over- and under-segmentation problem in medical images. Another experiment with network connections are Plus connections between the successive Down and Up blocks in a DeepUNet (Li et al., 2018), that avoid the convergence on the local optimal solution, improving the performance of very deep networks in complex image segmentation tasks. MultiResUNet (Ibtehaz & Rahman, 2020) better handles medical images with noises, perturbations or lack of clear boundaries and ResUNet-a (Diakogiannis et al., 2020) enhances the understanding capability of the network by including pyramid scene parsing pooling, residual connections, atrous convolutions and multi-tasking inference, similarly to ASPP-Unet (P. Zhang et al., 2018), that learns contextual information at multiple scales using Atrous Spatial Pyramid Pooling technique. The newest and most exciting modification is nnU-Net (Isensee et al., 2020), a network that automatically configures itself, including pre-processing, network architecture, training and post-processing for any new task.

2.6 Shrub cover mapping

The research on how state-of-the-art classification tools perform in complex land cover mapping tasks is generally scarce (Mahdianpari et al., 2018). Shrubs class is a very general and heterogeneous group of vegetation with individuals of variable shapes, sizes, and distribution patterns, forming irregular and complex clusters of individuals (Guirado et al., 2017). High intra-class and low inter-class variance is a challenge causing difficulties to distinguish them from their surroundings (Hung et al., 2014) or other vegetation classes. Mahdianpari et al. (2018) used multispectral data, containing more complementary information, as a way to alleviate the problem of classification of spectrally similar vegetation types. They also found InceptionResNetV2 as the most efficient state-of-the-art convnet (compared to DenseNet121, InceptionV3, VGG16, VGG19, Xception and ResNet50) for classifying complex multispectral remote sensing wetlands scenes, when it reached an F1 score of 93%. In their pursuit of maximizing the distinction between the target vegetation type (weeds) and the surroundings, Hung et al. (2014) proposed to consider phenological stage highlighting the differences in the vegetation appearance as the most promising approach, but also performing the survey at lower flight altitudes (below 100m (Ashapure et al., 2019)) or using higher resolution sensor to obtain more detail.

A study with similar objective to this work – shrubs detection is (Guirado et al., 2017). Objects of interest are *Ziziphus lotus* shrubs, however, it is surrounded by bare soil with sparse vegetation unlike shrubs in my case, that are located irregularly in a complex heterogeneous landscape. After combining GoogleLeNet with data augmentation, transfer learning (fine tuning) and pre-processing, F1 score of 97% was achieved. Pre-processing techniques improving the detection performance the most were background elimination and long-edge detection, and only random flipping, scaling, cropping and brightness were used for data augmentation.

2.7 Hyperparameters

Hyperparameters are variables which determine the structure of the network and how is it trained. Among the most commonly tuned hyperparameters related to network structure are number of hidden layers and units (or neurons), dropout and activation function. Adding layers between the input and output layer and increasing the number of units can prevent underfitting and generally improve the performance, however, deeper neural networks might become more difficult to train (Bengio, 2012). Garg et al. (2019) used 19 padded 5x5 convolutional layers in their mUnet and outperformed state-of-the-art U-Net and fully convolutional neural network (FCN) in a land-use-land-cover (LULC) classification task. P. Zhang et al. (2018) found using 64 initial feature maps and 11 layers as optimal for the overall accuracy, regardless of the input image. Dropout is a regularization technique that randomly 'drops out' a given percentage of neurons to avoid overfitting. Co-adaptations among neurons are reduced and each neuron is made to learn more robust feature extractors (F. Zhang et al., 2015). Generally, the values are between 20-50%[10], lower dropout rates can have too little impact, while too high ones can cause an inefficient learning. Activation function introduces non-linearity into learning. Fast rectifier activation function (ReLU) is the most popular choice for hidden layers, while sigmoid is commonly used in the output layer of binary and softmax of multi-class predictions. Optimizers belong to this category of hyperparameters, too. Well-known Stochastic Gradient Descent (SDG) has a downside of a need for a good learning rate tuning, which is solved with optimizers such as Adam, that have adaptive learning rate. Adam is also computationally efficient, not memory-demanding and can therefore handle well large data or a lot of parameters (Kingma & Ba, 2017). Data preprocessing, that is further addressed in the next section, can be also viewed as a model hyperparameter.

Learning rate, number of epochs or batch size are only some examples relating to the training. The speed of updating network's parameters is defined by learning rate. Large values lead to fast learning but risk to miss the minimum of the loss function ('Exploding gradient'), while too low values slow down the training and may also fail to converge ('Vanishing Gradient')[11]. Generally, 0.01 (Bengio, 2012) is a good starting point, but decaying learning rate is the optimal solution. The number of epochs determines

[10] https://towardsdatascience.com/what-are-hyperparameters-and-how-to-tune-the-hyperparameters-in-a-deep-neural-network-d0604917584a
[11] https://towardsdatascience.com/neural-networks-parameters-hyperparameters-and-optimization-strategies-3f0842fac0a5

how many times the network sees the whole training dataset during learning. Decreasing validation accuracy despite increasing training accuracy is a sign of overfitting and the training should be stopped. This is another popular hyperparameter to tune but also brings along the dilemma of an increased performance at the cost of an unevenly increased training time. Hussain et al. (2019) yielded 5.5% higher accuracy with increasing the number of epochs fourfold, which came at the expense of a longer training time. Batch size is the number of samples in subsets of training data that affects the speed of learning. Larger batch sizes tend to slow down the convergence and generalize worse on test data, creating a so-called 'generalization gap'. Smaller batch sizes generally perform better, with 32 as a good default value (Keskar et al., 2017; Masters & Luschi, 2018). While some (Bengio, 2012) argue that it can be optimized independently of other hyperparameters, others (Goyal et al., n.d.) suggest that treating learning rate as a function of batch size can minimize the generalization gap and speed-up the training. Nevertheless, using yet another optimization method – the batch normalization (Ioffe & Szegedy, 2015), one can use bigger learning rates and batch sizes, while improving the performance and speeding-up the training (Iglovikov et al., 2017; Ramanath et al., 2019; Volpi & Tuia, 2017).

Hyperparameter optimization is the last step before getting the final results on a test data, that aims to improve the performance of a model and is restricted by time, money and computational power. The configuration of model hyperparameters is generally problem specific and can't be estimated from data, thus is often set arbitrarily by the user before starting the training. The most widely used automatized strategies for finding the best configuration of hyperparameters is grid search and random search. The former evaluates every possible configuration of parameters specified in a grid, which makes it slow, computationally expensive and unable to work with many hyperparameters. On the other hand, the latter replaces the grid with random sampling, which is a more efficient approach able to work with many hyperparameters and explore wider space in less time. However, both share the disadvantage of each guess being independent from the previous ones[12]. This can be better handled with manual search that can better exploit the previous experience but can get expensive and tedious. Lastly, the Bayesian optimization aims to solve all of the above-mentioned problems by predicting the target metrics from hyperparameter configuration.

2.8 Pre-processing

Due to memory limitations of hardware, it is a common practice to tile or down-sample large images before they are fed into the model. However, these methods can cause unpredictable errors in the model's output.

Tiling presents additional hyperparameters (such as tile size and the amount of overlap) and can degrade the classification results, especially in the border regions (Reina et al., 2020). U-Net, for instance, can perform poorly near borders of images because of a bias resulting from padded convolution (Stoian et al., 2019). Therefore, introducing a certain amount of tile overlap can help to

[12] https://www.datacamp.com/community/tutorials/parameter-optimization-machine-learning-models

overcome this issue (Rakhlin et al., 2018). Hung et al. (2014) reported improved classification accuracy with the tile size that could fully capture individual plants from the target class but warned from using too big tiles that could introduce noise from neighbouring plants. Also according to Bao (2019) the receptive field depends on the size of the object of segmentation. He achieved higher segmentation accuracy and better localization by employing two different sizes of receptive fields in his dual-branch FCNN – small for segmentation of small objects and large for segmentation of bigger objects and better localization. Conversely, for mapping the vegetation extent Flood et al. (2019) argue that the tile size should be bigger than the objects and should cover groups of the individuals from the target vegetation class, rather than individuals. The resolution of the imagery is an important factor to be considered when deciding the objective of the task. Because the spatial structure of canopies is related to size of the individual relative to the pixel size (Fricker et al., 2019), clusters of similar pixels will represent individual trees in HR imagery, while they will be able to represent only entire stands of trees with a coarser spatial resolution. Another important thing to look for is the amount of context. Larger tiles perform better because they capture more context of the image and the color and texture features are more consistent (Hung et al., 2014). P. Zhang et al. (2018) in their binary classification used only images containing both classes to enhance the efficiency of the training. The success of vegetation detection lies in the comparison of differences between the textural and structural characteristic of the target class and the surrounding vegetation (Kattenborn et al., 2020). This means that even RGB imagery that has low spectral but high spatial resolution can be very useful in vegetation mapping, which is great news for low-cost UAV datasets. Even better performance can be presumed by combining high spatial resolution sensors with high spectral resolution (multi- or hyperspectral data). P. Zhang et al. (2018) demonstrated that 8-band datasets achieved the highest overall accuracies, decreasing for 4-band, CIR and finally RGB data.

Multiscale saliency of a patch affects the classification performance (Kim et al., 2011; F. Zhang et al., 2015) and could be also considered as a hyperparameter. Down-sampling, reducing the dimensionality of an image, enables faster processing of data or capturing a broader context, which maintains large structural elements but loses some fine detail. Therefore, the suitability of this approach appears to be highly case-specific. While for some tasks filtering out only the most pronounced information can boost the performance (Müllerová et al., 2017; Rakhlin et al., 2018), for others where the information contained within the object is as important as the context it might not be as desirable (Reina et al., 2020; Shaban et al., 2019; W. Zhang et al., 2019). The information loss and object distortion cause by resizing of large tiles can be undesirable also in case of using pre-trained networks (Zheng et al., 2016). However, it seems that the target class is not the only factor that affects rescaling, but also the season (in vegetation classification), pixel resolution, type, depth and some parameters of the network and last but not least the objective of the study and its respective tradeoffs. In a study of a giant hogweed (*Heracleum mantegazzianum*) (Müllerová et al., 2017), a monocarpic perennial herbaceous flowering plant, resampling helped to resolve the problem of the noise springing from very fine spatial resolution of UAV images, that was overwhelming the relevant spatial patterns and consequently hampering the classification accuracy. However, this approach was useful only in autumn and in slightly blurred images. In the summer images the success rate dropped, likely because merging pixels into mixed pixels during

this phenological stage created more confusion in vegetation recognition. Rakhlin et al. (2018) found scale 1:2 as a perfect tradeoff between image resolution, receptive fields and depth of the model and Reina et al. (2020) encountered a convergence of F1 score at this scale, however, at the same time the latter study and others (W. Zhang et al., 2019; Zheng et al., 2016) also got the best results with almost no re-scaling.

Balancing a dataset can be considered as another pre-processing method aiming to improve the results. It means obtaining a more even distribution of classes in the dataset. This can be achieved by under-sampling the majority class or over-sampling the minority class. Other techniques include interpolating minority-class data points or penalizing misclassification of the minority class. While the under-sampling method reduces the number of samples in the abundant class and can lead to a loss of a critical information, over-sampling grows the number of samples in the minority class and can lead to overfitting [13]. Which approach to choose depends on the problem and amount of available data. Wei & Jr (2013) achieved the highest accuracy by using balanced training dataset (50% of the dataset was the target class) with any proportions of the target class in the test data. Others (F. Zhang et al., 2015) simply exclude the patches that don't contain the target class entirely from the dataset.

2.9 Data augmentation and transfer learning

One of the main problems in the remote sensing domain, the lack of labelled data for model training, can be overcome by data augmentation or transfer learning (Scott et al., 2017).

Data augmentation is a technique of artificially increasing the size of a dataset by applying label-preserving random transformations to the original images. The model has access to a bigger volume of labelled data to learn from, potential correlation between patches in the batch are reduced, and the model sees more diverse aspects of the data, allowing it to encode the desired invariance. As a result, more robust feature descriptors (F. Zhang et al., 2015) are created and the generalization ability is improved (Volpi & Tuia, 2017). New images can be generated using many different strategies, such as rotating, flipping, zooming or cropping. However, the choice should be relevant and meaningful with regards to the type of the problem. For example, the original U-Net (Ronneberger et al., 2015) uses elastic deformations on the available biomedical data, because it is the most common variation in tissue and it efficiently simulates realistic deformations. For multi-class labelling tasks, e.g. land cover classification, contextual information at multiple scales is important since features of different land cover types and ground objects usually exist at various scales (P. Zhang et al., 2018). Diakogiannis et al. (2020) apply rotations and zooming in/out (a sort of re-scaling, that was already discussed in the previous sections) on the dataset containing mostly urban land cover classes, Kattenborn et al. (2020) use rotating, shearing (0-0.2 radians), shifting (0–15%) and horizontal flipping in the assessment of plant species and (Li et al., 2018) advocate primarily for shift, rotation and scale variations. Similarly as in case of tiling, data augmentation also can affect predictions in border regions. Reina et al. (2020)

[13] https://medium.com/analytics-vidhya/what-is-balance-and-imbalance-dataset-89e8d7f46bc5

reported that the results of a flipped image differed from the ones in the original image. On the other hand, Long et al. (2015) argued that random mirroring and "jittering" the images by translating them up to 32 px did not have a noticeable impact on the performance. The usefulness and type of data augmentation may, therefore, depend on the problem domain. Augmentation can be applied offline, in a pre-processing stage, or online, so called real-time augmentation. The former is usually used with small datasets and the augmented images become a part of the training set, so the model sees them multiple times. The latter is applied to big datasets and does not include saving the images on disk, the model sees different images at each epoch and therefore generalizes better[14].

Transfer learning is a supervised learning method, that uses weights from a network that was already trained and fine-tuned, usually on a bigger dataset with samples similar to the ones we want to apply it to and offers a promising alternative to feature design. It can be implemented using pre-trained model as is, using it as a feature extractor or fine-tuning it[15]. As compared to a network trained from scratch that starts with randomly initialized weights, employing transfer learning cuts down costs and training time, while achieving high performance even with small sized datasets (Hussain et al., 2019; W. Zhang et al., 2019). Hussain et al. (2019) observed almost double accuracy using the pre-trained model even on image categories different from the ones in the dataset that the model was originally trained on.

Transfer learning has been enabled with the advent of publicly available large datasets, among the most famous ones ImageNet[16], CIFAR[17] or MNIST[18] that gave rise to pre-trained models, such as VGG-16, Inception-v3 and ResNet50, amidst the most popular ones for the image classification task[15]. In the specific case of U-Net, one of these networks is simply used as the encoder. One such example is TernausNet (Iglovikov & Shvets, 2018), that uses VGG11 network pre-trained on ImageNet as an encoder, while in other works (Ulmas & Liiv, 2020) a pre-trained ResNet50 is used. The latter experienced the worst F1 scores of scrub and herbaceous vegetation (0.16), coniferous forest (0.32) and bare rocks (0.41) classes, denoting that these land cover types may be a challenge for pixel level segmentation. High resolution aerial images can bring along big intra-class and small inter-class variances in pixel values, which can cause difficulties in discrimination of some land cover classes (Diakogiannis et al., 2020; Mahdianpari et al., 2018). Transfer learning is also helpful in applications where it is problematic to collect a large volume of training data.

[14] https://towardsdatascience.com/data-augmentation-techniques-in-python-f216ef5eed69
[15] https://towardsdatascience.com/beginners-guide-to-transfer-learning-on-google-colab-92bb97122801
[16] http://www.image-net.org/
[17] https://www.cs.toronto.edu/~kriz/cifar.html
[18] http://yann.lecun.com/exdb/mnist/

3 Materials and methods

This section describes materials and methods used in this thesis. The work consists of four main parts:

- labelling the data,
- creating the dataset,
- training an ML model for an automatic classification and
- developing a method to improve its performance on my data.

First three sub-sections describe the geographical area under study where photos were taken, the data themselves and the initial data visualization with QGIS.

The following sub-section describes the tools and methods used for data labelling and creating the main dataset and its sub-datasets and explains the rationale behind these processes.

The last part details the U-Net and justifies why I used this particular CNN.

All the methods used to improve the performance are described in the section 4.

3.1 Study area

Quinta da França is a 500 ha property located near Covilhã in Castelo Branco District, surrounded by the Zêzere River and Ribeira de Caria stream. The local climate is mild and generally warm, with an average annual temperature of 13.5°C and precipitation around 1082 mm[19]. The coldest month, January, with an average temperature of 6.2°C, is also the wettest (162 mm of rainfall). On the other hand, the warmest months, July and August (average temperature of 21.9°C and 22.2°C, respectively), are the driest ones of the year (10 mm of rainfall) and are therefore critical regarding the risk of forest fires.

As it can be seen in Figure 3, the farm is divided into three main zones:

1. Quinta de Cima: Northwest area with beef production and permanent irrigated grazing pastures. Corn (for silage and grain) and hay fodder are produced here.
2. Quinta de Baixo: South area with another cattle production and all sheep production. In this area are permanent irrigated pastures, rainfed, natural pastures and improved natural pastures.
3. Serra: Northeast area with oak forest.

[19] https://en.climate-data.org/europe/portugal/covilha/covilha-6944/

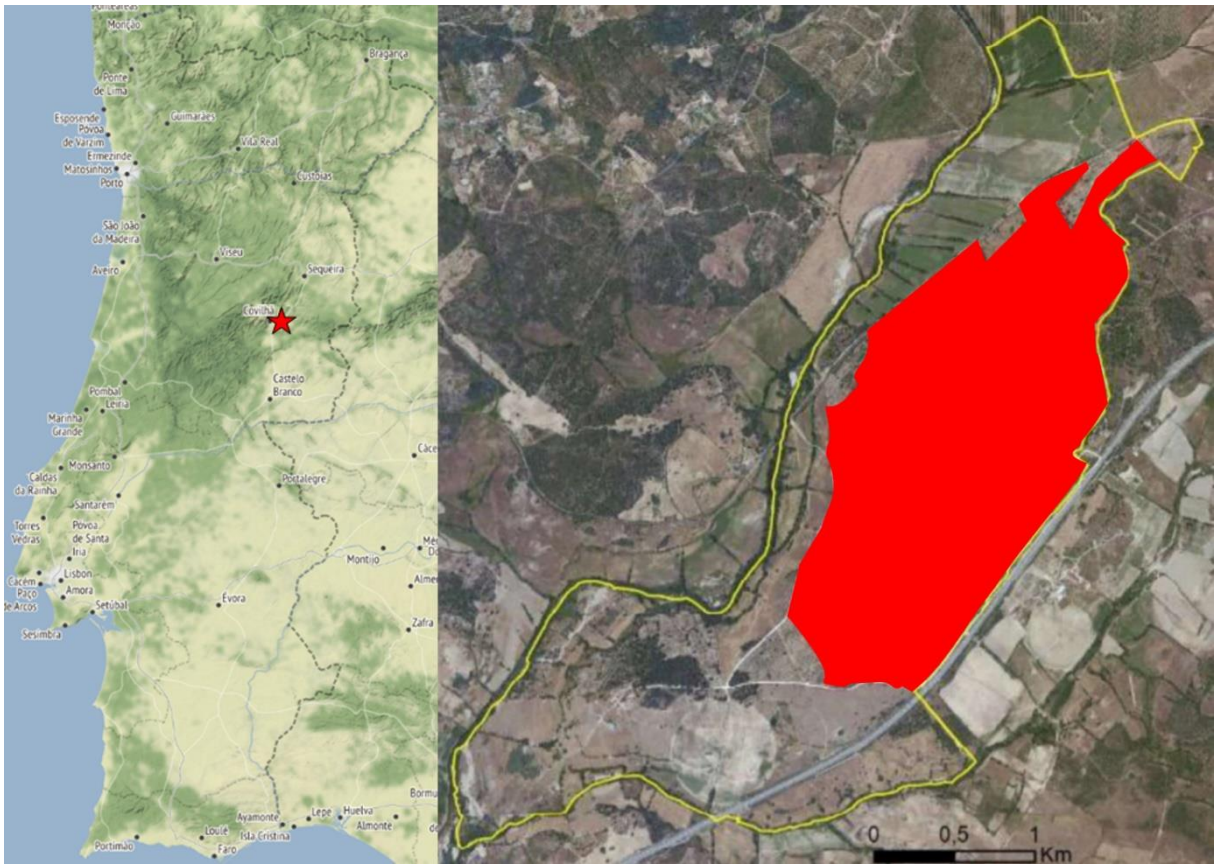


Figure 3 Left: Location of Quinta da França in Portugal (Source: QGIS). Right: The zone of the oak forest Serra (red) (Source: Terraprima -Sociedade Agrícola Lda., 2012)

The farm's sheep and bovine animals graze at Quinta de Cima and Quinta de Baixo (Simões, 2019). The forest in Serra, previously closed for animals, was divided by a fence in January 2018 into two parcels of about 100 ha each: a southern grazing parcel, to test the effect of cattle presence on vegetation structure (grazing, trampling, etc.), and a northern parcel without grazing (wild herbivores are present but in low density)[20]. In June 2018 the grazing parcel was open to a group of about 60 cows. The animals maintain permanent access to the site since then. However, the use of the forest space by cattle tends to increase in late spring-early summer (May-July), possibly related to the simultaneous availability of resting areas, with shadows and green forage. Mechanized removal of shrubs along main tracks (dirt roads) is maintained at both parcels.

The dominant plant species in the forest site is pyrenean oak (*Quercus pyrenaica*), that is naturally occurring in the region. Maritime pine (*Pinus pinaster*) plantation area is also an important component of the forest cover. In the riparian areas, black alder (*Alnus glutinosa*) and ash (*Fraxinus excelsior*) are the key species. In the shrub stratum, besides the dominant brooms species (*Cytisus multiflorus* and *C. scoparius*), there can be also found hawthorn (*Crataegus monogyna*), blackberry (*Rubus ulmifolius)* and grey willow *(Salix atrocinerea)*.

[20] https://www.terraprima.pt/pt/projecto/23

The images were taken in the test area, marked as a red star in Figure 4, which is located within the grazing parcel – the area of interest for the study.

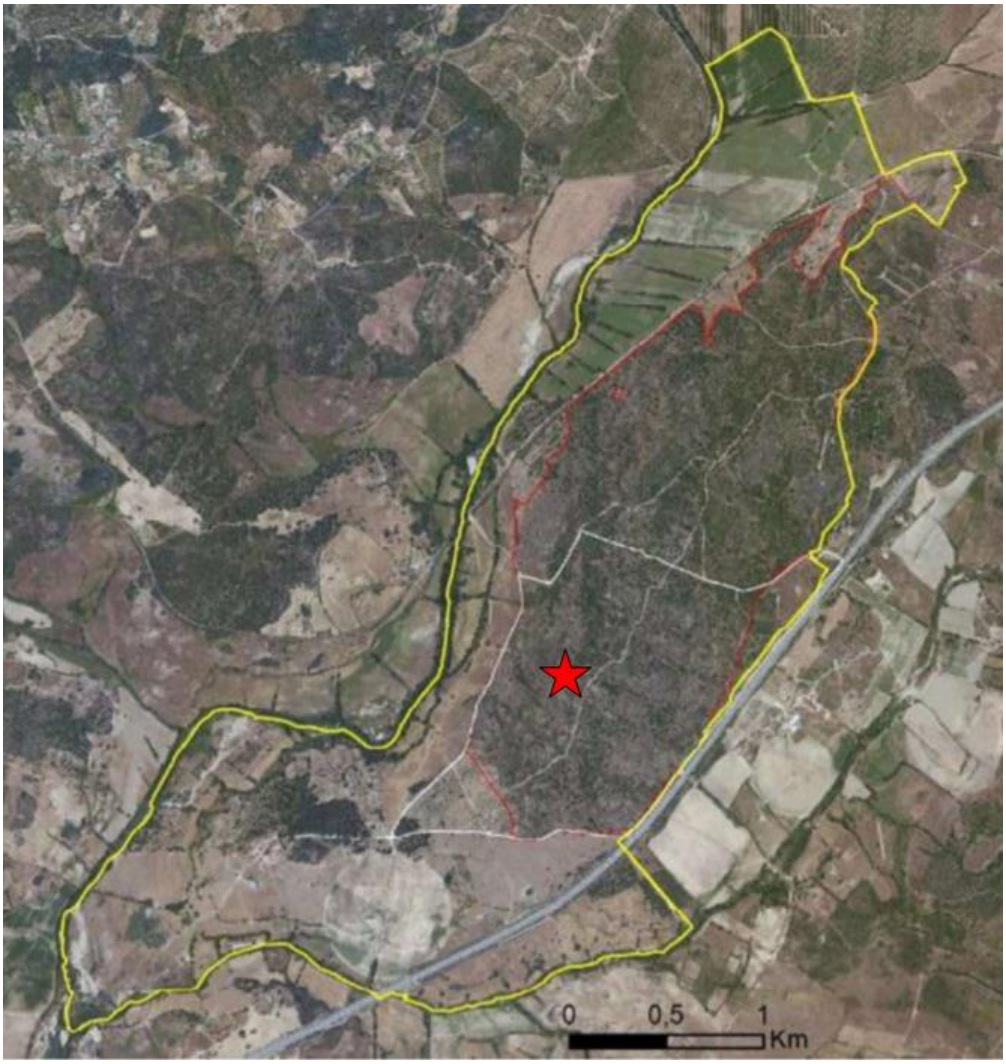


Figure 4 Aerial view of Quinta da França (yellow border), oak forest perimeter (red border) and grazing parcel (white border). The red star depicts the location of the test data used in this thesis (Source: *GO - SILVPAST - Terraprima*, n.d.)

3.2 Data description

The images were acquired by hexacopter with two cameras: VIS GITUP2 camera with RGB filter (370 – 680 nm) and 170° lens (fish-eye) and NIR Mapir Survey2 NDVI camera (Red: 660 nm, NIR: 850nm), with 90° lens. 16MP ((4608 x 3456) px) sensor Sony Exmor IMX206 (Bayer RGB) was used. The flight altitude relative to the take-off point was 120m, velocity 5m/s and photos were taken every 5s. The drone was assembled by Terraprima.

Two test sets of images were provided by Terraprima for this thesis: 1) a set of samples of the orthomosaic for the test area and 2) a set of original images that compose the orthomosaic.

The first set consisted of nine (3361 x 3361) px orthomosaics in TIFF format. The test area was clipped from the orthomosaic of the full forest area, which was composed by images captured in several flights with an approximate scan area of 20 ha per flight. All nine samples covered an area of approximately (200 x 200) m, with spatial resolution 4 cm. The size of the area was defined arbitrarily, large enough to include different land cover types and small enough to be fast to process and to test the methods. The images in the set were:

- R and NIR band and NIR composite image taken in July 2019;
- R and NIR band, NIR composite and RGB image taken in January 2019;
- NIR and RGB image from July 2016.

Example of an image taken from both of the mentioned cameras can be seen in Figure 5 below:

Figure 5 Sample RGB image taken with VIS camera (left) and NIR image taken with NDVI camera (right)

Flights took place in different seasons to exploit the differences due to vegetation phenology. Namely, to take advantage of the sharp distinction between perennial (green) shrubs and the senescent (yellow) herbaceous vegetation in the summer, and to take advantage of the deciduous canopy cover in the winter to facilitate spotting shrubs under the deciduous oak trees. Even though the seasonal features were not expressed enough in the images to improve the visual interpretation of the data for labelling, since the detection accuracy depends on phenological phase of the vegetation that can even partly compensate for lower spectral resolution (Müllerová et al., 2017), some of these features, that were not readily perceived by the human eye, can still facilitate the ML classification and help to achieve better performance. Naturally, high spectral resolution still plays an important role in vegetation mapping, especially for less distinct species. Software Agisoft was used for orthorectification. Due to continuous technical issues with these images, likely resulting from the overlapping discrepancies in the orthomosaic, I abandoned the work with this set and substituted it with the second set.

The second set was composed of 21 (4608 x 3456) px original TIFF images in RGB, which were captured for the same test area during a single flight, that took place in August 2019. Because of the small size of the test area and the overlap between consecutive images, the images were highly similar. The disadvantages of these images are fisheye and motion blur, which causes distortion and makes the

annotation more challenging, especially so in peripheral areas of the images. Later, I obtained two more sets of data for final testing: 49 winter images from December 2019 and 45 summer images from August 2020.

Unlike hyper- or multispectral datasets used in many vegetation cover classification studies (Fricker et al., 2019; Langford et al., 2019; Makantasis et al., 2015; Yue et al., 2015), this thesis uses ordinary RGB images. Limited number of spectral channels makes the presented method more convenient for use in combination with most aerial imaging systems, including off-the-shelf UAVs, and wider range of data.

3.3 QGIS data visualization

In the initial phases of the work I used Semi-Automatic Classification Plugin (SCP), a free open source QGIS plugin for supervised land cover classification of remote sensing images. The objective was to fast and easy visualize and benchmark the classification performance of a working land cover classification tool on my data, and to spot classes that may have a higher risk of being misclassified.

A broad spectrum of image raster management is available in QGIS; application of colour ramps, raster calculator, band manipulation and control of various features such as brightness and contrast. All these have a potential to ease image interpretation. The SCP was a number one choice for classification exercise, because it offers several tools for image pre- and post-processing and it was developed for the purposes highly aligned with the ones of this thesis.

The experiment was conducted on a small scale using five images from the first dataset. I tested RED and NIR bands from the NIR images from January and July 2019 and RGB from July 2016 for visualization purposes. In total, I created seven classification projects, inspecting different settings with two main purposes – clearer data visualization for improved image interpretation and better classification results. Four main classes were identified: trees, shrubs, shadows and ground. The ground class is an aggregate of bare soil and herbaceous cover, that is mostly senescent in the late summer. Figure A 1 and Figure A 2 in the Appendix are examples of using different channel settings for easier recognition of certain classes. Table A 1 in the Appendix is a summary of tested tools with the most significant impact and the corresponding conclusions.

Despite achieving more recognizable vegetation and bare soil visualization, I did not manage to produce images with clear distinction between shrubs and trees. In general, the objects from different classes are easy to confuse and exchange, especially pixels in border parts. The target group – shrubs, due to its structure, size and distribution often exhibits features similar to other classes and the information contained in the within-class pixels can differ widely, i.e. there are small inter-class and high intra-class diversities, which causes a high level of confusion. The main problems with identifying objects from individual classes during labelling are summarized in Table 1.

Table 1 The main problems with identifying objects from individual classes during labelling

		Classification				
		Trees	**Shrubs**	**Shadows**	**Rocks**	**Ground**
Reference (visual classification)	**Trees**		Darker border parts Possible misclassifi-cation of small trees for shrubs	Border parts Shadowed parts of trees	-	-
	Shrubs	Lighter border parts Possible misclassifi-cation of big shrubs for trees		General difficulty to distinguish dark shrubs from shadows	Border parts of shrubs growing around rocks	Border parts Possible misclassifi-cation of small shrubs for dense grass
	Shadows	Border parts Shadowed parts of trees	General difficulty to distinguish shadows from dark shrubs		-	-
	Rocks	-	Border parts of shrubs growing around rocks	-		Border parts Darker rocks Small rocks difficult to spot
	Ground	-	Border parts Possible misclassifi-cation of dense grass for small shrubs	-	Border parts	

Misinterpreted pixels in training input corrupted the classification results. Photointerpretation errors of sample pixels collected randomly for reference raster had further negative impact on accuracy statistics. Another big issue in a specific trial was that pixels belonging to rocks' class, due to the small size and scarcity of this class in study images, were entirely absent in generated sample pixels, which resulted in 0 pixels classified as rocks in accuracy statistics. Table 2 reveals, that the algorithm had problems distinguishing trees from bare soil. From Figure 6 is apparent, that the most problematic are border pixels on the irradiated side, as well as some shadowed parts of the trees, that may resemble darker areas of the bare soil class. That the classification accuracies depend on whether the area is sunlit or shaded was demonstrated by Lopatin et al. (2019), who showed that even when shadows were included during model calibration, the predictions in shaded areas of canopies were generally inaccurate and lead to misclassification rates between 65% and 100%. The overall accuracy was 47.90% and Kappa coefficient 0.34. Full Area Based Error Matrix and visualized classification results are displayed in Table 2 and Figure 6.

Table 2 Area Based Error Matrix (Source: QGIS)

	Reference: the estimated area proportion of each class						
Classified: the estimated area proportion of each class	**Trees [%]**	**Shrubs [%]**	**Shadows [%]**	**Rocks [%]**	**Bare soil [%]**	**Classification raster estimated area [m^2]**	**Total [%]**
Trees [%]	12.23	0.00	0.00	0.00	1.53	5501.62	13.75
Shrubs [%]	0.93	7.46	0.00	0.00	4.66	5222.35	13.06
Shadows [%]	4.75	1.58	11.09	0.00	0.00	6968.71	17.42
Rocks [%]	2.91	4.36	5.81	0.00	5.81	7555.49	18.89
Bare soil [%]	14.49	5.27	0.00	0.00	17.12	14750.63	36.88
Total [%]	35.30	18.67	16.90	0.00	29.13	39998.79	
Reference raster estimated area [m^2]	14121.00	7469.00	6759.00	0.00	11650.00	39998.00	
Producer's accuracy (PA)[21] [%]	34.63	39.96	65.61	nan	58.79		
User's accuracy (UA)[22] [%]	88.89	57.14	63.64	0.00	46.43		

[21] The producer's accuracy for each class is calculated as the ratio of correctly classified samples and the column total.
[22] The user's accuracy for each class is calculated as the ratio of correctly classified samples and the row total.

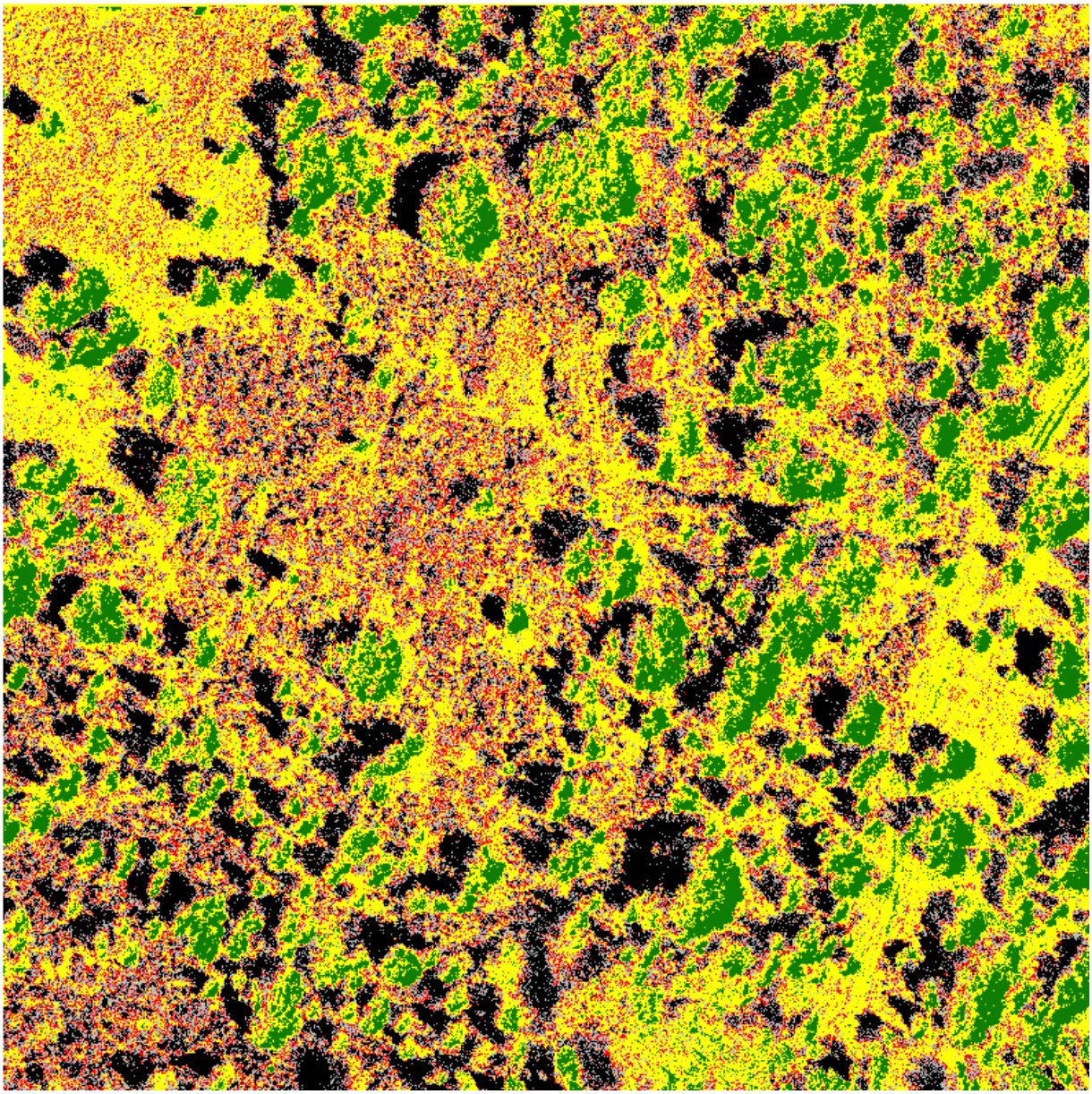


Figure 6 Visualized classification results. Pixel colour codes: Green – Trees, Red – Shrubs, Black – Shadows, Gray – Rocks, Yellow – Bare soil (Visualized in QGIS)

3.4 Machine learning model

I chose a cloud service Google Colab for training and evaluating the model. The main reason was the free access to computing resource. The GPUs available in Colab often include Nvidia K80s, T4s, P4s and P100s[23]. Colab comes with other advantages too, such as ready to use environment with pre-installed important packages, easy use with Google Drive, or importing datasets from Kaggle[24]. The

[23] https://research.google.com/colaboratory/faq.html
[24] https://www.kaggle.com/

deep learning methods were implemented using Keras[25] (version 2.4.3) with TensorFlow[26] backend (version 2.3.0). Free version of the service comes at a cost of memory limit of 12GB and time limit of 12h. This can become a challenge for some experiments, however this thesis also aims to explore the set ups with a reasonable tradeoff between working within these limits and still yielding good results. This increases the usability and practicality for future students with limited access to advanced virtual machines that would like to build upon or further extend this thesis.

As a basis for the work I used a U-Net model[27] created for a Kaggle competition hosted by geoscience data company TGS[28]. The model (hereinafter the TGS U-Net) was originally developed to segment regions containing subsurface salt deposits in seismic images. The training dataset, provided by TGS, consisted of 4000 seismic grayscale images with dimensions (101 x 101) px and their corresponding masks. The TGS U-Net architecture, as in the original U-Net, extracts features with convolutional layers in the encoding part and restores the original size of the image in the decoding part. However, unlike the original U-Net, TGS U-Net uses the input image with size of (128 x 128 x 3). The size is gradually reduced, while the depth is increased (from (128 x 128 x 3) to (8 x 8 x 256)), and then the size is gradually increased, while the depth is decreased (from (8 x 8 x 256) to (128 x 128 x 1)).

The main building block of the TGS U-Net consists of two consecutive 2D convolutional layers with batch normalization and ReLU activation function. Batch normalization was stated by the author[29] to significantly improve the training. The number of filters starts at 16 and is doubled at every convolution step. There are four such blocks in the encoder side, each followed by max pooling layer, that halves the image dimensions, and a dropout layer. The fifth convolutional block forms a bottleneck with the maximum depth and minimum spatial dimensions (Table 9) after which comes the decoder side, with four symmetrical deconvolution layers (i.e. transposed convolutions) concatenated with the feature maps from the encoder side. After comes a dropout layer and the convolutional block, which helps the model to assemble a more precise output. The number of filters is halved at each step, while the resolution is doubled. Ultimately, the output of a binary classification is sigmoid, which assigns each pixel a probability of belonging to the target class. The model is trained with Adam optimizer with a learning rate of 1e-5. Predictions are compared to labels with binary cross entropy loss function. TGS U-Net also uses accuracy[30] to evaluate the performance. However, this metric has a major disadvantage – it doesn't deal well with class imbalance. (For this reason I also included precision and recall and I picked their harmonic mean, the F1 score, as the main indicator for the classification evaluation, since it is a more appropriate measure of accuracy for datasets where one class overpowers another.) Keras callbacks are used to save the weights if the validation loss improves, and early stopping is implemented if the validation loss doesn't improve for 10 consecutive epochs to prevent overfitting. Learning rate is

[25] https://keras.io/
[26] https://www.tensorflow.org
[27] https://github.com/hlamba28/UNET-TGS
[28] https://www.kaggle.com/c/tgs-salt-identification-challenge
[29] https://towardsdatascience.com/understanding-semantic-segmentation-with-unet-6be4f42d4b47
[30] The overall accuracy = (TP + TN) / (total number of individuals tested)

reduced when the validation loss doesn't improve for five consecutive epochs. For each pixel the probability of belonging to the target class (salt) is calculated, with the threshold of 0.5.

The dataset is split into training and validation set with ratio 9:1. The validation set is never used in the training process, it is only used to evaluate the model's performance. There are 50 epochs with batch size of 32.

The model was trained on P4000 GPU, took less than 20 mins to train and achieved accuracy of 0.92. It was slightly overfitting, likely due to small number of training images.

The detailed architecture of the TGS U-Net can be seen in Figure 7.

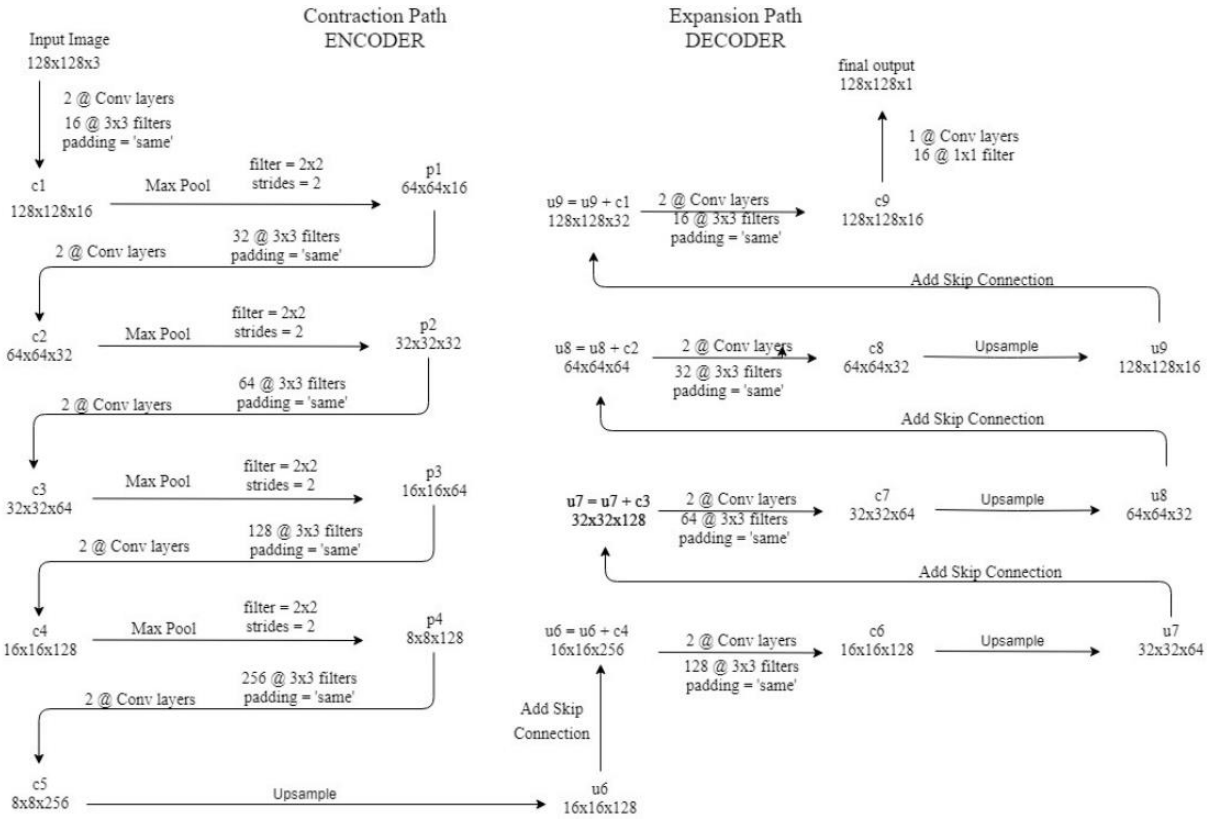


Figure 7 Detailed architecture of the used model. 2@Conv layers – two consecutive Convolution Layers; c1-c9 – the output tensors of Convolutional Layers; p1-p4 – the output tensors of Max Pooling Layers, u6-u9 – the output tensors of up-sampling (transposed convolutional) layers (Source: https://towardsdatascience.com/understanding-semantic-segmentation-with-unet-6be4f42d4b47)

3.5 Description of datasets

The following subsections describe the development of training data; from the processing of raw images and labelling procedure to tiling annotated images along with their corresponding binary masks into final sub-datasets used in the experiments.

3.5.1 The development of the main dataset

This section describes the process of creating and labelling the dataset. Due to input requirements of the annotation tool used in this thesis, all 21 RGB images in TIFF format had to be first converted into PNG. After that, images were sliced into smaller square-shaped tiles with dimensions (800 x 800) px, as depicted in Figure 8, which corresponds to approximately (50 x 50) m patches of land (in case of sections of image perpendicular to the drone). The tile size was chosen based on the size of the objects of interest and the amount of context, with the main objective to make visual recognition for labelling easier. Adjacent tiles have overlap of 39 pixels in horizontal and 136 pixels in vertical direction. The code can be found at https://github.com/aggiungi1procione/Thesis---Shrub-detection-with-U-Net in the file *tif_to_png.py*. In total, 630 tiles were generated from the original images (30 tiles per image). Out of these, only 13 were selected for labelling, due to the time-consuming nature of this process. All 13 tiles came from the same image. The selected tiles, as well as the complete list of names of the images and the tiles produced from them can be found in Table A 2 in the Appendix. During the selection, I aimed for a sample of tiles that would be representative for every part of the original image with different land-cover configuration and that would contain all 4 classes (shrubs, trees, shadows and rocks) and the classes would be represented approximately evenly. This, of course, was not achievable in case of rocks, that were very scarce in the images.

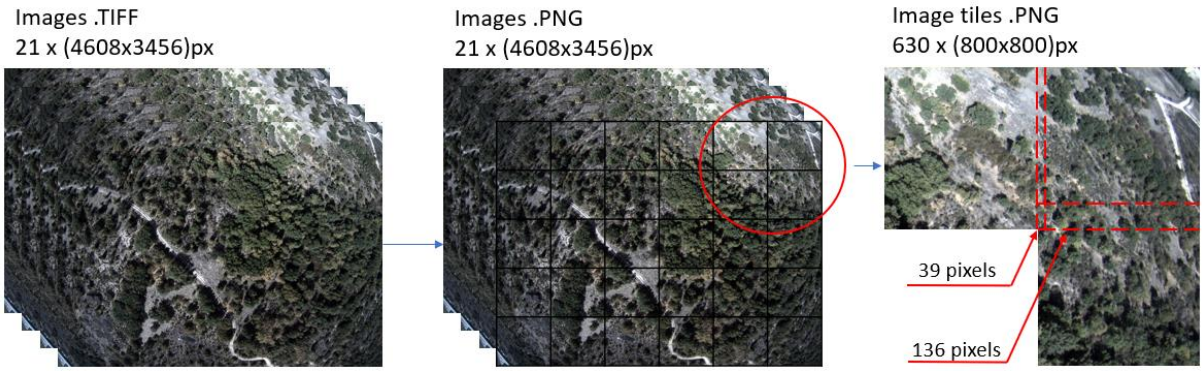


Figure 8 Flowchart of slicing original images into tiles and an illustration of tile overlap

For annotation purposes I chose Labelbox[31] – a professional platform for labelling and managing the training data. There were several reasons for choosing this platform. First of all, tidy interface and an

[31] https://labelbox.com/

Overview menu helped to better track my many experimental projects and datasets and to keep them well organized. Furthermore, easy adding or deleting assets made manipulation with datasets very flexible. Another reason was the choice of segmentation tools, where in addition to traditional *Pen* tool it was possible to use *Superpixel* tool, which calculates segment clusters of pixels based on their color and a segment cluster size given by the user. This tool was more efficient in annotating objects with complex boundaries, such as vegetation. In general, labeling, editing and erasing of the labels was user-friendly. Another advantage was a built-in function to temporarily manipulate brightness and saturation levels of images that was a fast and useful way for better understanding and interpreting of the land cover. It was also possible to control the opacity level of created labels, which was very much appreciated during re-assessment and validation process with supervisors. Finally, the platform is well-documented and backed by a professional support. A big handicap of the platform was occasional failure to save labeled objects. This was especially dangerous because it was only possible to see during revision. Multiple revisions and corrections of the labeled dataset were necessary.

Land cover classification requires fine-grained understating of an image and its context, meaning that dense pixel-level annotation like semantic or instance segmentation is needed. While the former labels each pixel with a corresponding class, the latter also classifies each instance of a class separately. For the purposes of this thesis semantic segmentation is sufficient. Table 3 shows pixel share of the four classes in the dataset. As mentioned earlier, I exploited segmentations approximated by superpixels to facilitate the annotation process, rather than selecting individual pixels. The final product of the process was a set of 13 hand-crafted dense pixel-level semantic segmentation maps, where each pixel was assigned a label of a corresponding class (Figure 9). Pixel-based classification maps capture well the geometry of an image, such as corners and fine elements, but can face issues like noise or an incorrect characterization of context dependent classes (Stoian et al., 2019)

Table 3 Pixel share of classes in the dataset

Class	Shrubs	Trees	Shadows	Rocks
Pixel count	1 746 204	4 042 008	1 213 720	90 313
Pixel share	20.99%	48.58%	14.59%	1.09%

The central issue of labelling that could significantly impact the classification results is faulty labelling. This could happen in three ways, namely: (1) Incorrect interpretation of vegetation types. Sometimes problematic distinguishing between the classes made labelling more challenging. Because the flights were not taken at noon one factor helping to differentiate between trees and shrubs was the size of a shadow. In general, bigger shadows could be ascribed to trees, smaller to shrubs. This was, however, not a universal tool because the area contains also young trees that are smaller and thus cast smaller shadow that can be easily swapped with the shadow of bigger shrub species. (2) Incorrect interpretation of border regions. As well as the classification plugin described in 3.3 struggled to differentiate between classes in these regions, the *Superpixel* tool used for annotating, that was described earlier in this section, also often struggled to correctly adhere to boundaries of complex vegetation thus manual selection of pixels was unavoidable. This could lead to assigning an incorrect class to border pixels,

since the visual interpretation was also very difficult. (3) Incoherent class labelling. This concerns mainly shadows. Because this class also coexists with other classes, e.g. shadows within trees, it was difficult to keep consistency while labelling and it could be the case that similar groups of pixels were once labelled as trees and once as shadows. Because pixel-based species classification at high spatial resolution is highly affected by within-canopy variation caused by shadows, Lopatin et al. (2019) decided to completely exclude shadows and only classify sunlit areas, which improved the general performance. I decided not to do this, because I found it generally tricky to draw a line between what is a shadow within trees and what is not anymore, and also because shadows are a part of the canopy structure and will be present in most of the datasets, therefore I found it reasonable to include them in the training to achieve more comprehensive and robust model.

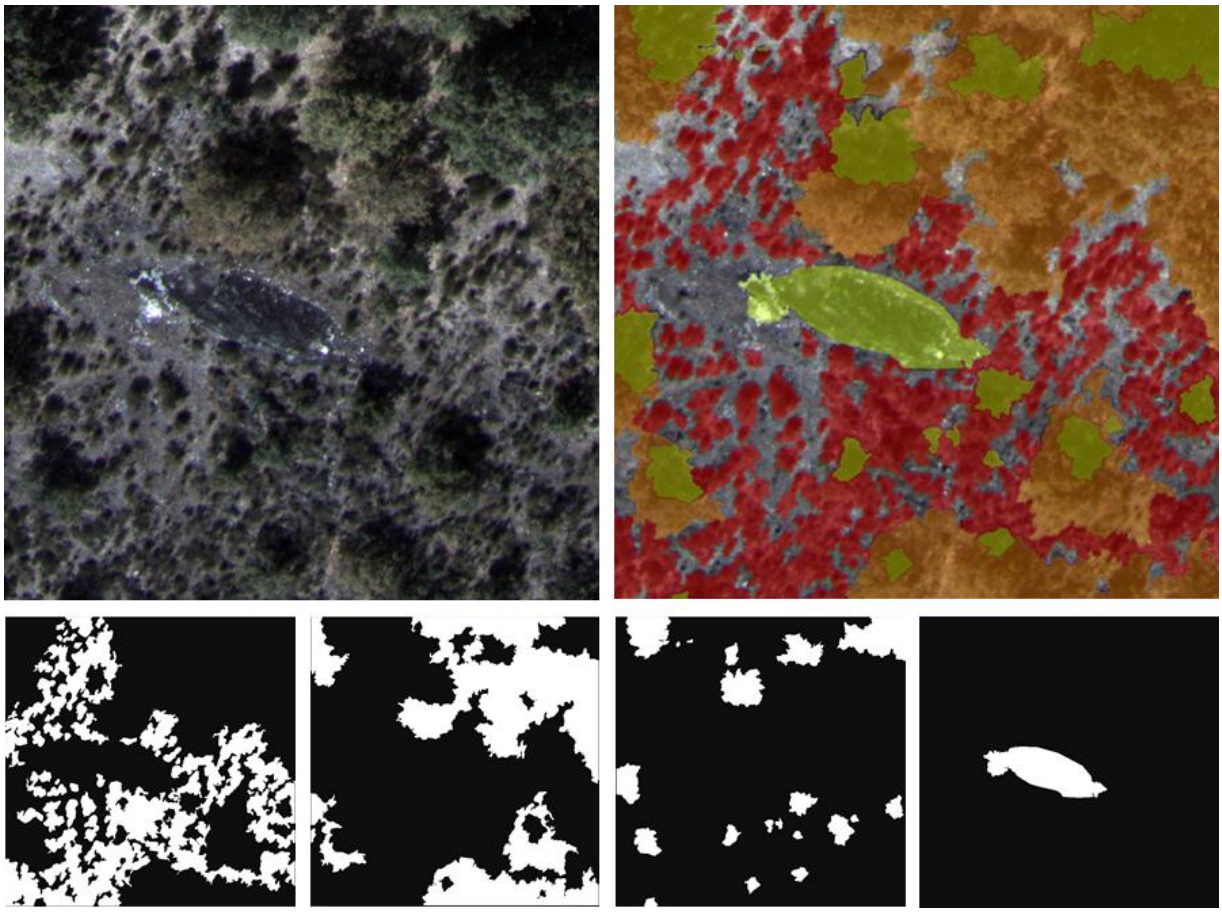


Figure 9 An example of a labelled tile and its binary masks. Up-left: original image tile, up-right: labeled image tile (red - shrubs, orange - trees, yellow - shadows, light yellow - rocks). Bottom (from left): binary mask of shrubs, trees, shadows and rocks

When the tiles were labelled, I exported the masks in JSON file. Since only JSON or CSV formats were available for the export, I used a code at https://github.com/aggiungi1procione/Thesis---Shrub-detection-with-U-Net in the file *masks_download_from_JSON.py* to filter URIs of individual binary masks in PNG format from the export file and to download them into separate class folders. The complete dataset directory structure was then:

```
├── test1
│   ├── test1_tiles
│   ├── test1_mask_shrubs
│   ├── test1_mask_trees
│   ├── test1_mask_shadows
│   └── test1_mask_rocks
└──
```

3.5.2 The development of sub-datasets

This section explains the development of the final training data for experiments and summarizes the process in Figure 10.

To observe the influence of different amount of captured context, five sub-datasets of different sized patches were tiled from the main dataset. The same tiling algorithm as in 3.5.1 was used and can be found at https://github.com/aggiungi1procione/Thesis---Shrub-detection-with-U-Net in the file *masks_download_from_JSON.py.* The development of the main dataset was used, yielding datasets with the following features:

A. 832 patches (64 per tile) with dimensions (100 x 100) px. Overlap of 0 px in both dimensions. These patch dimensions were selected to replicate dimensions of the original data that were provided for the TGS challenge. This sub-dataset was intended to serve as a baseline for the other experiments.
B. 208 patches (16 per tile) with dimensions (200 x 200) px. Overlap of 0 px in both dimensions.
C. 117 patches (9 per tile) with dimensions (300 x 300) px. Overlap of 50 px in both dimensions.
D. 52 patches (4 per tile) with dimensions (400 x 400) px. Overlap of 0 px in both dimensions.
E. 52 patches (4 per tile) with dimensions (500 x 500) px. Overlap of 200 px in both dimensions.

No further tiling was done, since the patch overlap was becoming too big, generating highly similar patches and yielded sub-datasets were too small. Also, the F1 score was expected to converge with the patch size of approximately half of the original tile, maximum though with the 70% size of the original tile (Reina et al., 2020). Unlike some approaches that mind the class-cover proportions of the patch (Buscombe & Ritchie, 2018; Kattenborn et al., 2020; Langford et al., 2019; Watanabe et al., 2018), all of my patches were used as they were tiled and there was no further selection. Using U-Net, the spatial information and correlations among classes matter for the learning process. Classes are spatially unevenly distributed and their frequency varies across patches in my sub-datasets. This stochasticity can positively contribute to the robustness of the model.

To see the impact of rising the number of samples in the datasets on the learning process, data augmentation was applied to all sub-datasets, generating three sets per each, with the size of around 800 samples (consistent with the baseline sub-dataset A), 1600 samples (double the baseline sub-dataset A) and 3800 (imitating the dataset from the TGS challenge, that consisted of 4000 samples). I

used random rotations (probability=0.5, max left and right rotation of 15°, after which the images were not rendering correctly anymore), skews (probability = 0.7, magnitude = 0.5), flips through both y and x axis (probability = 0.9) and also random brightness (probability=0.4, min_factor=0.8, max_factor=1.2) for the augmentation, because they reproduce effects that could be naturally present in the remote sensing imagery. To further increase the diversity of the data, I also used small randomized elastic distortions (probability = 0.5, grid_width = 4, grid_height = 4, magnitude = 5) and shears (probability = 0.4, tilt along y and x axis up to 10°). All the data augmentation was done with an image augmentation library Augmentor[32].

The sub-datasets were subsequently fed to the network using different model input dimensions and thus rescaling the patches in various scales. This was to investigate whether down-scaling could improve the performance by better filtering the relevant spatial patterns or it would hamper it by leading to a too big loss of information. The effect of this strategy on training time was also of interest.

All of the experiments will be further explained in section 4.

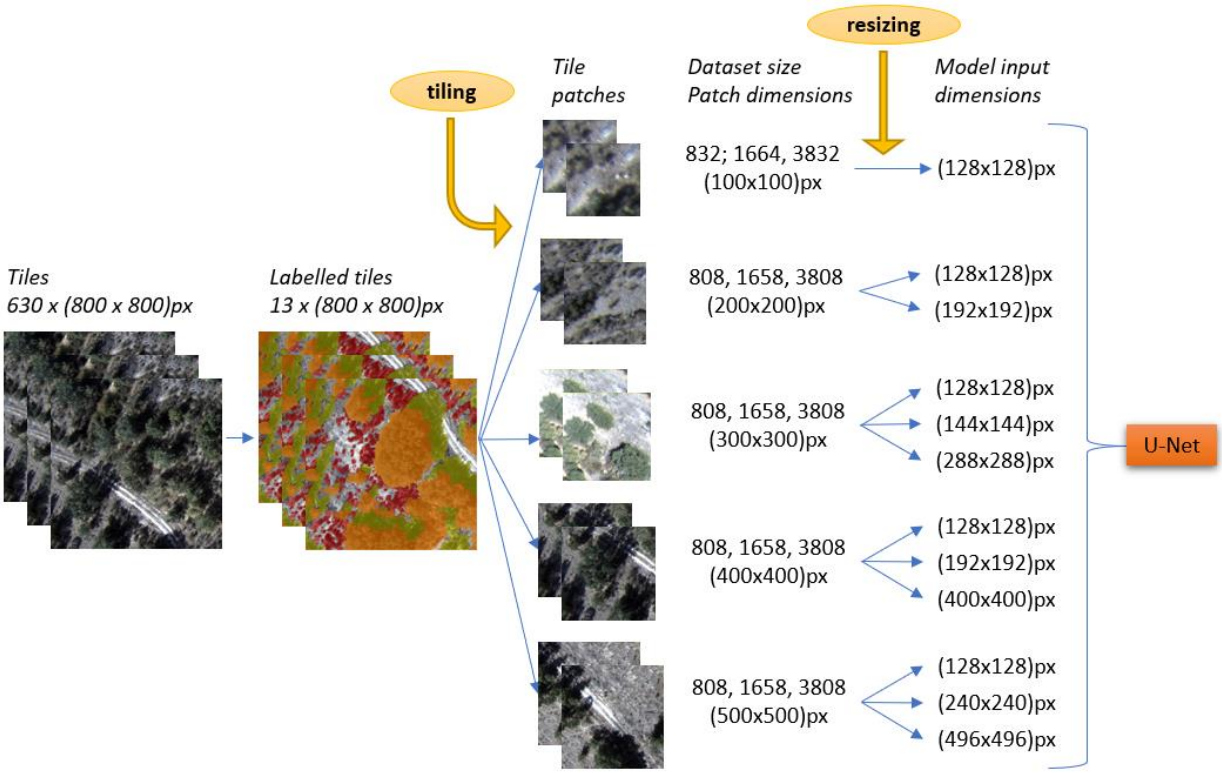


Figure 10 Flowchart of the development of the final sub-datasets for experiments

[32] Marcus D Bloice, Peter M Roth, Andreas Holzinger, Biomedical image augmentation using Augmentor, Bioinformatics, https://doi.org/10.1093/bioinformatics/btz259

3.5.3 The development of test sets

Test sets were created in a similar fashion as was described in 3.5.1 and 3.5.2. I picked a (4608 x 3456) px original TIFF image, converted it to RGB, sliced into smaller square-shaped tiles with dimensions (800 x 800) px and picked two (or one) for labelling. Labelled tiles were then sliced further according to the experiment needs.

I created four different test sets:

1. Using one (800 x 800) px tile from the same image from which training tiles were taken (Table A 2 in the Appendix)
2. Using two (800 x 800) px tiles from other images that were taken during the same flight, as the previous image (Table A 2 in the Appendix)
3. Using two (800 x 800) px tiles from one image from the new summer set (August 2020)
4. Using two (800 x 800) px tiles from one image from the new winter set (December 2019)

The reasoning of the selection was to see the performance on highly similar data (1 and 2), on seasonally similar data (3) and on highly distinct data, taken during different phenological stage (4). Two tiles represent around 15% of the dataset. I picked only one tile for test set 1, because the other remaining tiles either contained mostly trees or were very difficult to interpret (significant curvatures and blur in the border image regions). All selected tiles and their location in the original images can be found in the Appendix Figure A 3, Figure A 4, Figure A 5, Figure A 6 and Figure A 7.

All the datasets can be found at https://www.kaggle.com/biankatn/thesis-shrub-detection-with-unet.

4 Experiments

This part describes the methods and experiments performed in order to improve the detection results. I briefly explain their purpose, set up, procedures and the data used. The main performance measure was F1 score because of the unbalanced nature of the used dataset.

4.1 The baseline: sub-dataset A

I set a baseline by running the original TGS U-Net model, described in the section 3.4, on my data in a set of four experiments – one for each of the classes described in Table 4. I adjusted the code so that the model accepted 3-chanelled RGB input. The code can be found at https://github.com/aggiungi1procione/Thesis---Shrub-detection-with-U-Net in the file *model.ipynb.* Sub-dataset A and its augmented versions (3.5.2 The development of sub-datasets) were used for this purpose. With the train : validation split 9 : 1, the first (non-augmented) sub-dataset A consisted of 748 training and 84 validation patches. The two augmented versions were used only with the shrubs class. The summary of all performed experiments can be found it Table 4. The total number of parameters in the model was 1.18 million.

Table 4 The summary of experiments for the base model with sub-dataset A, (100 x 100) px patches

No.	Class	Dataset size (train-set : val-set)	Patch dimensions (height x width) [px]	Model input dimensions (height x width) [px]
1	Shrubs	832 (748:84)	100 x 100	128 x 128
2	Trees			
3	Shadows			
4	Rocks			
5	Shrubs (augmented)	1664 (1497:167)		
6	Shrubs (augmented)	3832 (3448:384)		

4.2 Sub-datasets B-E: patch size, scale and data augmentation

This is a set of experiments exploring the impact of the patch size and rescaling of the model input on the performance. Data augmentation is also being assessed simultaneously. All these experiments are visually summarized in Figure 10. Because of memory constraints and, most of all, time constraints, not all of them were feasible to do. For these reasons I omitted rescaling experiments with the biggest sub-datasets (3808 instances), even though they exhibited the best performance. I also left out rescaling experiments with the worst performing (the smallest) sub-datasets (808 instances). In the end, I ran 21 experiments in total, with sub-datasets B-E and only with the shrubs class. These experiments are summarized in Table 5 – Table 8. Table 9 summarizes how the size of input images with different initial spatial dimensions changes as it travels through U-Net's pipeline. The total number of parameters was 1.18 million for all experiments.

The assumptions were the following:

1) The patch size:
   a) Building on the studies of Kattenborn et al. (2020) and Reina et al. (2020), the accuracy is expected to improve with increasing patch size, because bigger patch captures more spatial context. This is illustrated in Figure 11.
2) The model input size:
   a) Resizing images to smaller resolutions may lead to a loss of information (W. Zhang et al., 2019). Reina et al. (2020) indeed achieved a better performance with minimal down-scaling,
   b) whereas according to (Müllerová et al., 2017) and (Rakhlin et al., 2018), down-scaling the input patch can benefit the results by better filtering the relevant spatial patterns. This can, therefore, depend on the content of the images and what is the target group. My goal was to figure out which approach would work for my data.

I tested scales 1:1 (patch size close to the original tile size), according to (Reina et al., 2020), and 1:2 according to (Rakhlin et al., 2018). Because the input has to be compatible with the 4 max-pooling layers contained in the architecture of the TGS U-Net, and therefore must be divisible by 2^4, the scales were not always exactly that.

For demonstration purposes, I also ran some of the more time and memory challenging experiments. Adjustments to the code were necessary because the sub-datasets were too large to fit in memory provided by the Google Colab. Namely, a custom generator[33], loading the dataset from the hard disk into memory in batches, was implemented. The code can be found at https://github.com/aggiungi1procione/Thesis---Shrub-detection-with-U-Net in the file *model.ipynb.* However, the purpose was only to validate the hypotheses presented in this section and these experiments, regardless of their performance, were not considered in the further experimenting because of their time-consuming nature, that made them impractical.

[33] https://medium.com/@mrgarg.rajat/training-on-large-datasets-that-dont-fit-in-memory-in-keras-60a974785d71

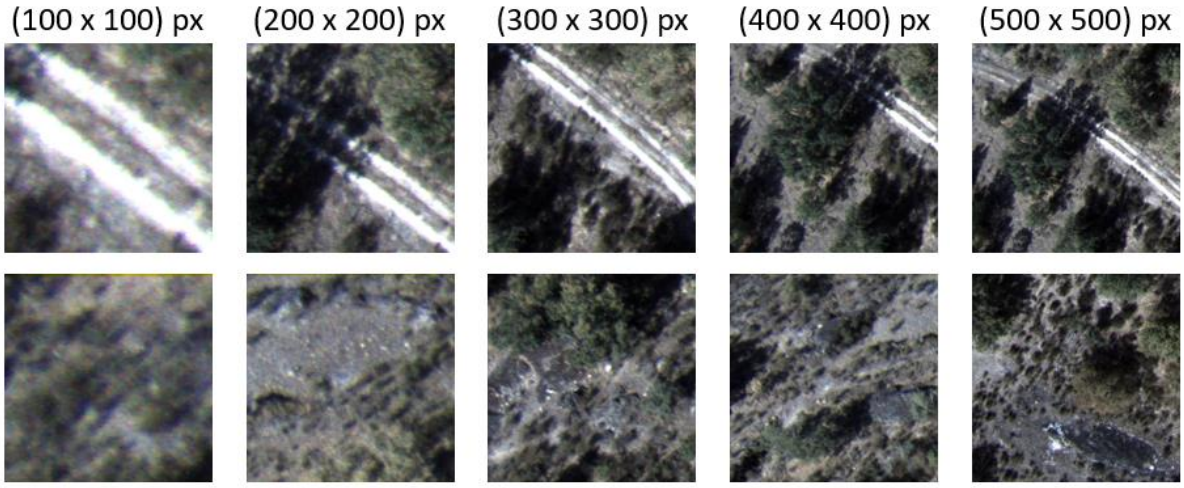


Figure 11 Examples of patches with different sizes (from left: sub-dataset A, sub-dataset B, sub-dataset C, sub-dataset D, sub-dataset E)

Table 5 The summary of experiments with the sub-dataset B, (200 x 200) px patches

No.	Class	Dataset size (train-set : val-set)	Patch dimensions (height x width) [px]	Model input dimensions (height x width) [px]
1	Shrubs (augmented)	808 (727: 81)	200 x 200	128 x 128
2	Shrubs (augmented)	1658 (1492:166)		128 x 128
3	Shrubs (augmented)	1658 (1492:166)		192 x 192
4	Shrubs (augmented)	3808 (3427:381)		128 x 128
5	Shrubs (augmented)	3808 (3427:381)		192 x 192

Table 6 The summary of experiments with the sub-dataset C, (300 x 300) px patches

No.	Class	Dataset size (train-set : val-set)	Patch dimensions (height x width) [px]	Model input dimensions (height x width) [px]
1	Shrubs (augmented)	808 (727: 81)	300 x 300	128 x 128
2	Shrubs (augmented)	1664 (1497:167)		128 x 128
3	Shrubs (augmented)	1664 (1497:167)		144 x 144
4	Shrubs (augmented)	3808 (3427:381)		128 x 128
5	Shrubs (augmented)	3808 (3427:381)		144 x 144
6	Shrubs (augmented)	3808 (3427:381)		288 x 288

Table 7 The summary of experiments with the sub-dataset D, (400 x 400) px patches

No.	Class	Dataset size (train-set : val-set)	Patch dimensions (height x width) [px]	Model input dimensions (height x width) [px]
1	Shrubs (augmented)	808 (727: 81)	400 x 400	128 x 128
2	Shrubs (augmented)	1658 (1492:166)		128 x 128
3	Shrubs (augmented)	1658 (1492:166)		192 x 192
4	Shrubs (augmented)	1658 (1492:166)		400 x 400
5	Shrubs (augmented)	3808 (3427:381)		128 x 128

Table 8 The summary of experiments with the sub-dataset E, (500 x 500) px patches

No.	Class	Dataset size (train-set : val-set)	Patch dimensions (height x width) [px]	Model input dimensions (height x width) [px]
1	Shrubs (augmented)	808 (727: 81)	500 x 500	128 x 128
2	Shrubs (augmented)	1652 (1486:166)		128 x 128
3	Shrubs (augmented)	1652 (1486:166)		240 x 240
4	Shrubs (augmented)	1652 (1486:166)		496 x 496
5	Shrubs (augmented)	3808 (3427:381)		128 x 128

Table 9 Summary of changes in size inside U-Net depending on the model input size

Model input size	Size at the bottleneck (min. spatial size, max. depth)	Model output size
128 x 128 x 3	8 x 8 x 256	128 x 128 x 1
144 x 144 x 3	9 x 9 x 256	144 x 144 x 1
192 x 192 x 3	12 x 12 x 256	192 x 192 x 1
240 x 240 x 3	15 x 15 x 256	240 x 240 x 1
288 x 288 x 3	18 x 18 x 256	288 x 288 x 1
400 x 400 x 3	25 x 25 x 256	400 x 400 x 1
496 x 496 x 3	31 x 31 x 256	496 x 496 x 1

Different models will be from now on denoted with the following name structure: Sub-dataset letter-dataset size_model input dimensions (further info where it applies) (e.g. A-832_128x128 (dropout=0.05)).

Because I had limited time and computational resources, I picked the model that demonstrated the best tradeoff between performance and training time for my situation for the following experiments, which was the case no.3 in Table 6 (C-1664_144x144).

4.3 Balancing the datasets

The purpose of experiments presented in this section is to see the impact of creating training datasets with different target class representations on the model's performance. I analyzed the non-augmented sub-dataset C, that contains 117 patches, for the percentage of shrub pixels and subsequently I filtered out patches containing less than 1% (P. Zhang et al., 2018) and less than 45% (Wei & Jr, 2013) of shrub pixels. I then enlarged these sub-dataset C versions to 1664 instances via data augmentation and I compared the results with the results of an equally sized unfiltered sub-dataset C. The notation of these models will be C-1664_144x144(1%), C-1664_144x144(45%) and C-1664_144x144 (unfiltered), respectively.

Although under-sampling may lead to a loss of a critical information13[13], the model already seemed to suffer some degree of overfitting, which could be further worsen with over-sampling, thus the former was applied for the study. Moreover, no additional data were readily available for the over-sampling method.

4.4 Hyperparameter tuning

Hyperparameter tuning is another way to improve model's performance before proceeding to testing phase. This section addresses the impact of different initial number of filters, dropout rate and batch size on the performance. The search was manual and I used the following values:

1. The initial number of filters: The default number of filters at the beginning of the TGS U-Net was 16. I also explored if a deeper network with 32 and 64 filters at the beginning can learn better features. The choice was inspired by P. Zhang et al. (2018). Model C-1664_144x144 was used. Table 10 shows the change in depth of a model based on the initial number of filters.
   The total number of parameters in the models was 1.18, 4.71 and 18.82 million, respectively.

   Table 10 Summary of changes in size inside U-Net depending on the initial number of filters

No. of filters	Model input size	Size at the bottleneck (min. spatial size, max. depth)	Model output size
16	144 x 144 x 3	9 x 9 x 256	144 x 144 x 1
32	144 x 144 x 3	9 x 9 x 512	144 x 144 x 1
64	144 x 144 x 3	9 x 9 x 1024	144 x 144 x 1

2. The dropout rate: The default dropout rate of the TGS U-Net was set to a low value of 0.05 because this regularization technique was reported by the author[29] as not very efficient for the particular TGS competition. I tested whether higher dropout rates, namely 0.2^{10}, 0.5^{10} (F. Zhang et al., 2015) and 0.75 (P. Zhang et al., 2018), can prevent the model from overfitting on my data. In this case, model C-1664_144x144(45%) described in the previous section 4.3 was used.
3. The batch size: The default batch size of the TGS U-Net was set to 32 that is generally viewed as an appropriate value (Bengio, 2012; Keskar et al., 2017; Masters & Luschi, 2018). Further, I experimented with smaller subsets of 15 samples fed to the network at a time and bigger ones, with the size of 50 samples, limited by the available GPU memory. Here similarly, the model C-1664_144x144(45%) was used.

I also examined using different optimizer (Nadam) and activation function (ELU, that supposedly learns representations more robust to noise (Iglovikov et al., 2017)), but the resulting differences were insignificant and therefore I decided not to include them in this thesis.

4.5 Test data

In this final test phase, all experiments were evaluated on independent tests sets described in 3.5.3. Since the models were trained on summer data, they were not expected to perform well with highly dissimilar winter images. For this reason, only some of the best performing models were evaluated on the test set 4. Table A 8 in the Appendix describes which test sets were used for the individual models.

5 Results

5.1 The baseline: sub-dataset A

1. Shrubs (832 x (100 x 100) px patches)

The results on validation data: accuracy = 0.46, precision = 0.23, recall = 0.96, f1 score = 0.31.

It can be noted, that the TGS U-Net in its original set up performs poorly with the small sub-dataset of shrubs class. The recall is high, which means that almost all shrubs are classified correctly. However, this metric alone is not representative, because as it can be seen in Table 11, there are more FPs than TPs and TNs combined, which means that the model labels too many pixels from other classes as shrubs. This is why the precision is so low. Figure 12 illustrates the confusion of bare land for shrubs, but this is the case also with trees and shadows. The first image in the figure is the original patch with black contours defining shrubs, the second is the binary mask with white polygons corresponding to ground truth shrubs, following is the heatmap of the probability of shrubs presence, with black contours to help to better orientate in the prediction and the last image is binary prediction after applying a threshold of probability bigger than 50%. Contours defining shrubs are present here in red to help to distinguish it from the black background.

Using predictions in form of continuous maps (heatmaps), instead of discrete classes, can be particularly useful in case of landscapes with a lot of transitions among vegetation species or types, where pixels can contain more than one vegetation type, even in VHR imagery (Kattenborn et al., 2020). Because a lot of shrubs in the area of interest occurs around and even under the trees, the heatmaps in combination with an expert opinion could be in fact more useful than binary maps in decision making process regarding the landscape management.

Table 11 Confusion matrix of predictions on validation data. Shrubs class, 832 patch-dataset

	Actual positives	**Actual negatives**
Predicted positives	206 996	707 120
Predicted negatives	8 403	453 737

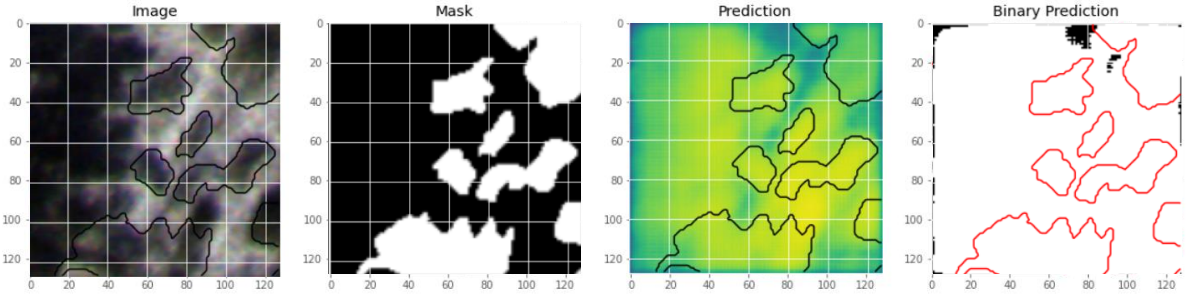


Figure 12 Visual example of the performance on validation data. Shrubs class, 832 patch-dataset. From left: image patch, binary mask, prediction and binary prediction with threshold 0.5.

2. Trees (832 x (100 x 100) px patches)

The results on validation data: accuracy = 0.79, precision = 0.83, recall = 0.83, f1 score = 0.83

In case of tree class, the model performs much better even with the small sub-dataset. Table 12 shows, that correctly labeled pixels (TPs, TNs) are many more, than the incorrectly labeled ones (FPs, FNs). The model can detect 83% of all trees (recall) and 83% of all pixels labeled as trees are actually trees (precision). Figure 13 shows two examples – one of a high recall and precision and the other of a low precision.

Table 12 Confusion matrix of predictions on validation data. Trees class, 832 patch-dataset

	Actual positives	**Actual negatives**
Predicted positives	623 697	127 504
Predicted negatives	126 486	498 569

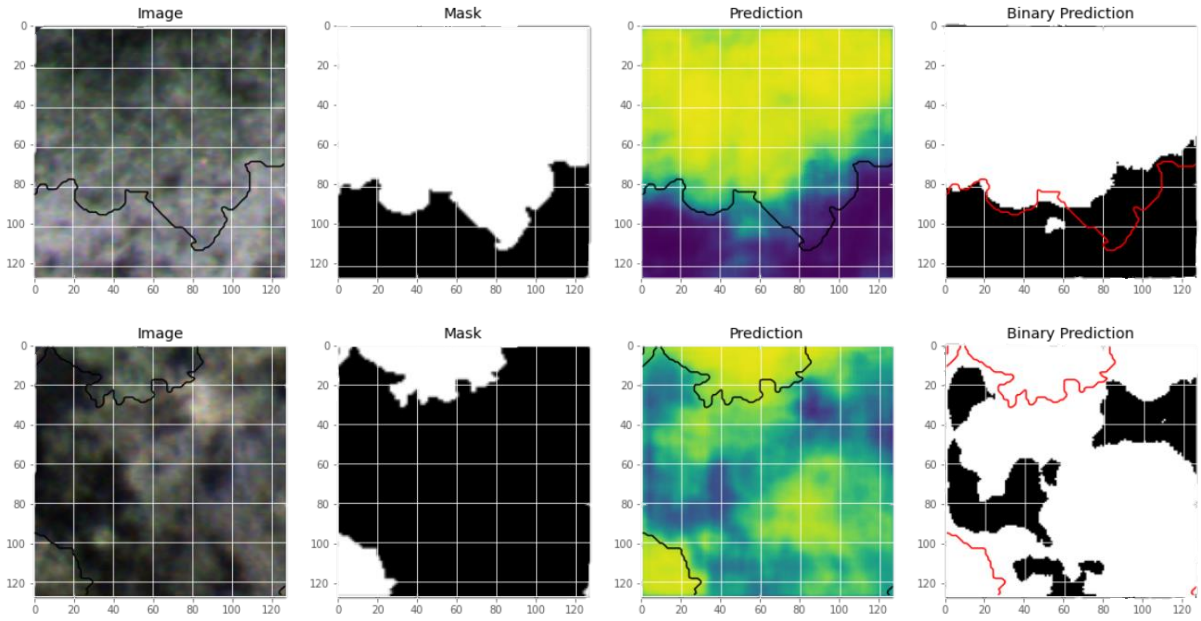


Figure 13 Two visual examples of the performance (top: better, with high recall and precision; bottom: worse, with low precision) on validation data. Trees class, 832 patch-dataset. From left: image patch, binary mask, prediction and binary prediction with threshold 0.5.

3. Shadows (832 x (100 x 100) px patches)

The results on validation data: accuracy = 0.90, precision = 0.77, recall = 0.63, f1 score = 0.69

Similarly like trees, pixels are mostly labeled correctly with respect to the shadows class (Table 13). The overall quality of detection is worse, than in case of trees. The model struggles to identify 37% of all shadows. Figure 14 displays a problem to identify shrub pixels in the object's boundaries.

Table 13 Confusion matrix of predictions on validation data. Shadows class, 832 patch-dataset

	Actual positives	**Actual negatives**
Predicted positives	150 786	45 729
Predicted negatives	87 891	1 091 850

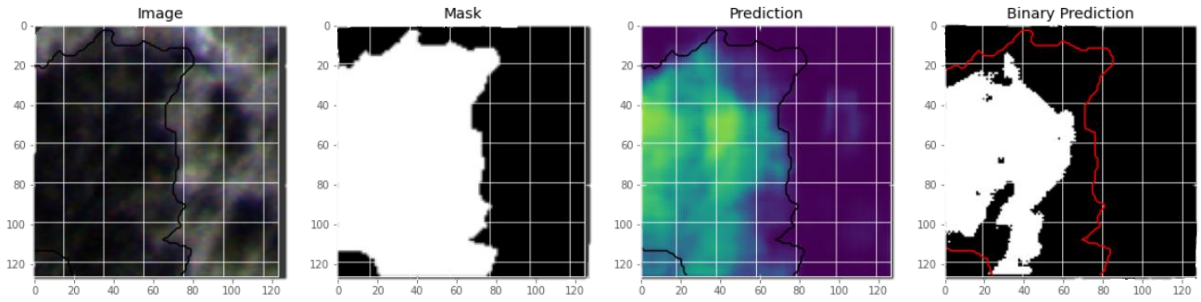


Figure 14 Visual example of the performance on validation data. Shadows class, 832 patch-dataset. From left: image patch, binary mask, prediction and binary prediction with threshold 0.5.

4. Rocks (832 x (100 x 100) px patches)

The results on validation data: accuracy = 0.99, precision = 0.72, recall = 0.18, f1 score = 0.29

Table 14 perspicuously shows the disproportional representation of rocks class. The accuracy is as high as 99.5% only because pixels belonging to this class are so few. The recall 18% indicates that the model clearly fails to detect majority of rocks. From Figure 15 can be concluded, that illumination by sunlight has an impact on the detection quality.

Table 14 Confusion matrix of predictions on validation data. Rocks class, 832 patch-dataset

	Actual positives	**Actual negatives**
Predicted positives	1 358	540
Predicted negatives	6 001	1 368 357

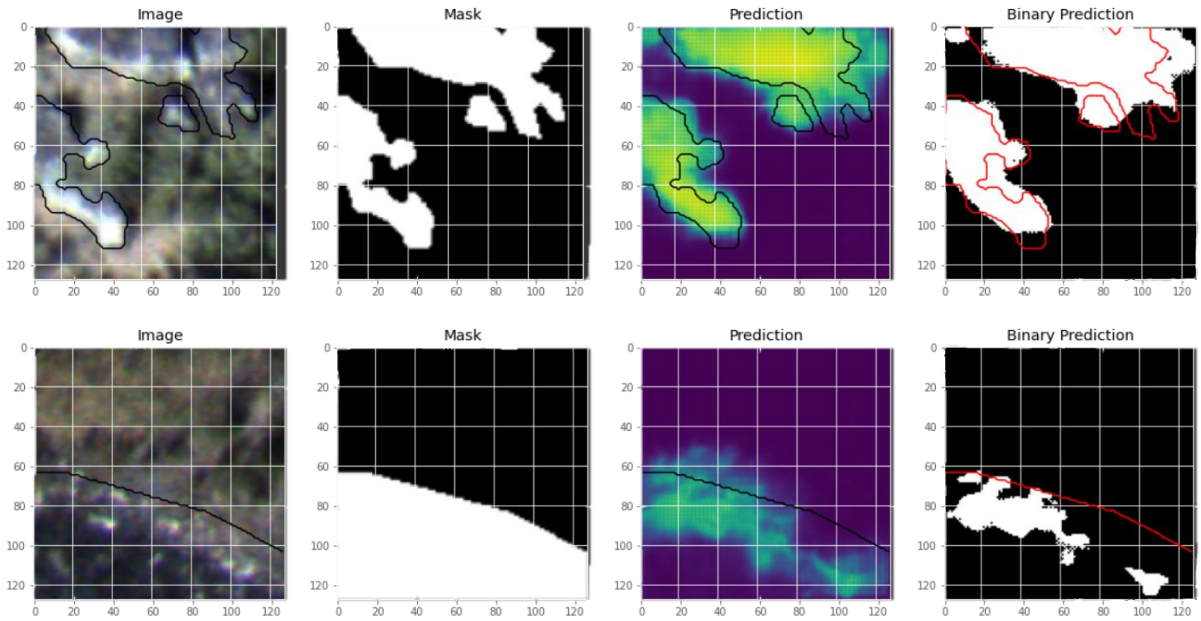


Figure 15 Two visual examples of the performance (top: better, bottom: worse) on validation data. Rocks class, 832 patch-dataset. From left: image patch, binary mask, prediction and binary prediction with threshold 0.5.

5. Shrubs – augmented (1664 x (100 x 100) px patches)

The results on validation data: accuracy = 0.84, precision = 0.71, recall = 0.56, f1 score = 0.63

Augmenting the sub-dataset twofold brought significant improvement, increasing the F1 score from 0.31 to 0.63. Precision also more than doubled (from 0.23 to 0.71), so in the results there is now more of the target pixels, than the ones from different classes. As a consequence, the recall dropped. The model stopped labeling too many pixels as the target class, which is also reflected in big decrease in FPs – from 51% to only 5% (Table 11 and Table 15, respectively; for details see Table A 3 in the Appendix).

Table 15 Confusion matrix of predictions on validation data. Shrubs class, 1664 patch-dataset

	Actual positives	**Actual negatives**
Predicted positives	347 615	139 200
Predicted negatives	275 509	1 973 804

6. Shrubs – augmented (3832 x (100 x 100) px patches)

The results on validation data: accuracy = 0.85, precision = 0.73, recall = 0.64, f1 score = 0.68

With further augmentation of the data, slight improvements in precision and F1 score are achieved, whereas the recall jumps up for almost 10%. A fraction of FNs now moved to TPs (Table 16 and Table A 3 in the Appendix), more pixels from the target group are now detected. Figure 16 shows the detection improvement.

Table 16 Confusion matrix of predictions on validation data. Shrubs class, 3832 patch-dataset

	Actual positives	**Actual negatives**
Predicted positives	930 008	349 334
Predicted negatives	513 684	4 498 430

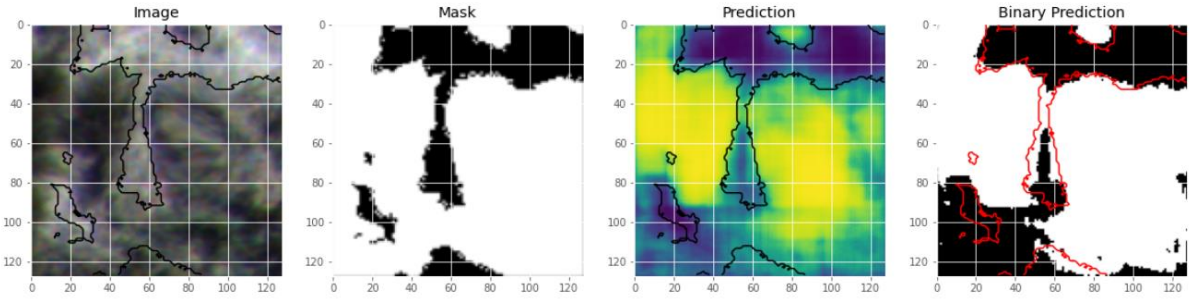


Figure 16 Visual example of the performance on validation data. Shrubs class, 3832 patch-dataset. From left: image patch, binary mask, prediction and binary prediction with threshold 0.5.

5.2 Sub-datasets B-E: patch size, scale and data augmentation

5.2.1 Summary

Table 17 is a summary of results of all performed experiments. Two best results per sub-dataset are always in bold. For all but sub-dataset B the higher value is the result of the time-consuming experiments that took 46-60 hours to train.

Table 17 The summary table of results of all performed experiments

Sub-dataset name	Patch dimensions (height x width) [px]	Sub-dataset size	Model input dimensions (height x width) [px]	F1 score
Sub-dataset B	200 x 200	808	128 x 128	0.61
		1658	128 x 128	0.76
			192 x 192	0.75
		3808	128 x 128	**0.83**
			192 x 192	**0.82**
Sub-dataset C	300 x 300	808	128 x 128	0.55
		1664	128 x 128	0.80
			144 x 144	0.82
		3808	128 x 128	0.83
			144 x 144	**0.86**
			288 x 288	**0.90**
Sub-dataset D	400 x 400	808	128 x 128	0.78
		1658	128 x 128	0.79
			192 x 192	**0.85**
			400 x 400	**0.90**
		3808	128 x 128	0.84
Sub-dataset E	500 x 500	808	128 x 128	0.00
		1658	128 x 128	0.80
			240 x 240	**0.87**
			496 x 496	**0.90**
		3808	128 x 128	0.82

Figure 17 is a qualitative comparison of the results of the three most time demanding experiments and the model C-1664_144x144. It is apparent, that many additional training hours do not bring a lot of benefit in this regard.

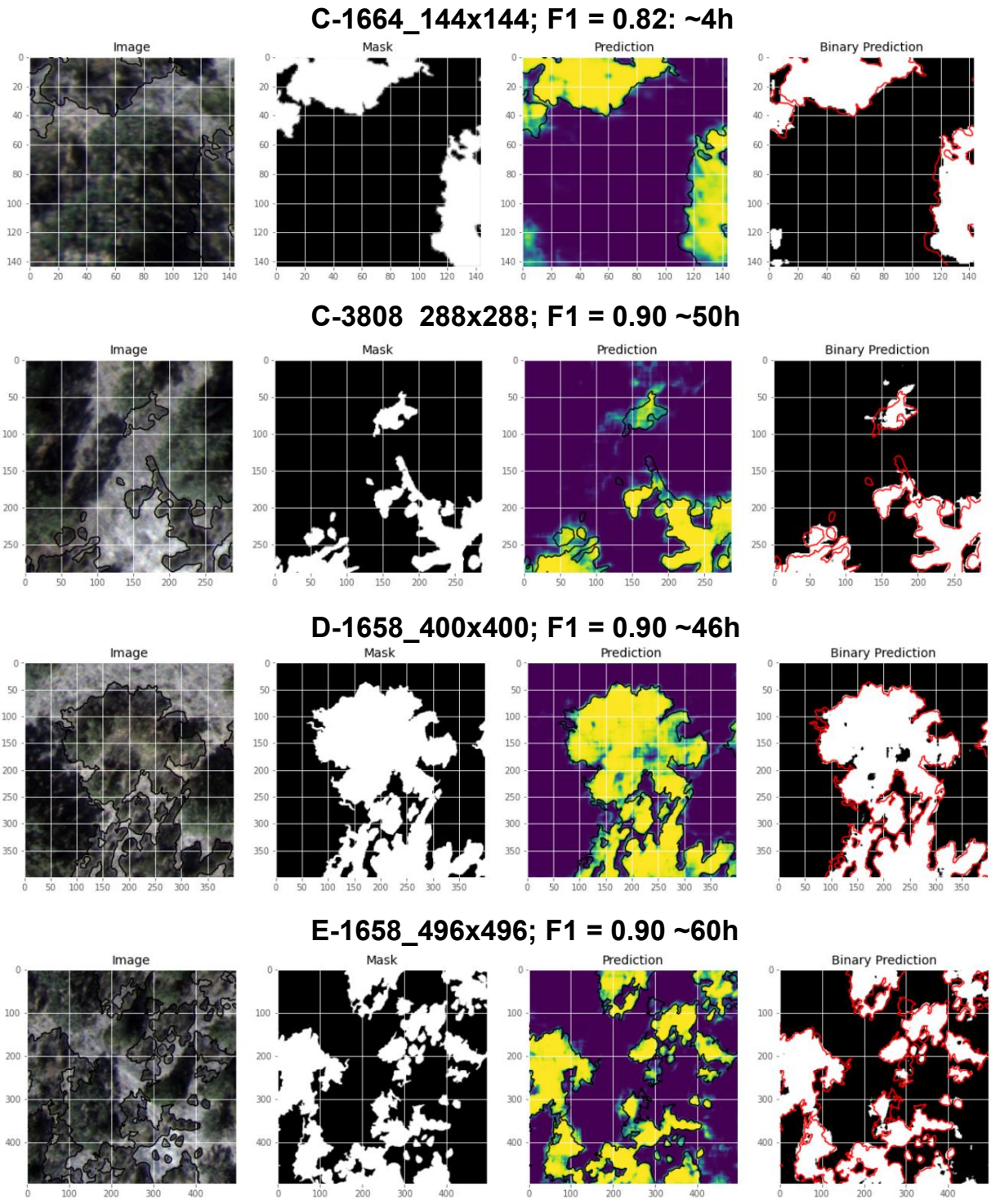


Figure 17 Qualitative comparison of the performance of the models with the longest training time and the best tradeoff model regarding time and performance, C-1664_144x144.

5.2.2 Impact of data augmentation and patch size

Figure 18 presents the impact of data augmentation within each sub-dataset on F1 score. The blue series symbolize 808-instance datasets, the green 1658-instance datasets and the red 3808-instance

datasets. Clearly, the F1 score improves with growing dataset size. This trend is especially strong with sub-dataset B. For the sub-datasets with bigger patch sizes the difference is not that big, especially between the bigger datasets (1658 and 3808) – the performance seems to be converging. Sub-dataset D has the flattest trendline.

The impact of patch size among sub-datasets can be also observed. Here, the sub-dataset D experiences the best performance with the smallest 808-instance dataset size, whilst the patch size doesn't seem to provoke much of a difference among 3808-instance sized datasets and the F1 values oscillate around 0.83 for all sub-datasets. Despite running the model on sub-dataset E with 808 instances multiple times it always failed to detect any shrubs. Neural networks don't assure to reach a global optima, so probably the learning converged into some bad local minima, but at the moment I can't offer a proper explanation for this phenomenon. This model is treated as an outlier with regards to discussion. All models' input dimensions are (128 x 128) px.

However, applying random data augmentation techniques yields different data every time, influencing class distribution in the dataset (Figure 19). This may impact the results as well. Model (C-1664_144x144) with newly generated data, using slightly different set of augmentation techniques, performed a bit worse than its predecessor (0.05 drop in accuracy 0.04 in recall).

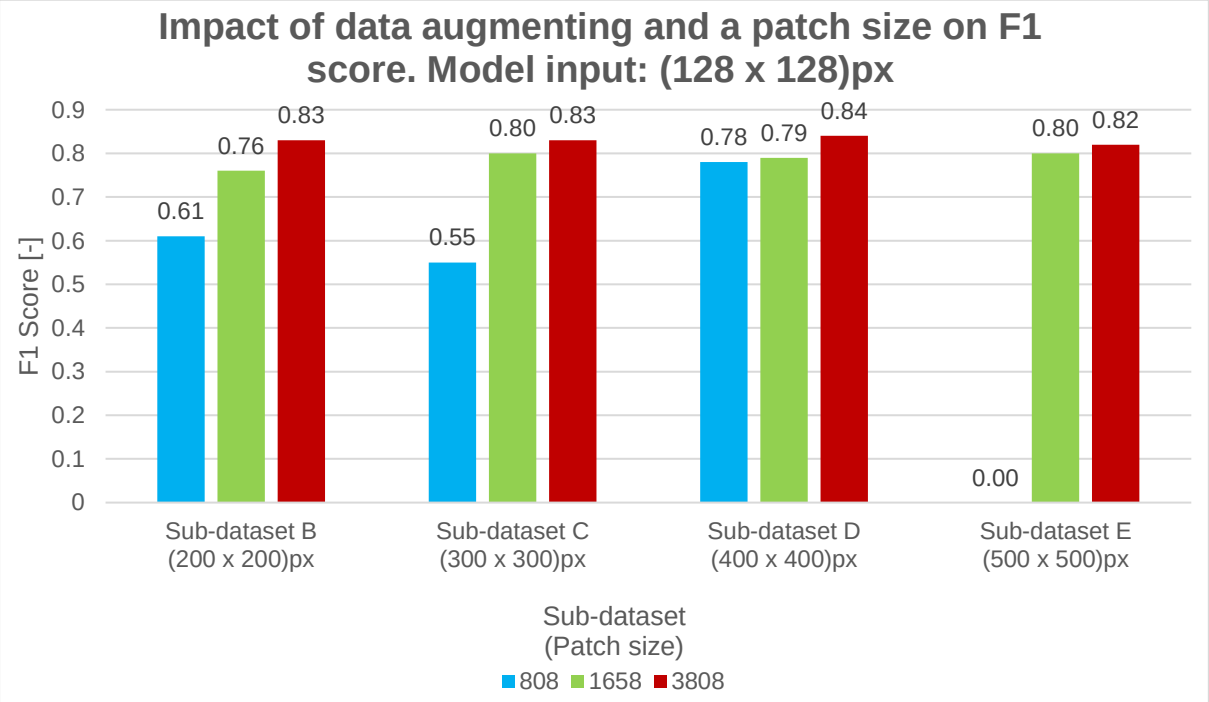


Figure 18 Impact of data augmenting and a patch size on F1 score. Model input: (128 x 128) px

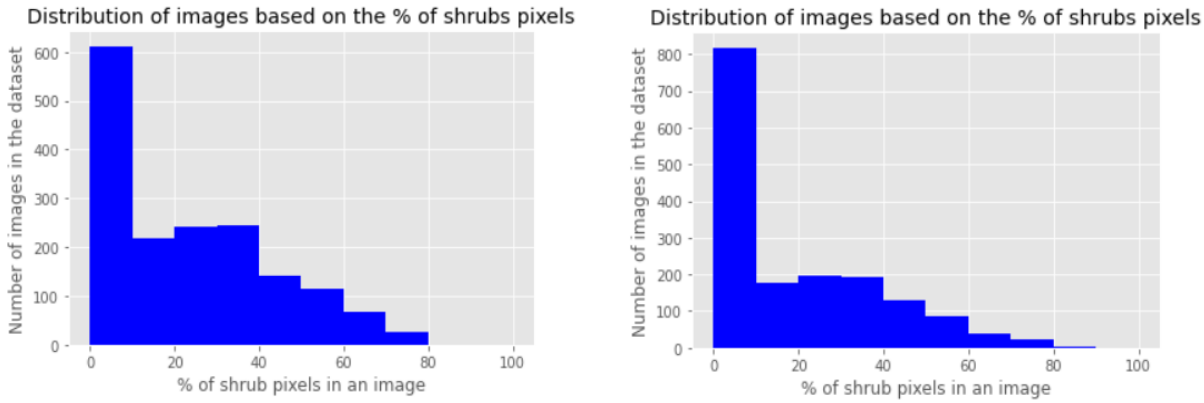


Figure 19 Impact of data augmentation on class distribution in the dataset: two runs of random augmentations of sub-dataset C (1664 patches) with the same techniques

5.2.3 Impact of down-scaling

Figure 20 takes a closer look at the impact of down-scaling of patches on the performance. The red series are the base patch size ((128 x 128) px), which therefore represents different scaling ratios for different patch sizes (from 42.67% to 25.6% of the original patch size). The yellow series represent approximately 50% of the original patch size and the green ones roughly 100%, which means basically no down-scaling. While the impact is indistinct in case of sub-dataset B, for the bigger patch sized datasets it is definitely better to minimize down-scaling, since it has negative impact on F1 score. However, for the sake of a tradeoff between the performance and time, the scale 1:2 seems to be generally the most viable option, similarly as for Rakhlin et al. (2018). The best results plateau on 0.90, which is an important information to consider, since each of these three experiments has a different training time (50h for sub-dataset C, 46h for sub-dataset D and 60h for sub-dataset E). On average, 3808-instance sub-dataset C is the best performing one regarding F1 score.

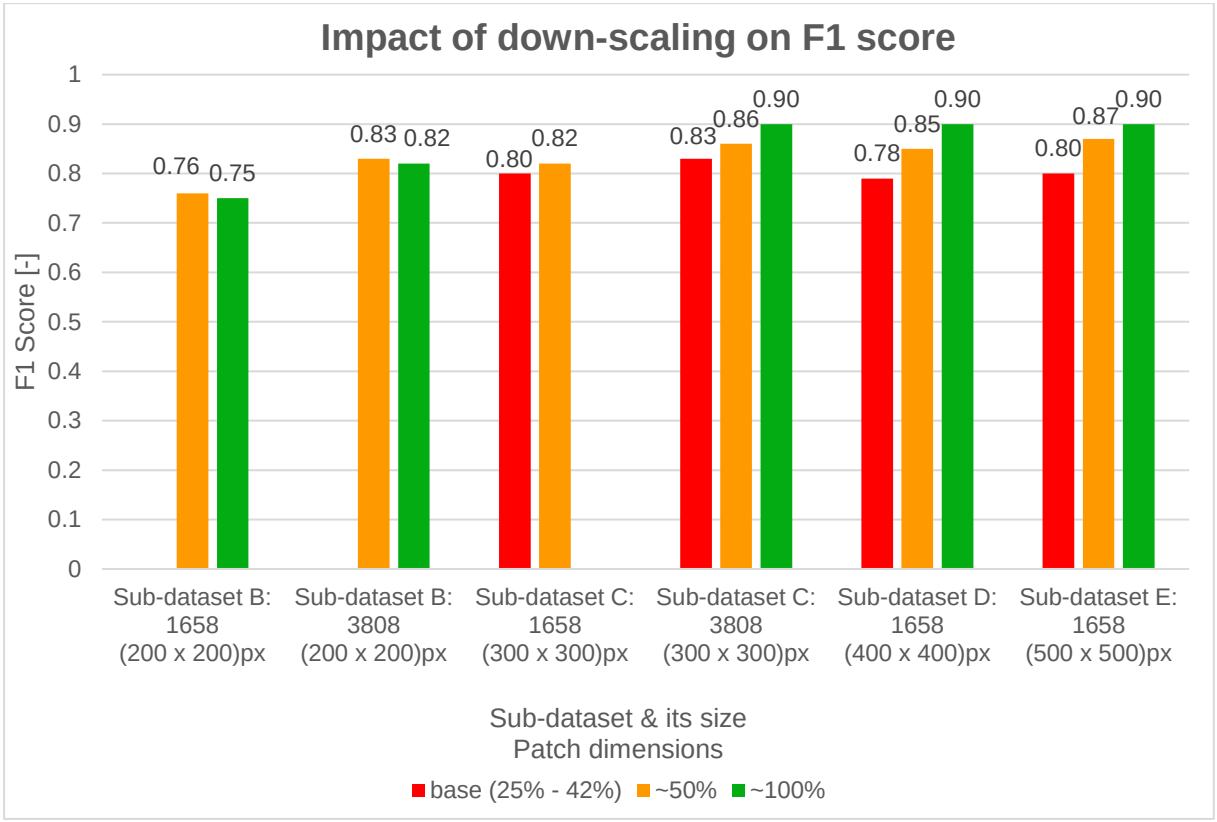


Figure 20 Impact of down-scaling on F1 score

5.2.4 Detailed results on validation set

Sub-dataset B: (200 x 200) px patches

1. 808 patches; model input dimensions: (128 x 128) px

The results on validation data: accuracy = 0.83, precision = 0.80, recall = 0.49, f1 score = 0.61.

Comparing to the sub-dataset A, twice as big patch size helped to achieve similar F1 score with almost five times smaller dataset. 80% of the pixels marked as shrubs actually belong to this class, however only half of them gets detected. Figure 21 is a perfect example of a borderline case, where it is not clear whether the model struggles to classify shrubs on extremely deformed images or the labels are just incorrect.

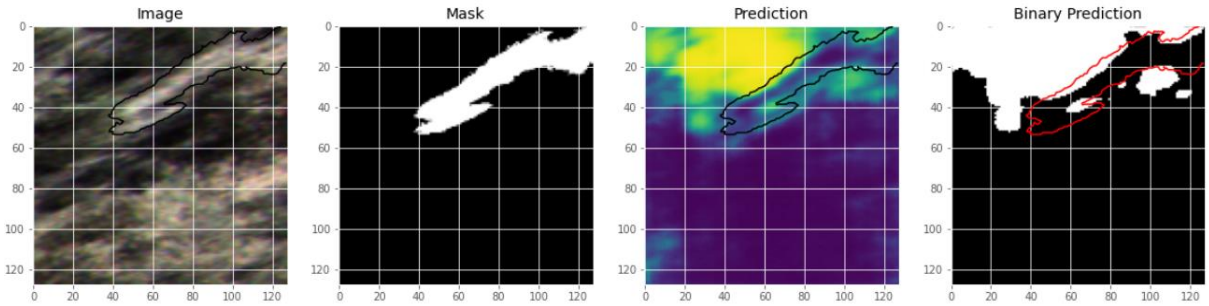


Figure 21 Visual example of the performance on deformed validation data. Shrubs, 808 x (200 x 200) px patch-dataset, model input: (128 x 128) px. From left: image patch, binary mask, prediction and binary prediction with threshold 0.5.

2. 1658 patches; model input dimensions: (128 x 128) px

The results on validation data: accuracy = 0.87, precision = 0.90, recall = 0.65, f1 score = 0.76.

Augmenting the sub-dataset to twice the size leads to great improvements – 10%, 16% and 15% higher precision, recall and F1 score, respectively. Figure 22 shows the improvement in performance in comparison to the best result with sub-dataset A (Figure 16).

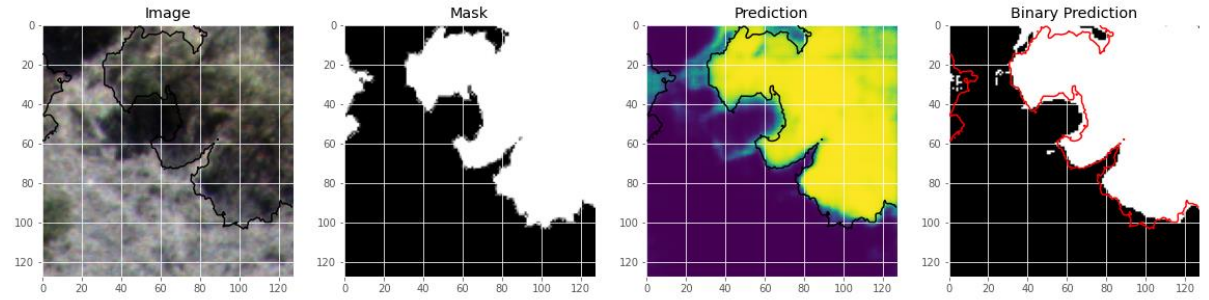


Figure 22 Visual example of the performance on validation data. Shrubs, 1658 x (200 x 200) px patch-dataset, model input: (128 x 128) px. From left: image patch, binary mask, prediction and binary prediction with threshold 0.5.

3. 1658 patches; model input dimensions: (192 x 192) px

The results on validation data: accuracy = 0.87, precision = 0.86, recall = 0.67, f1 score = 0.75.

Almost no downscaling of the patches (96% of the original patch size) basically didn't impact the results. What it did impact, though, was the training time, that grew from about 4.5 hours to 9.5.

Figure 23 poses the same question as Figure 21 – does the model struggle to work with augmented data, or it actually performs well and what is incorrect are the labels?

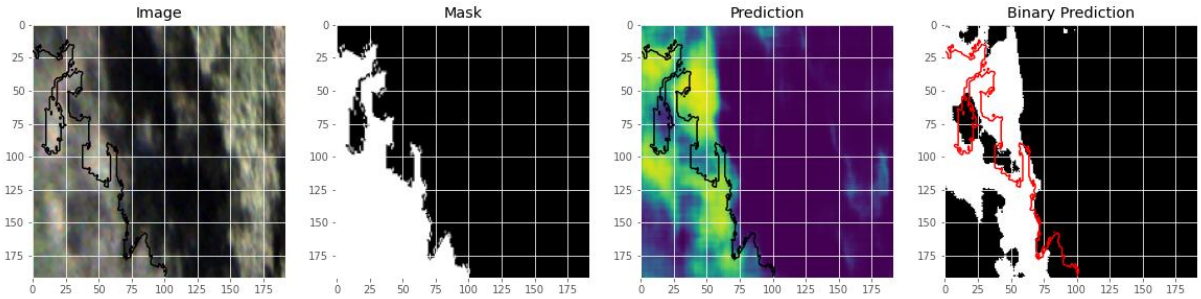


Figure 23 Visual example of the performance on deformed validation data. Shrubs, 1658 x (200 x 200) px patch-dataset, model input: (192 x 192) px. From left: image patch, binary mask, prediction and binary prediction with threshold 0.5.

4. 3808 patches; model input dimensions: (128 x 128) px

The results on validation data: accuracy = 0.90, precision = 0.94, recall = 0.75, f1 score = 0.83.

Certainly, additional augmentation improves the performance, F1 grows for 8%. On a patch of land of approximately (12 x 12) m (Figure 24), for the purposes of landscape planning and grazing management, this particular result could be viewed as satisfactory.

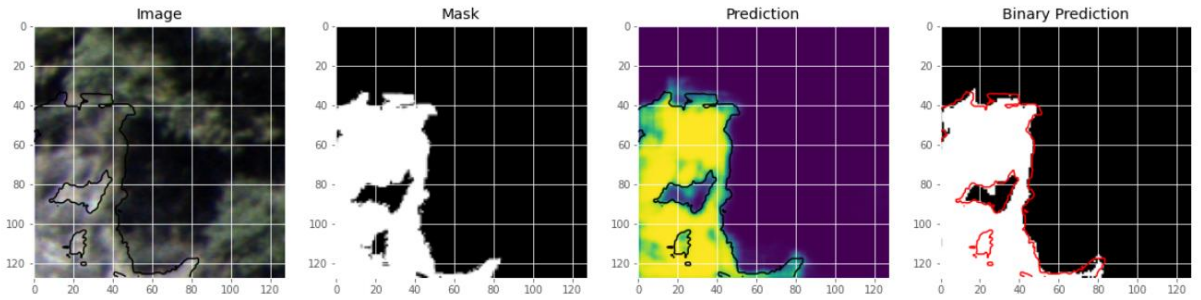


Figure 24 Visual example of the performance on validation data. Shrubs, 3808 x (200 x 200) px patch-dataset, model input: (128 x 128) px. From left: image patch, binary mask, prediction and binary prediction with threshold 0.5.

5. 3808 patches; model input dimensions: (192 x 192) px

The results on validation data: accuracy = 0.91, precision = 0.90, recall = 0.76, f1 score = 0.82.

The experience with minor downscaling was the same also in this case, it had basically no effect on the results. The improvement of 1-2% is not worth the disproportionate increase in training time.

However, analyzing confusion matrices of all these experiments (Table A 4 in the Appendix), one can see gradual decrease in FNs (from 11.87% with 808 patches and model input (128 x 128) px to 5.86% with 3808 patches and model input (192 x 192) px), so the presented techniques are slightly helping the model to better detect shrubs.

Sub-dataset C: (300 x 300) px patches

1. 808 patches; model input dimensions: (128 x 128) px

The results on validation data: accuracy = 0.66, precision = 0.48, recall = 0.64, f1 score = 0.55.

Increasing the size of the patch for another 100 pixels in both dimensions yielded many times in a row worse results than with (200 x 200) px patches, the F1 score dropped from 0.61 to 0.55, although the biggest drop was recorded for the precision (from 0.80 to 0.48). The model started to label too many pixels from other classes as shrubs.

2. 1664 patches; model input dimensions: (128 x 128) px

The results on validation data: accuracy = 0.84, precision = 0.96, recall = 0.68, f1 score = 0.80.

Bigger patch size brought expected outcome – the performance improved in comparison to same sub-dataset size of (200 x 200) px patches. From Table 18 may be seen that FPs are below 1% (0.92%), which didn't happen in the previous experiments. That also explains the very high precision.

Table 18 Confusion matrix of predictions on validation data. Shrubs, 1664 x (300 x 300) px patch-dataset, model input: (128 x 128) px

	Actual positives	**Actual negatives**
Predicted positives	554 954	25 096
Predicted negatives	258 583	1 897 495

3. 1664 patches; model input dimensions: (144 x 144) px

The results on validation data: accuracy = 0.88, precision = 0.96, recall = 0.72, f1 score = 0.82.

Rescaling to 48% of the original patch brings only minimal improvements. In this case, it is more understandable than previously, since the rescaling from (128 x 128) px only changed for about 5% of the original patch size, unlike in case of (200 x 200) px patches (64% of the original patch size with (128 x 128) px and 96% with (196 x 196 px)). The model's capacity to correctly interpret even complex shapes is undeniably improving (Figure 25).

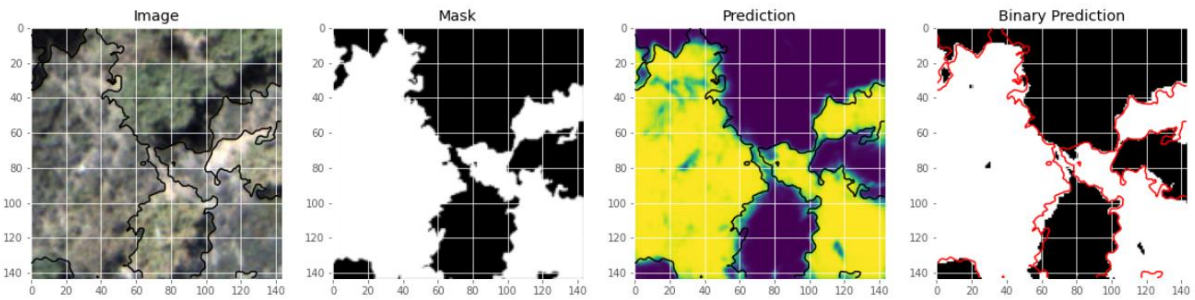


Figure 25 Visual example of the performance validation data. Shrubs, 1664 x (300 x 300) px patch-dataset, model input: (144 x 144) px. From left: image patch, binary mask, prediction and binary prediction with threshold 0.5.

4. 3808 patches; model input dimensions: (128 x 128) px

The results on validation data: accuracy = 0.85, precision = 0.97, recall = 0.73, f1 score = 0.83.

Further extension of the sub-dataset brings marginal improvements. The model takes almost 9 hours to train. In Figure 26 another illustration of a great performance with even complex shapes.

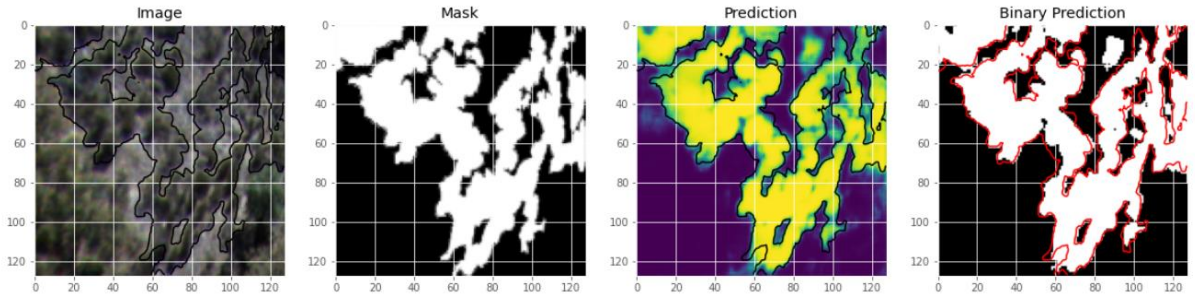


Figure 26 Visual example of the performance on validation data. Shrubs, 3808 x (300 x 300) px patch-dataset, model input: (128 x 128) px. From left: image patch, binary mask, prediction and binary prediction with threshold 0.5.

5. 3808 patches; model input dimensions: (144 x 144) px

The results on validation data: accuracy = 0.88, precision = 0.96, recall = 0.77, f1 score = 0.86.

In this case as well, the same small difference in rescaling ratios brings only small improvements. However, this is the best performance achieved so far. The training takes only about 30 minutes more than in the previous experiment. The confusion matrix with a notably high rates of correctly classified pixels can be seen in Table 19. The qualitative evaluation is illustrated in Figure 27.

Table 19 Confusion matrix of predictions on validation data. Shrubs, 3808 x (300 x 300) px patch-dataset, model input: (144 x 144) px

	Actual positives	**Actual negatives**
Predicted positives	1 607 796	62 251
Predicted negatives	474 914	5 755 455

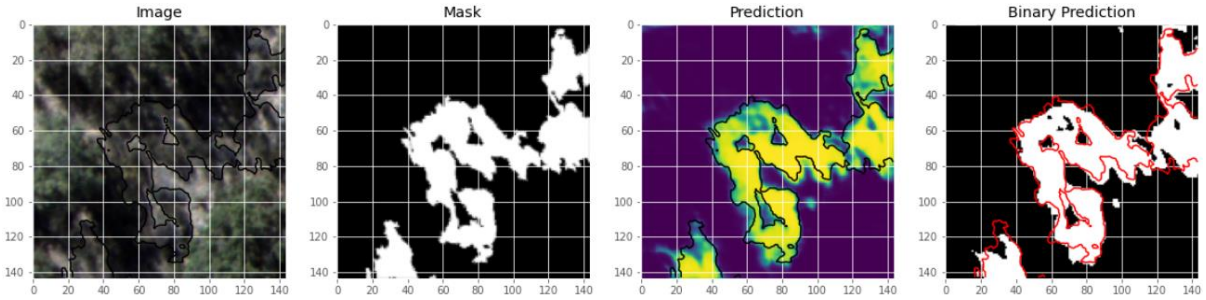


Figure 27 Visual example of the performance on validation data. Shrubs, 3808 x (300 x 300) px patch-dataset, model input: (144 x 144) px. From left: image patch, binary mask, prediction and binary prediction with threshold 0.5.

6. 3808 patches; model input dimensions: (288 x 288) px

The results on validation data: accuracy = 0.94, precision = 0.90, recall = 0.90, f1 score = 0.90.

The trend of increasing performance with the increasing scale continues with the scale of 96%. However, even though F1 score raised for 4%, the training took around 50 hours. This is enormous time-consumption, especially considering that visually it is hard to tell the difference from the previous two experiments that take four times less to train.

Generally, the percentage of FPs was very low in these experiments, usually below 1%, with an exception of the smallest 808-instance dataset (Table A 5 in the Appendix). The model is now pretty good at distinguishing between shrubs and other classes.

Sub-dataset D: (400 x 400) px patches

1. 808 patches; model input dimensions: (128 x 128) px

The results on validation data: accuracy = 0.79, precision = 0.97, recall = 0.65, f1 score = 0.78.

With the patch size (400 x 400) px, this experiment significantly outperformed all of the previous ones in this category of small 808-instance datasets. Exceptionally high precision indicates that the model thinks there are more shrubs than there actually are.

2. 1658 patches; model input dimensions: (128 x 128) px

The results on validation data: accuracy = 0.81, precision = 0.98, recall = 0.67, f1 score = 0.79.

Improvement of 1-2% takes 5 hours – approximately double the time of the 808-patch dataset.

3. 1658 patches; model input dimensions: (192 x 192) px

The results on validation data: accuracy = 0.87, precision = 0.97, recall = 0.75, f1 score = 0.85.

Despite the model takes 9 hours to train, rescaling the patches to 48% instead of 32% brings noteworthy improvement of 6% in F1 score. Only around 7% of all pixels are FNs and 0.6% are FPs (Table 20 and Table A 6 in the Appendix). Considering the tradeoff between time and performance, this is the second best model, after the 3808-patch sub-dataset C with model input dimensions (144 x 144) px. The illustration of the performance is in Figure 28.

Table 20 Confusion matrix of predictions on validation data. Shrubs, 1658 x (400 x 400) px patch-dataset, model input: (192 x 192) px

	Actual positives	**Actual negatives**
Predicted positives	1 311 234	37 863
Predicted negatives	433 427	4 336 900

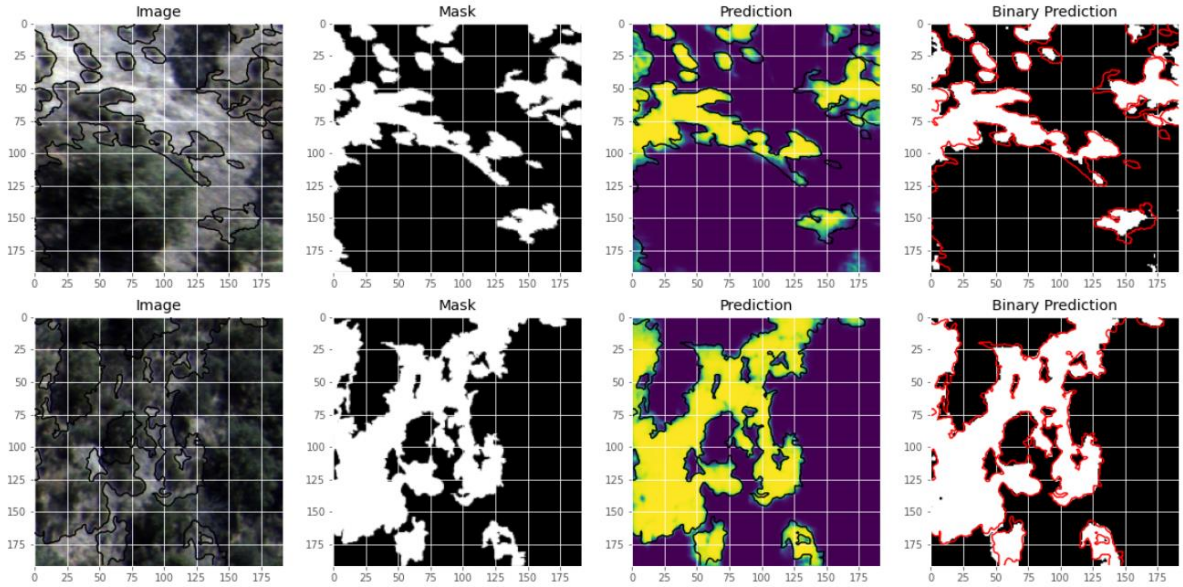


Figure 28 Two visual examples of the performance on validation data. Shrubs, 1658 x (400 x 400) px patch-dataset, model input: (192 x 192) px. From left: image patch, binary mask, prediction and binary prediction with threshold 0.5.

4. 1658 patches; model input dimensions: (400 x 400) px

The results on validation data: accuracy = 0.96, precision = 0.92, recall = 0.89, f1 score = 0.90.

Another significant improvement of 5% in F1 score is achieved by using the 1:1 rescaling ratio. Considering only the performance this is the second best result, after the 3808-patch sub-dataset C with model input dimensions close to the original patch, (288 x 288) px. This performance, however, also comes at a price of 46-hour long training.

5. 3808 patches; model input dimensions: (128 x 128) px

The results on validation data: accuracy = 0.84, precision = 0.99, recall = 0.73, f1 score = 0.84.

Six percent drop in F1 score comes with a relief of only 10.5 hours needed for the training. However, model seems to perform better and takes less time to train under conditions described in point 3 (i.e. 1658-instance dataset with (192 x 192) px model input size).

Sub-dataset E: (500 x 500) px patches

1. 808 patches; model input dimensions: (128 x 128) px

The results on validation data: accuracy = 0.64, precision = 1.00, recall = 0.00, f1 score = 0.00.

The model under these conditions repeatedly performed very bad. It was incapable to learn any relevant information about the class, labeling essentially all pixels in the validation set as non-shrubs (Table 21).

It took two hours for the model to learn to predict outputs such as Figure 29.

Table 21 Confusion matrix of predictions on validation data. Shrubs, 808 x (500 x 500) px patch-dataset, model input: (128 x 128) px

	Actual positives	**Actual negatives**
Predicted positives	285	0
Predicted negatives	472 896	853 923

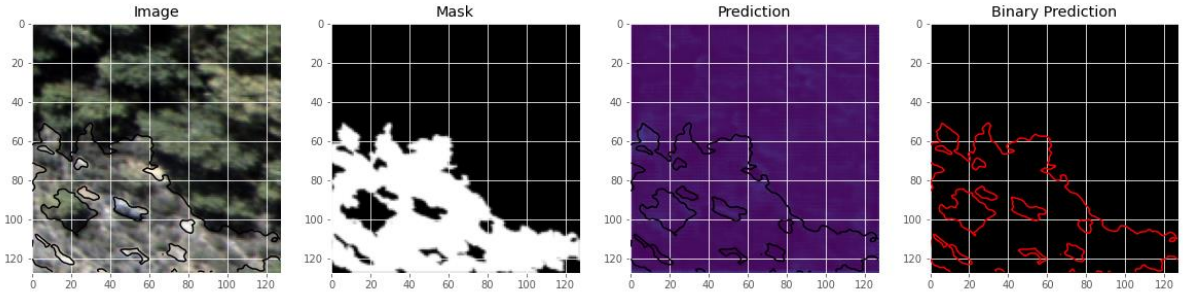


Figure 29 Two visual examples of the performance (top: better, bottom: worse) on validation data. Shrubs, 808 x (500 x 500) px patch-dataset, model input: (128 x 128) px. From left: image patch, binary mask, prediction and binary prediction with threshold 0.5.

2. 1652 patches; model input dimensions: (128 x 128) px

The results on validation data: accuracy = 0.79, precision = 0.98, recall = 0.68, f1 score = 0.80.

Doubling up the number of instances brought a dramatic turn – a sudden raise in F1 score from zero to 0.8 in only 4.5 hours. The model can detect 68% of the shrub pixels and only 2% of the detected pixels are from another class. Figure 30 brings two visual examples.

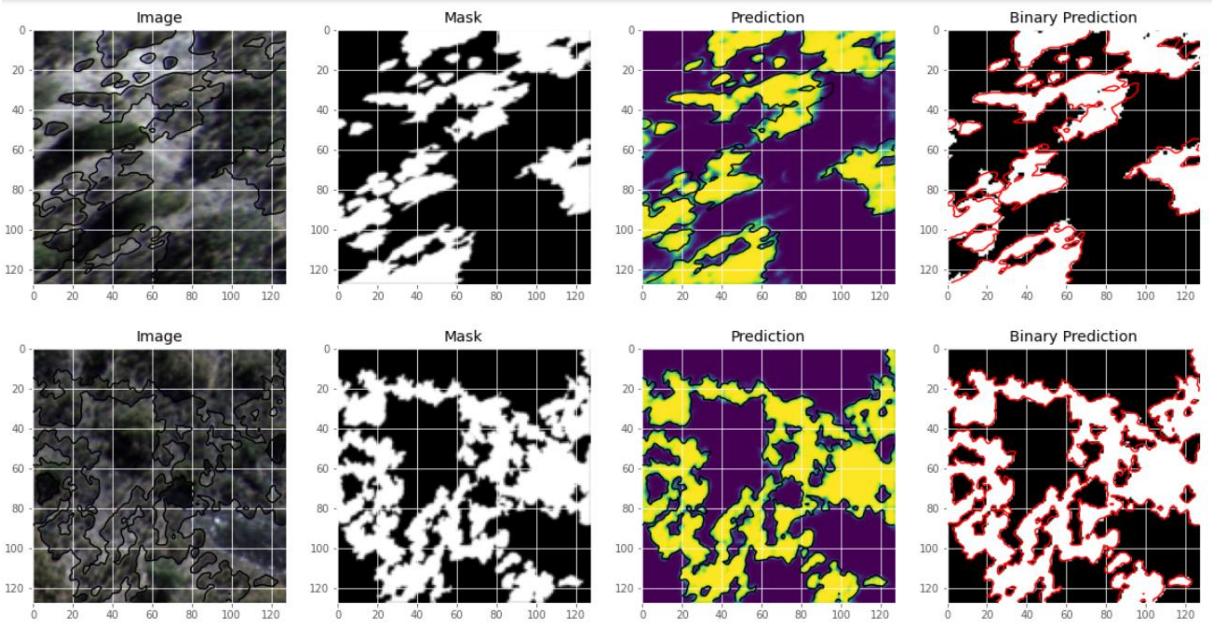


Figure 30 Two visual examples of the performance on validation data. Shrubs, 1652 x (500 x 500) px patch-dataset, model input: (128 x 128) px. From left: image patch, binary mask, prediction and binary prediction with threshold 0.5.

3. 1652 patches; model input dimensions: (240 x 240) px

The results on validation data: accuracy = 0.89, precision = 0.98, recall = 0.78, f1 score = 0.87.

Down-scaling the patch to only 48% of the original (500 x 500) px one brings about 7% improvement in F1 score. The precision remains at 98%, whereas the recall raises up for 10%. The training lasted 15.8 hours.

4. 1652 patches; model input dimensions: (496 x 496) px

The results on validation data: accuracy = 0.95, precision = 0.96, recall = 0.86, f1 score = 0.90.

In this case, almost no down-scaling was applied. The F1 score is 0.90, the highest score achieved in these experiments. Recall improved for additional 8%, generating the all-low number of FNs – only 3.56% (Table A 7 in the Appendix). This is the third time that an F1 score of 0.90 was achieved, however, at a cost of 60 hours of training.

5. 3808 patches; model input dimensions: (128 x 128) px

The results on validation data: accuracy = 0.79, precision = 0.99, recall = 0.70, f1 score = 0.82.

Getting back to down-scaling the patch to 25.6% of the original one, the F1 score is only 2% higher than in case 2. (0.8) but takes around 10.5 hours to train – more than double the time of case 2.

Table 22 Confusion matrix of predictions on validation data. Shrubs, 3808 x (500 x 500) px patch-dataset, model input: (128 x 128) px

	Actual positives	**Actual negatives**
Predicted positives	1 454 548	7 387
Predicted negatives	622 363	4 158 006

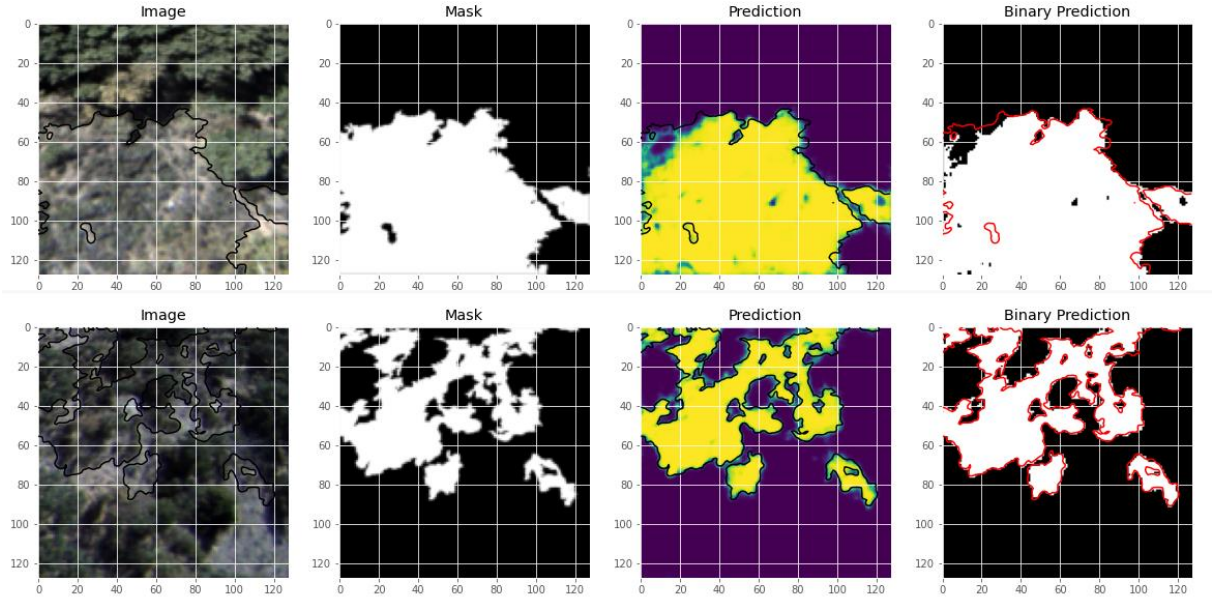


Figure 31 Two visual examples of the performance (top: better, bottom: worse) on validation data. Shrubs, 3808 x (500 x 500) px patch-dataset, model input: (128 x 128) px. From left: image patch, binary mask, prediction and binary prediction with threshold 0.5.

5.2.5 Performance fluctuation

All the above-described experiments were run just once, with four exceptions: models B-1658_128x128, C-1664_144x144 and C-1664_144x144(newly augmented) with newly augmented data using the same augmentation techniques, and C-3808_288x288. These models underwent several runs to see how much did results differ among runs. All parameters were therefore held constant for each individual model during all the runs. I chose model C-1664_144x144 as the one I used in the majority of the experiments and I used it with two separate datasets that were augmented with the same techniques but were not identical, then I chose C-3808_288x288 as the well performing one and B-1658_128x128 as another model with acceptable results within a relatively short training time. Averages, absolute uncertainties and ranges[34] of the metrics evaluated on validation data can be seen in Table 23. These results are, of course, insufficient for any statistical purposes. Many more runs would be needed to obtain robust distribution of the output fluctuations, but this was not feasible because of computational and time constraints. The average range of all the results is 0.04, 0.03 after excluding the major outliers in model B-1658_128x128 – recall and subsequently F1 score, and 0.02 after excluding the entire model B-1658_128x128. Since there was only one run out of overall 13 with such a difference in recall (and F1), I considered the 0.02 range as the most representative one. Ranges in the Table 23 do not always equal double the value of the corresponding absolute uncertainty as a result of rounding.

[34] Range was calculated as the difference between maximum and minimum measured value. It is double the value of an absolute uncertainty.

Table 23 Averages, absolute uncertainties and ranges of metrics evaluated on validation data of four different models

No.	1	2	3	4
Model	B-1658_128x128	C-1664_144x144	C-1664_144x144 (newly augmented)	C-3808_288x288
No. of runs	4	3	3	3
Total time [h]	~17	~13	~13	~150
$Accuracy_{avg}$ ± ∆ Accuracy	0.85 ± 0.03	0.88 ± 0.01	0.83 ± 0.00	0.92 ± 0.02
Accuracy range	0.05	0.01	0	0.04
$Precision_{avg}$ ± ∆ Precision	0.90 ± 0.01	0.95 ± 0.01	0.95 ± 0.00	0.89 ± 0.01
Precision range	0.01	0.03	0.01	0.02
$Recall_{avg}$ ± ∆ Recall	0.59 ± 0.09	0.72 ± 0.02	0.68 ± 0.01	0.89 ± 0.04
Recall range	0.19	0.04	0.02	0.04
$F1_{avg}$ ± ∆ F1	0.71 ± 0.08	0.82 ± 0.02	0.79 ± 0.00	0.89 ± 0.02
F1 range	0.17	0.04	0.01	0.03

5.3 Balancing the datasets

The main metrics evaluating the performance of each experiment can be found in Table 24. With raising representation of shrub pixels in the dataset dropping accuracy, precision and F1 score can be noticed. From an overview of shrub distribution in the different versions of the sub-dataset C (Figure A 8 in the Appendix) can be seen that there is about 18% of patches with less than 1% of shrub cover in the non-filtered (non-augmented) sub-dataset C, while around 87% patches contain less than 45% of shrub cover. This disproportion could be the reason of a different magnitude of the drop in accuracy, precision and F1 score between experiments 1 and 2, and 2 and 3. From Figure A 8 (6) in the Appendix can be also seen that shrubs are slightly over-represented in the 3rd experiment, which leads to a raise in recall. Figure 32 and Figure 33 show the qualitative difference in classification between model C-1664_144x144(1%) and C-1664_144x144(45%) on the same patch.

Table 24 Summary of results using datasets with different extent of shrub representation

No.	**Model**	**Shrubs in the dataset [%]**	**Accuracy**	**Precision**	**Recall**	**F1 score**
1	C-1664_144x144(unfilt.)	24	0.88	0.96	0.72	0.82
2	C-1664_144x144(1%)	29	0.86	0.86	0.77	0.81
3	C-1664_144x144(45%)	60	0.74	0.73	0.78	0.76

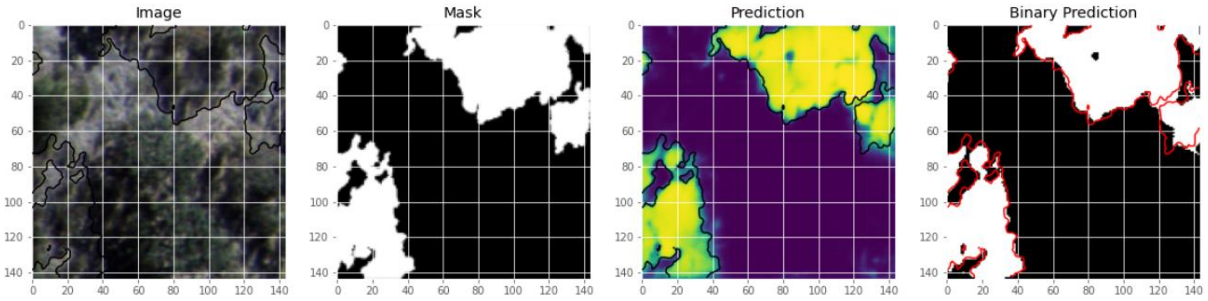


Figure 32 Illustration of the classification results of model C-1664_144x144(1%)

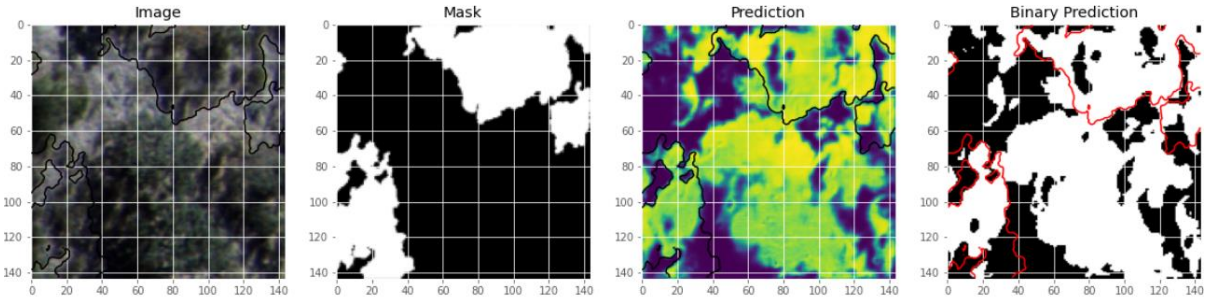


Figure 33 Illustration of the classification results of model C-1664_144x144(45%)

Figure A 8 (3) & (6) in the Appendix is a perfect example of the randomness of the re-distribution of class representation when data augmentation is applied. Therefore, implementing this technique can be a contributor of a more difficult reproducibility of the experiment.

5.4 Hyperparameter tuning

The impact of increasing depth of the network on the performance is summarized in Table 25. There is not much of a difference in the accuracies and precisions, however the drop in recall, leading to drop in F1 score, is apparent for the deepest network with 64 initial filters. The network gets worse in noticing shrubs. Moreover, it takes around 30 hours to train, while the first two models only take around 5 and 10 hours, respectively. Considering the little difference in performance between the model no. 1 and 2, the former one – the model in original setting using only 16 filters, looks like the best compromise between time and performance.

Table 25 The summary of the impact of increasing network's depth on the performance

No.	Number of filters	Accuracy	Precision	Recall	F1 score
1	16 (original setting)	0.88	0.96	0.72	0.82
2	32	0.89	0.97	0.75	0.84
3	64	0.87	0.97	0.62	0.76

Increasing the dropout rate obviously spoils the performance (Table 26). The only exception is recall that raises to 1 with the highest dropout rate. This could be misleading though, since the model could be simply labeling most of the pixels as shrubs. The best performance is achieved with the smallest

dropout rate, that was a part of the original setting. There is less deterioration in metrics between dropout rates 0.05 and 0.2 but begins to be more apparent with bigger dropout rates. The change is especially pronounced between dropouts 0.2 and 0.5, where the decline especially in accuracy, but also in precision and F1 score, is significant (0.3, 0.12 and 0.08, respectively).

Table 26 The summary of the impact of different dropout rate on the performance

No.	Dropout rate	Accuracy	Precision	Recall	F1 score
1	0.05 (original setting)	0.74	0.73	0.78	0.76
2	0.2	0.71	0.73	0.75	0.74
3	0.5	0.41	0.61	0.71	0.66
4	0.75	0.41	0.60	1.00	0.75

Finally, as Table 27 shows, the average performance doesn't change much within this small range of batch sizes. There is only about maximum of 30 minutes difference in training time between consecutive models.

Table 27 The summary of the impact of different batch size on the performance

No.	Batch size	Accuracy	Precision	Recall	F1 score
1	15	0.73	0.76	0.78	0.77
2	32 (original setting)	0.74	0.73	0.78	0.76
3	50	0.72	0.76	0.78	0.77

5.5 Test data

The test phase showed generally worse performance of the models on the test data. The F1 score is usually the highest with the test set 2, but an absolute highest F1 score of 0.77 was achieved by model C-3808_288x288 with the test set 1. The second highest F1 score (0.76) was achieved by models C-3808_288x288, C-1664_144x144(45%), C-1664_144x144(BS=15) and E-1658_240x240 using test set 2. The highest F1 score averaged over test sets 1, 2 and 3 was thus achieved by model C-3808_288x288, reaching 0.72. Figure 34 – Figure 36 show the performance of this model on test sets 1-3 and Figure 37 illustrates its performance on test set 4. The average F1 score achieved by each model on all 3 test sets (excluding test set 4) followed the trend from Figure 18, where it was raising for the models using increasingly big patch sizes, peaked with the models using (300 x 300) px patch and plateaued for the rest of the models. A trend of improved F1 score with the raising dataset size as well as raising model input size could be also observed, with exception of models E-. The winter images performed badly, as it was expected. Therefore, values on test set 4, if the evaluation was performed at all, were generally treated as outliers and were not considered in the analysis.

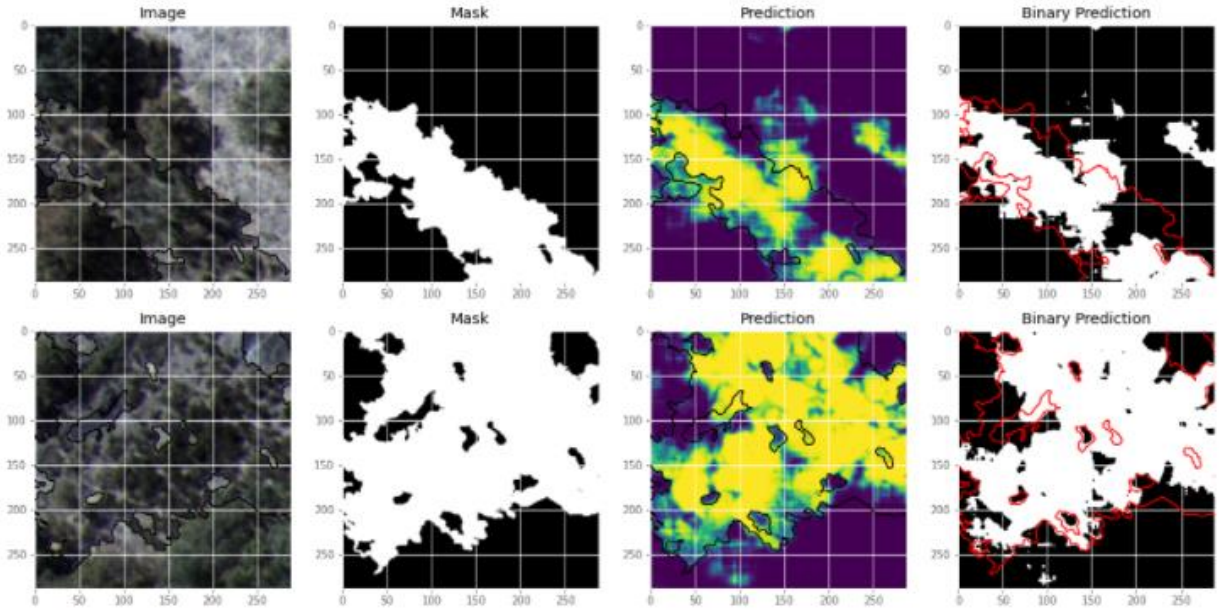


Figure 34 Visual example of the performance of C-3808_288x288 model on test set 1. From left: image patch, binary mask, prediction and binary prediction with threshold 0.5. Accuracy = 0.89, Precision = 0.84, Recall = 0.72, F1 score = 0.77.

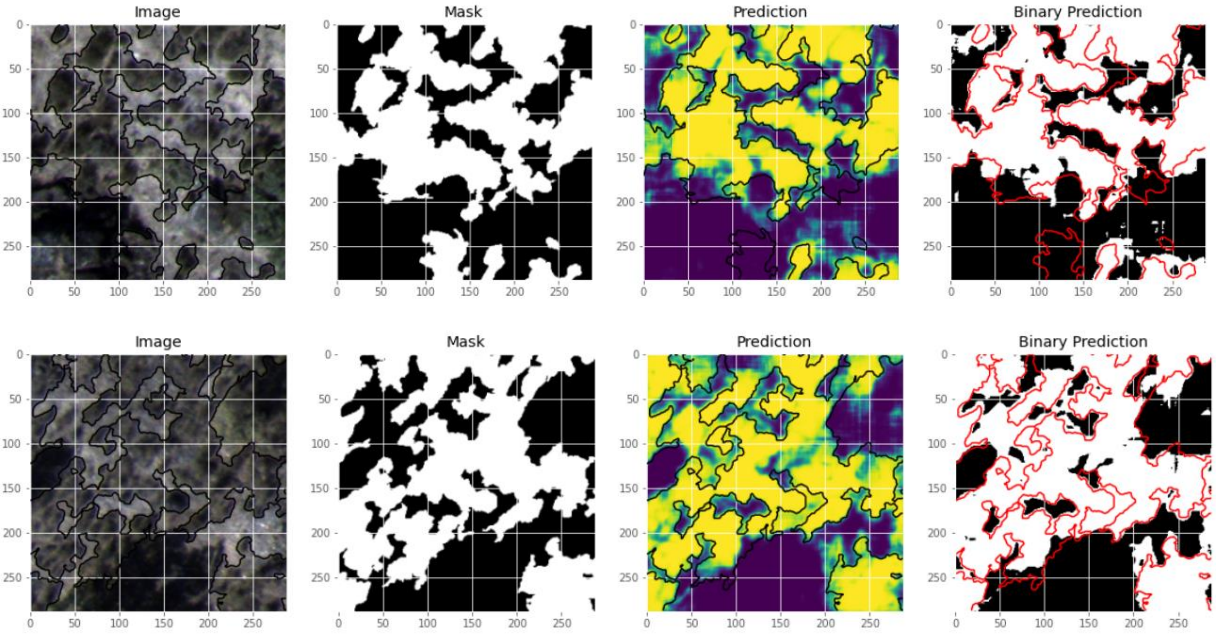


Figure 35 Visual example of the performance of C-3808_288x288 model on test set 2. From left: image patch, binary mask, prediction and binary prediction with threshold 0.5. Accuracy = 0.74, Precision = 0.76, Recall = 0.76, F1 score = 0.76.

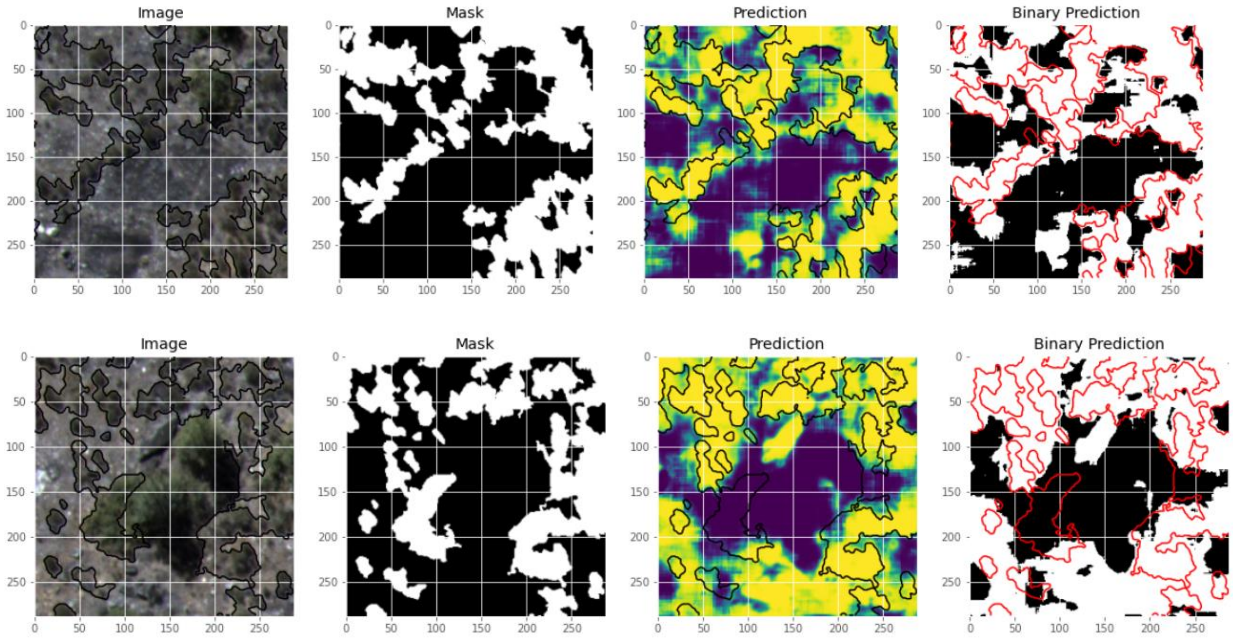


Figure 36 Visual example of the performance of C-3808_288x288 model on test set 3. From left: image patch, binary mask, prediction and binary prediction with threshold 0.5. Accuracy = 0.67, Precision = 0.56, Recall = 0.71, F1 score = 0.62.

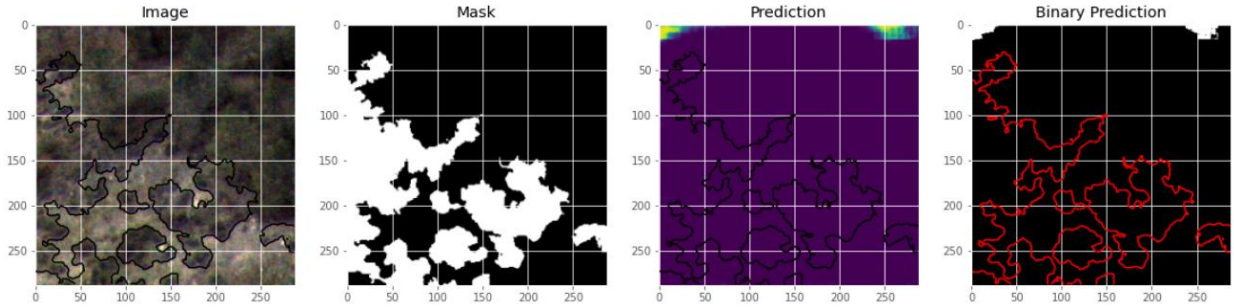


Figure 37 Visual example of the performance of C-3808_288x288 model on test set 4. From left: image patch, binary mask, prediction and binary prediction with threshold 0.5. Accuracy = 0.63, Precision = 0.12, Recall = 0.00, F1 score = 0.00.

For Balancing the datasets, the highest average F1 score (0.71) was achieved by C-1664_144x144(45%) and this was also the model with the smallest difference between the validation (0.78) and test F1 score. The highest performance was generally achieved with test set 2 and the second highest with test set 1. The best F1 score achieved on test set 3 was 0.68 with the model C-1664_144x144(45%).

Regarding the Hyperparameter tuning, raising the number of filters brought exactly opposite results as the validation. Model C-1664_144x144(filters=32) performed worst on all three test sets (average F1 score = 0.60), while C-1664_144x144(filters=64) showed the highest average F1 score 0.70 for all test sets (more specifically 0.64, 0.73 and 0.72, respectively for test sets 1, 2 and 3). C-1664_144x144(filters=16) performed similarly with test sets 1 and 2 (0.66 and 0.74, respectively) and

only got worse with test set 3 (0.62). Test set 2 had the highest average F1 score (0.71), while test sets 1 and 3 performed similarly on average (0.62 and 0.63, respectively). In case of a dropout rate, the testing followed the validation evaluation trend and models using higher dropouts were gradually performing worse on all test sets. Best average F1 score was once again achieved by test set 2 (0.73), while the second best by test set 3 (0.61). Finally, smaller batch sizes 15 and 32 showed better average F1 scores (0.70 and 0.71, respectively, as opposed to 0.64 for batch size 50). Test set 2 with the highest average F1 score (0.74) was followed by test set 1 and 3 (0.66 and 0.65, respectively).

The complete summary of the results can be found in Table A 8 Table A 5in the Appendix.

6 Discussion

In this section I analyze the results of individual experiments.

6.1 The baseline: sub-dataset A

From the results in 5.1it is clear that the overall performance improves with the raising volume of augmented data, with the highest F1 score (0.68) when training with the biggest sub-dataset (3832 patches). This is not a surprise – the model has more examples to learn from and the data augmentation aids in encoding more invariance, making the learning more robust. The performance seems to be converging, but there is still a potential for further improvement by data extension through augmentation. A thing to notice here is a different percentage representation of the shrubs class in each validation dataset – 15.65%, 22.77% and 22.95% (from Table 11, Table 15 and Table 16, respectively; more detail can be found in Table A 3 in the Appendix). This is caused by splitting of the dataset into train and validation sets, as well as applying random data augmentation techniques, which may lead to different class distribution. Even though the differences are not big, similar stochastic situation can be expected in the training set, where it can directly influence results. More controlled way of generating new data and subsequent splitting them into train and validation sets could eventuate into more reproducible results.

Another important thing that resurfaced was significantly better performance of the tree class, that outperformed shrubs even with the non-augmented dataset (832 patches). Similar results were achieved using the classification tool in QGIS (3.3). This can be ascribed to the fact, that trees are a much more balanced class without any artificial adjustments to the data, with 48.58% pixel representation across the dataset, while shrubs only account for 20.99%, but the most valid reason seems to be the fact, that trees are simply more distinct to other classes and suffer less from high intra- and low inter-class variance.

However, even the most augmented sub-dataset A performs poorly in comparison to other sub-datasets. This might be caused by the fact, that the patches are too small for the model to recognize any overall pattern. There are assumably too many patches consisting of only a part of one object and not capturing enough of the background context, preventing the model from learning sufficient amount of the inter-class correlations and leading to worse performance. Moreover, contrary to the other sub-datasets the patches here are up-scaled (from 100 to 128 px), which can bring more blur into them, making it even more difficult to see relevant patterns. On the top of that, scales larger than one don’t incur much performance improvement because there is no additional information gained, and instead they occupy more space in GPU (Zheng et al., 2016).

Accuracy is only slightly worse comparing to the accuracy achieved by the model in the TGS competition (0.85 and 0.92, respectively). This can be caused by the fact that the shapes in the original dataset are simpler, therefore this approach may not be the right one for vegetation classes.

Generally speaking, the performance, especially so for the shrubs class, is not extraordinary. Small patch sizes probably fail to capture enough of spatial detail and fine-grained boundaries between the class and the background.

6.2 Sub-datasets B-E: patch size, scale and data augmentation

The results presented in section 5.2 evidently support data augmentation as a means of improving the performance. I compared F1 scores of models trained on 808-, 1658- and 3808-instance datasets in Figure 18. The biggest differences among models trained on the same sub-dataset were between 808- and 1658-instance datasets for the smaller patch-sized sub-datasets B and C. This is likely caused by insufficient amount of information provided by only 808 instances in a dataset, whereas doubling this amount to 1658 seems to be already satisfactory. On the other hand, expanding the datasets further to 3808 does not anymore yield such big differences and the F1 score begins to plateau. The improvements are a couple of percent, while the training time generally doubles. This is normal, for example Hussain et al. (2019) gained only 4% improvement in accuracy with increasing the size of the dataset tenfold. Obviously, this also affects time greatly, a variable that had a big importance in this study. Thereby I didn't proceed with experiments when the improvements in performance started to be too disproportionate to the time increments. Hence, this means that the presented results are not necessarily the best achievable ones and I would recommend further testing in this direction. For the augmentation, I used a high number of deformation types with generally high probability and magnitude, much higher than in case of e.g. Kattenborn et al. (2020), which could have corrupted the performance. Random data augmentation could be treated as another hyperparameter, since changing the deformation types (Reina et al., 2020) or their argument values could yield different classification results. Excluding deformation technique all together could potentially yield better results, since according to Sauder (2014) CNNs are not robust to deformations. An augmentation technique that I didn't use and could reduce the misclassification problem is a multi-scale augmentation (Li et al., 2018; P. Zhang et al., 2018). Training the model on a data at multiple scales can be useful for the classification of land cover with high spatial heterogeneity (P. Zhang et al., 2018) and could be worth of experimenting with. Another interesting thing to notice is that the representation of shrubs can vary for even 14% among datasets (e.g. D-808_128x128 and D-1658_400x400, details in Table A 6 Table A 5in the Appendix. Impacts of this were not further explored but it could be another topic for a future investigation.

Patch size seems to have no significant impact on the performance of models using the biggest 3808-instance dataset, all the values oscillate around 0.83. The situation is similar for models using (300 x 300) px patches and bigger training on 1658-instance sub-datasets, that also have equal F1 score (0.80). This indicates that increasing the patch size is not justified anymore once the amount of samples in the training dataset is sufficient, and increasing it beyond (300 x 300) px also doesn't improve the

classification results, similarly as in case of Hung et al. (2014), instead it increases time and computational requirements. This same trend can be observed also in Figure 20. From Figure 18 can be also noticed that model D-3808_128x128 has the highest F1 score on average and the flattest trendline. It seems, that (400 x 400) px patches capture optimal amount of context and at the same time are capable of maintaining the most relevant information when down-scaled to (128 x 128) px (32% of the original size). Patch size is not the only hyperparameter introduced by tiling. The tile (in my case the patch) overlap may be another important factor to consider because it can affect the quality of predictions in border regions of the patches cropped from the original image. Cropping can lead to the boundary effect that introduces a bias in these regions due to zero padding, an artifact that is more pronounced when patches are stitched together afterwards. Solutions to this problem are averaging overlapping layers (Huang et al., 2018) or cropping away the edges of the output labelled image (Reina et al., 2020; Ronneberger et al., 2015; Stoian et al., 2019), which is considered to be a superior approach (Huang et al., 2018). The work of Iglovikov et al. (2017) went even further on this and added a cropping layer directly to the output layers of the network that do the cropping automatically and losses on boundary artifacts are not back-propagated. This thesis doesn't address this issue but exploring the effect of this phenomenon on the particular data used in this study could be a good next step in the further experimentations.

The harmful impact of down-scaling to a coarser resolution can be best seen in Figure 20. Degrading the imagery into too small resolutions significantly hampers the ability to detect structures and textures. The closer is the size of the rescaled patch to the original size, the higher F1 score is achieved. This is especially important in cases where the size of the objects of interest is already small, because with down-scaling they become even harder to see (Audebert et al., 2018). Moreover, for shrub detection not only the context is important, but also the information present within the class. Excessive downscaling thus indeed leads to the loss of relevant information (Reina et al., 2020; W. Zhang et al., 2019), however it is an interesting technique for shortening the training time (Audebert et al., 2018), with the scale 1:2 as a good tradeoff between the little drop in performance but a shorter training time (Rakhlin et al., 2018). Certainly, the best results were brought by the three most time demanding experiments – C-3808_288x288, D-1658_400x400 and E-1658_496x496, all with a scale ~1:1. All yielded F1 score 0.90, but took 50, 46 and 60 hours to train, respectively. At the same time, F1 score dropped just for a few percent for the models using the same sub-datasets with the scale ~1:2, but the training times were much less (9.2, 9 and 15.8 hours, respectively). Besides, qualitatively they didn't bring much value, as was showed in Figure 17. Different patch size doesn't seem to have much of an impact on 1:1 and 1:2 rescaling for these sub-datasets. Patches (300 x 300) px seem to provide sufficient amount of spatial detail and an enlargement for additional 100-200 px brings marginal improvements.

The configuration of pre-processing techniques yielding the best results depends on the problem and on the object of interest (Guirado et al., 2017). Finding an optimal set of these methods for this particular problem would require further exhaustive research. One pre-processing method that could be helpful with this task is elimination of a background based on a color threshold in grayscale images (Guirado et al., 2017). In overall, I believe that one of the most important factors affecting the performance was

inaccurate labelling. There were cases when the model correctly classified pixels which I labeled incorrectly, similarly as for Rakhlin et al. (2018), thus, the performance might have been in fact better than indicated when compared to the reference data. Nevertheless, high quality labels remain to be one of the central elements of image classification success. The most problematic to classify correctly were generally the border regions of shrubs. This could be explained by U-Net's lack of geometric accuracy, which leaves the edges of the classes not sharp (Stoian et al., 2019), but also by the fact that labelling of these parts was very challenging and therefore it could happen that they were labelled incorrectly from the very beginning. This issue was vanishing with the growing size of sub-datasets, patch dimensions and model input size.

6.3 Balancing the datasets

This experiment showed dropping performance in all metrics but recall with the increasing proportion of the shrub class representation in the dataset (Table 24). According to the precision trend, the model's ability to distinguish shrubs from other land cover classes got worse, while the improvement in recall might have come from the fact that there were simply more shrub pixels to spot in the dataset. Moreover, the distribution of shrub pixels is shifted more to the right (Figure A 8), further favoring this class in the recall results.

Model C-1664_144x144(45%), using the most balanced dataset, was expected to perform the best, considering some other studies (Wei & Jr, 2013), but the opposite was true. The learning process seems to be not robust enough, possibly from two reasons: 1.) The used sub-dataset was created with heavy augmentation from only 15 patches, which significantly lowers down the representativeness of already small data sample. This small sub-dataset surely doesn't effectively cover the different arrangements of land cover in such a diverse heterogeneous scene as is present in my data, which means the model's recognizing abilities may not be sufficient with new data. The under-sampling had definitely led to the loss of important information. 2.) The suitability of this kind of approach depends on the problem and whether under- (or over-) representation is a normal characteristic of the target class in real life situations, too. Because the distribution of shrubs is random in reality, respecting this feature in the training data may yield better results.

6.4 Hyperparameter tuning

Similarly to the case of P. Zhang et al. (2018), adding more filters improves the performance only until certain point after which it starts to drop, disagreeing with the general notion that deeper networks achieve better accuracies (Li et al., 2018). In my case, there was a little gain especially in recall (+0.03) and subsequently F1 score (+0.02) when doubling the initial number of filters from 16 to 32, but both metrics experienced drop of -0.13 and -0.08, respectively, after further doubling of the filters to 64. Accuracy and precision remained more or less constant. Using more filters made the network deeper

and more complicated, which was probably not necessary for my kind of data or brought too many weights for the amount of available data that could cause overfitting. The model might have learnt more complex features that are actually shared with other classes such as trees, which created a confusion in the classification. But because trees are a majority class in the dataset – almost half, while shrubs represent only about ¼ (Table 3), the model decided to favor the tree class in ambiguous situations because it simply saw more pixels with the particular feature being labeled as trees, or more precisely 'non-shrubs'. This theory would explain the drop in recall between using 32 and 64 filters. The F1 score of the best performing model C-1664_144x144(filters=32) was 0.84 but took 10 hours to train, while the model C-1664_144x144(filters=16) achieved 0.82 F1 score in half the time.

Excluding the outlier recall (and therefore F1 score) of a model C-1664_144x144(45%), the metrics are generally getting worse with the increase of the dropout rate. The performance worsens considerably already with 0.5 dropout rate. Increasing the number of dropped neurons to 75% doesn't lead to a strong decline in accuracy and precision but recall jumps to 1. Given the low value of accuracy and precision, the model is evidently labeling most of the pixels as shrubs, because it was not able to learn enough of important features. With a difficult task that includes a landcover as complex as the one present by the data used in this thesis, the more neurons facilitate the learning process the better. Therefore, using high dropout rates might not be a reasonable choice in problems like this one.

The batch size in my case didn't have much of an impact on the results. Probably the range was not sufficiently big and more distinct results would be obtained with larger differences in the batch sizes. Some (Masters & Luschi, 2018) reported the best results when using batch size as small as 2 or 4, while others (Iglovikov et al., 2017) favored batch sizes as big as 128. It seems that this hyperparameter, as many others, is also depending on many factors such as the type of a problem or if e.g. a transfer learning was applied, as in the case of Iglovikov et al. (2017). While there might be a room for further exploration of a batch size tuning, it would be reliant on computational resources. Moreover, taking into account that the batch size of 32 is generally recommended as an optimum, and that no significant changes were recorded with batch size changes in my experiments, I would not recommend going deeper with the topic in this particular case. However, an interesting way forward could be exploring whether larger batch sizes are superior to larger patch sizes in terms of improving the performance, as in case of Iglovikov et al. (2017), who traded the receptive field size in favor of a larger batch size. They used a small training set (25 images) with quite different images, which is a peculiarity shared with this thesis, but our studies diverge in terms of the image contents and the usage of a transfer learning, and the outcomes may therefore differ.

There are many other hyperparameters that could be further explored in order to improve the classification results, but the optimal model generally depends on the used data[35], so it is not only important to tune the hyperparameters, but also to choose them diligently, since some of them may have significant impact on the results while others can have almost none.

[35] https://jakevdp.github.io/PythonDataScienceHandbook/05.03-hyperparameters-and-model-validation.html

6.5 Test data

In overall, models performed worse on the new data. The drop in performance was expected since some small overfitting on the training set is common and the validation data came from the same image. However, a lower decrease was expected. One possible explanation could be that I didn't split one big dataset into train, validation and test sets, but I used data from entirely separate images for testing (excluding test set 1). The best average performance of all the experiments was achieved by test set 2 (0.70), while test sets 1 and 3 performed equally (0.61, see Table A 9 in the Appendix). Worse performance of a test set 1 could simply have been caused by statistics – the test sets were not equal in terms of size, because the test set 1 only consisted of one patch while all of the other test sets consisted of two. An unlucky pick of a patch, i.e. having for example a different distribution than in train and validation data, even if coming from the same image could have caused difficulties during classification, causing bigger differences between validation and test results. Looking at the Figure A 3, the area selected as a test patch could really be perceived as a bit visually different from the rest, containing a big amount of bare land with a characteristic brightness as well as very dense shrubs on a skewed surface. I also tested this theory by switching the test and validation sets; i.e. I used a test set as a validation set and vice-versa. The validation set (patches from the same dataset as the training ones) performed better whether it was used for validation or testing. These results confirmed that my choice of a patch for the test set 1 was an unlucky pick somehow different from train and validation sets. Patches in the test set 3 came from different images taken in a different year, that was most likely the single biggest factor worsening the performance. The best evaluations were on test set 2 because these images were taken on the same day as the training ones and two patches were used, decreasing the risk of the chosen test data to be too area specific. The image from which training and validation patches were derived didn't supply various enough data and the patches were an unrepresentative sample of the shrub patterns in the area, e.g. an area full of shrubs that would be perpendicular to the drone (like in the second image in a test set 3, see Figure A 6 in the Appendix) is missing entirely in the training data, moreover all the patches are too close to each other and overlapping, which may be further decreasing the variety. A more robust model could have been obtained by training on a larger dataset of patches derived from different images, taken on different days and in different years, that would improve the representativeness of the data and could have increased the variety of features to learn. The winter images are too different to be extrapolated from the summer data. A separate model is necessary.

Higher testing performances (0.76 to 0.77) were generally achieved by models using bigger patch sizes and model input dimensions, in accordance with the validation results from the Sub-datasets B-E: patch size, scale and data augmentation section. Along with the data augmentation, these three methods proved to increase the classification performance, fulfilling the research objective no. III (1.4) of this thesis. The Hyperparameter tuning didn't bring any significant improvements in the performance, neither

for validation, nor for test sets. Generally, the gap between validation and test scores are relative to the data, selected metrics and models[36].

Comparing my test results to some other CNNs trained from scratch (Hussain et al., 2019), and considering the complexity of the land cover of the presented area and the main objective of this work, the presented methods proved to be feasible even for an ordinary user and the results satisfactory, especially considering that there is a potential for further improvements in performance, meeting the research objective no. IV (1.4).

[36] https://machinelearningmastery.com/the-model-performance-mismatch-problem/

7 Conclusions and future work

This thesis explored the potential of detecting the target vegetation type in a complex heterogeneous landscape with U-Net. Shrubs are wild plants with different shapes, sizes and distribution patterns. The difficulty of this task was increased further with the fact that the data contained more than species of shrubs scattered in the forest area. Shrubs are of a priority interest in terms of fire risk in dry Mediterranean regions and mapping them can serve as a basis for better informed land management and reduction of the forest fire hazard. This work consisted of two main parts: creating and manually labelling the datasets and developing a method to increase a detection accuracy using a U-Net neural network. The impact of data augmentation, tiling, rescaling, balancing the dataset and hyperparameter tuning (number of filters, dropout rate and batch size) was explored in this regard.

First, the task of creating a dataset was addressed. Labelling is an intensive and time demanding manual labor, because of which I was able to create only a very small training dataset of 13 (800 x 800) px tiles and later I relied on a heavy data augmentation. Distinguishing shrubs from other vegetation types in remote sensing images is a challenge for non-experts as well as for automatic detection methods. Particularly problematic were the border parts of the vegetation and skewed border regions of the fish-eye images. Consistency was problematic especially with labelling shadows when it was sometimes difficult to distinguish between the shadow as a separate class or as a part of another class. The presence of labeling inaccuracies is therefore certain. I consider a very little labelled data that were not sufficient for learning of all the important features from scratch and incorrect labels as the biggest weaknesses of this work. Using bigger datasets with patches derived from several images taken during multiple flights and employing them in the training and validation process could have a significant positive effect on the results. Besides this, classification accuracy depends on how distinguishable the shrubs are from the surrounding. Good timing regarding phenological stage (which was perfect in my case), lower flying altitudes or using a higher resolution sensor to obtain more detail could help to alleviate this issue. However, a big intra-class and small inter-class variance in spectral signatures of presented vegetation classes will remain a challenge.

Factors with the most significant impact on the detection capability were data augmentation, patch size and model input dimensions. The biggest datasets containing 3808 samples yielded the highest F1 score of around 0.83 for all models. Further enlargement of the datasets could increase the performance even more. Modest amount of labeled data resulted in heavy data augmentation. I believe that further improvements could be achieved mainly by relying on more labelled data from spatially independent samples, rather than on heavy augmentation that can lead to overfitting and in case of very little data it can mask the problem of low representativeness, that can be revealed with new data. Larger volumes of data would also give more space to create more proportional datasets without too much augmentation. However, augmentation is still an extremely useful tool and for the sake of the future research it would be useful to find the best configuration of the augmentation techniques suitable for these data. Regarding the patch size, high spatial heterogeneity of the vegetation in my study area made the task particularly demanding and simply using the window size bigger than the physical size of the

target plants was not enough. Rather than thinking in terms of individual plants, it was necessary to consider the structures many plants formed together. Increasing the patch size was benefitting the performance up to (300 x 300) px. Patches bigger than that didn't bring any significant improvement in performance, instead it increased the training time and computational demands. Tiling also brings along the boundary effect. This was not addressed in my study but assessing predictions in the border regions of this particular data would be also a good topic to investigate in the future, especially so for the data stitching. The last helpful technique was rescaling of the patches. Degrading the image resolution leads to a loss of information, but this technique was valuable for cutting down the training time, which was an important variable for me. Scale 1:2 significantly decreased the training time and didn't lead to a dramatic drop in performance.

With the increasingly balanced dataset, the F1 score was dropping for validation but increasing for testing. The highest F1 score averaged for all three test sets was 0.71. More experiments would be needed to draw a meaningful conclusions in this area, ideally with more diverse train and test data and without under-sampling and subsequent heavy augmentation of the training datasets. Experimenting with three hyperparameters: the number of filters, dropout rate and batch size didn't bring any significant benefits. The next steps could focus on finding an optimal configuration of pre-processing methods as well as hyperparameters for this particular task. The availability of R and NIR bands could be also explored more, as well as employing transfer learning.

This thesis demonstrated the capacity of U-Net for mapping the irregular shrub cover, presented methods improving the classification results and provided recommendations for a future research. The work has a potential to serve as an information tool for land planning and grazing management and could be also modified and repurposed to map other vegetation types, such as trees, or to be used as e.g. a forest inventory tool.

8 References

Ahmed, B., & Noman, M. A. A. (2015). Land cover classification for satellite images based on normalization technique and Artificial Neural Network. *2015 International Conference on Computer and Information Engineering (ICCIE)*, 138–141. https://doi.org/10.1109/CCIE.2015.7399300

Ashapure, A., Jung, J., Chang, A., Oh, S., Maeda, M., & Landivar, J. (2019). A Comparative Study of RGB and Multispectral Sensor-Based Cotton Canopy Cover Modelling Using Multi-Temporal UAS Data. *Remote Sensing*, *11*(23), 2757. https://doi.org/10.3390/rs11232757

Audebert, N., Le Saux, B., & Lefèvre, S. (2018). Beyond RGB: Very high resolution urban remote sensing with multimodal deep networks. *ISPRS Journal of Photogrammetry and Remote Sensing*, *140*, 20–32. https://doi.org/10.1016/j.isprsjprs.2017.11.011

Ayhan, B., & Kwan, C. (2020). Tree, Shrub, and Grass Classification Using Only RGB Images. *Remote Sensing*, *12*(8), 1333. https://doi.org/10.3390/rs12081333

Badrinarayanan, V., Kendall, A., & Cipolla, R. (2016). SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation. *ArXiv:1511.00561 [Cs]*. http://arxiv.org/abs/1511.00561

Baena, S., Moat, J., Whaley, O., & Boyd, D. S. (2017). Identifying species from the air: UAVs and the very high resolution challenge for plant conservation. *PLOS ONE*, *12*(11), e0188714. https://doi.org/10.1371/journal.pone.0188714

Bao, H. (2019). Investigations of the Influences of a CNN's Receptive Field on Segmentation of Subnuclei of Bilateral Amygdalae. *ArXiv*.

Bengio, Y. (2012). Practical recommendations for gradient-based training of deep architectures. *ArXiv:1206.5533 [Cs]*. http://arxiv.org/abs/1206.5533

Brandt, M., Tucker, C. J., Kariryaa, A., Rasmussen, K., Abel, C., Small, J., Chave, J., Rasmussen, L. V., Hiernaux, P., Diouf, A. A., Kergoat, L., Mertz, O., Igel, C., Gieseke, F., Schöning, J., Li, S., Melocik, K., Meyer, J., Sinno, S., … Fensholt, R. (2020). An unexpectedly large count of trees in the West African Sahara and Sahel. *Nature*, *587*(7832), 78–82. https://doi.org/10.1038/s41586-020-2824-5

Buscombe, D., & Ritchie, A. C. (2018). *2 Landscape classification with deep neural networks.* 23.

Cao, J., Leng, W., Liu, K., Liu, L., He, Z., & Zhu, Y. (2018). Object-Based Mangrove Species Classification Using Unmanned Aerial Vehicle Hyperspectral Images and Digital Surface Models. *Remote Sensing*, *10*(1), 89. https://doi.org/10.3390/rs10010089

Chen, L.-C., Papandreou, G., Kokkinos, I., Murphy, K., & Yuille, A. L. (2017). DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs. *ArXiv:1606.00915 [Cs]*. http://arxiv.org/abs/1606.00915

Chen, Y., Wang, K., Liao, X., Qian, Y., Wang, Q., Yuan, Z., & Heng, P.-A. (2019). Channel-Unet: A Spatial Channel-Wise Convolutional Neural Network for Liver and Tumors Segmentation. *Frontiers in Genetics*, *10*, 1110. https://doi.org/10.3389/fgene.2019.01110

Congedo, L. (2016). Semi-automatic classification plugin documentation. *Release*, *4*(0.1), 29.

Csillik, O., Cherbini, J., Johnson, R., Lyons, A., & Kelly, M. (2018). Identification of Citrus Trees from Unmanned Aerial Vehicle Imagery Using Convolutional Neural Networks. *Drones*, *2*(4), 39. https://doi.org/10.3390/drones2040039

Diakogiannis, F. I., Waldner, F., Caccetta, P., & Wu, C. (2020). ResUNet-a: A deep learning framework for semantic segmentation of remotely sensed data. *ISPRS Journal of Photogrammetry and Remote Sensing*, *162*, 94–114. https://doi.org/10.1016/j.isprsjprs.2020.01.013

Flood, N., Watson, F., & Collett, L. (2019). Using a U-net convolutional neural network to map woody vegetation extent from high resolution satellite imagery across Queensland, Australia. *International Journal of Applied Earth Observation and Geoinformation*, *82*, 101897. https://doi.org/10.1016/j.jag.2019.101897

Fourure, D., Emonet, R., Fromont, E., Muselet, D., Tremeau, A., & Wolf, C. (2017). Residual Conv-Deconv Grid Network for Semantic Segmentation. *ArXiv:1707.07958 [Cs]*. http://arxiv.org/abs/1707.07958

Fricker, G. A., Ventura, J. D., Wolf, J. A., North, M. P., Davis, F. W., & Franklin, J. (2019). A Convolutional Neural Network Classifier Identifies Tree Species in Mixed-Conifer Forest from Hyperspectral Imagery. *Remote Sensing*, *11*(19), 2326. https://doi.org/10.3390/rs11192326

Fröhlich, B., Bach, E., Walde, I., Hese, S., Schmullius, C., & Denzler, J. (2013). Land cover classification of satellite images using contextual information. *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences*, *II-3/W1*, 1–6. https://doi.org/10.5194/isprsannals-II-3-W1-1-2013

Gaetano, R., Ienco, D., Ose, K., & Cresson, R. (2018). A Two-Branch CNN Architecture for Land Cover Classification of PAN and MS Imagery. *Remote Sensing*, *10*(11), 1746. https://doi.org/10.3390/rs10111746

Garg, L., Shukla, P., Singh, S., Bajpai, V., & Yadav, U. (2019). Land Use Land Cover Classification from Satellite Imagery using mUnet: A Modified Unet Architecture: *Proceedings of the 14th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications*, 359–365. https://doi.org/10.5220/0007370603590365

Gbodjo, Y., Ienco, D., Leroux, L., Interdonato, R., Gaetano, R., & Ndao, B. (2020). Object-Based Multi-Temporal and Multi-Source Land Cover Mapping Leveraging Hierarchical Class Relationships. *Remote Sensing*, *12*, 2814. https://doi.org/10.3390/rs12172814

Getzin, S., Wiegand, K., & Schöning, I. (2012). Assessing biodiversity in forests using very high-resolution images and unmanned aerial vehicles. *Methods in Ecology and Evolution*, *3*(2), 397–404. https://doi.org/10.1111/j.2041-210X.2011.00158.x

Girshick, R. (2015). Fast R-CNN. *ArXiv:1504.08083 [Cs]*. http://arxiv.org/abs/1504.08083

GO - SILVPAST - Terraprima. (n.d.). Retrieved September 19, 2020, from https://www.terraprima.pt/pt/projecto/23

Goyal, P., Dollar, P., Girshick, R., Noordhuis, P., Wesolowski, L., Kyrola, A., Tulloch, A., Jia, Y., & He, K. (n.d.). *Accurate, Large Minibatch SGD: Training ImageNet in 1 Hour*. 12.

Graham, S., Chen, H., Gamper, J., Dou, Q., Heng, P.-A., Snead, D., Tsang, Y. W., & Rajpoot, N. (2019). MILD-Net: Minimal information loss dilated network for gland instance segmentation in colon histology images. *Medical Image Analysis*, *52*, 199–211. https://doi.org/10.1016/j.media.2018.12.001

Guirado, E., Tabik, S., Alcaraz-Segura, D., Cabello, J., & Herrera, F. (2017). Deep-Learning Convolutional Neural Networks for scattered shrub detection with Google Earth Imagery. *ArXiv:1706.00917 [Cs]*. http://arxiv.org/abs/1706.00917

Hellesen, T., & Matikainen, L. (2013). An Object-Based Approach for Mapping Shrub and Tree Cover on Grassland Habitats by Use of LiDAR and CIR Orthoimages. *Remote Sensing*, *5*(2), 558–583. https://doi.org/10.3390/rs5020558

Hu, F., Xia, G.-S., Hu, J., & Zhang, L. (2015). Transferring Deep Convolutional Neural Networks for the Scene Classification of High-Resolution Remote Sensing Imagery. *Remote Sensing*, *7*(11), 14680–14707. https://doi.org/10.3390/rs71114680

Huang, B., Reichman, D., Collins, L., Bradbury, K., & Malof, J. M. (2018). *Tiling and Stitching Segmentation Output for Remote Sensing: Basic Challenges and Recommendations*. Undefined. /paper/Tiling-and-Stitching-Segmentation-Output-for-Remote-Huang-Reichman/57dc15369c0be30023320cc53a3c17c7c9ee6737

Hung, C., Xu, Z., & Sukkarieh, S. (2014). Feature Learning Based Approach for Weed Classification Using High Resolution Aerial Images from a Digital Camera Mounted on a UAV. *Remote Sensing*, *6*(12), 12037–12054. https://doi.org/10.3390/rs61212037

Hussain, M., Bird, J. J., & Faria, D. R. (2019). A Study on CNN Transfer Learning for Image Classification. In A. Lotfi, H. Bouchachia, A. Gegov, C. Langensiepen, & M. McGinnity (Eds.), *Advances in Computational Intelligence Systems* (Vol. 840, pp. 191–202). Springer International Publishing. https://doi.org/10.1007/978-3-319-97982-3_16

Ibtehaz, N., & Rahman, M. S. (2020). MultiResUNet: Rethinking the U-Net Architecture for Multimodal Biomedical Image Segmentation. *Neural Networks*, *121*, 74–87. https://doi.org/10.1016/j.neunet.2019.08.025

Iglovikov, V., Mushinskiy, S., & Osin, V. (2017). Satellite Imagery Feature Detection using Deep Convolutional Neural Network: A Kaggle Competition. *ArXiv:1706.06169 [Cs]*. http://arxiv.org/abs/1706.06169

Iglovikov, V., & Shvets, A. (2018). TernausNet: U-Net with VGG11 Encoder Pre-Trained on ImageNet for Image Segmentation. *ArXiv:1801.05746 [Cs]*. http://arxiv.org/abs/1801.05746

Ioffe, S., & Szegedy, C. (2015). Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. *ArXiv:1502.03167 [Cs]*. http://arxiv.org/abs/1502.03167

Isensee, F., Jaeger, P., Kohl, S., Petersen, J., & Maier-Hein, K. (2020). nnU-Net: A self-configuring method for deep learning-based biomedical image segmentation. *Nature Methods*, 1–9. https://doi.org/10.1038/s41592-020-01008-z

Jiang, J., Hu, Y., Liu, C.-J., Halpenny, D., Hellmann, M. D., Deasy, J. O., Mageras, G., & Veeraraghavan, H. (2019). Multiple Resolution Residually Connected Feature Streams For Automatic Lung Tumor Segmentation From CT Images. *IEEE Transactions on Medical Imaging*, *38*(1), 134–144. https://doi.org/10.1109/TMI.2018.2857800

Kattenborn, T., Eichel, J., Wiser, S., Burrows, L., Fassnacht, F. E., & Schmidtlein, S. (2020). Convolutional Neural Networks accurately predict cover fractions of plant species and communities in Unmanned Aerial Vehicle imagery. *Remote Sensing in Ecology and Conservation*, rse2.146. https://doi.org/10.1002/rse2.146

Kerle, N., Nex, F., Gerke, M., Duarte, D., & Vetrivel, A. (2019). UAV-Based Structural Damage Mapping: A Review. *ISPRS International Journal of Geo-Information*, *9*(1), 14. https://doi.org/10.3390/ijgi9010014

Keskar, N. S., Mudigere, D., Nocedal, J., Smelyanskiy, M., & Tang, P. T. P. (2017). On Large-Batch Training for Deep Learning: Generalization Gap and Sharp Minima. *ArXiv:1609.04836 [Cs, Math]*. http://arxiv.org/abs/1609.04836

Kim, M., Warner, T. A., Madden, M., & Atkinson, D. S. (2011). Multi-scale GEOBIA with very high spatial resolution digital aerial imagery: Scale, texture and image objects. *International Journal of Remote Sensing*, *32*(10), 2825–2850. https://doi.org/10.1080/01431161003745608

Kinaneva, D., Hristov, G., Raychev, J., & Zahariev, P. (2019). Early Forest Fire Detection Using Drones and Artificial Intelligence. *2019 42nd International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, 1060–1065. https://doi.org/10.23919/MIPRO.2019.8756696

Kingma, D. P., & Ba, J. (2017). Adam: A Method for Stochastic Optimization. *ArXiv:1412.6980 [Cs]*. http://arxiv.org/abs/1412.6980

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2017). ImageNet classification with deep convolutional neural networks. *Communications of the ACM*, *60*(6), 84–90. https://doi.org/10.1145/3065386

Kussul, N., Lavreniuk, M., Skakun, S., & Shelestov, A. (2017). Deep Learning Classification of Land Cover and Crop Types Using Remote Sensing Data. *IEEE Geoscience and Remote Sensing Letters*, *14*(5), 778–782. https://doi.org/10.1109/LGRS.2017.2681128

Langford, Z. L., Kumar, J., Hoffman, F. M., Breen, A. L., & Iversen, C. M. (2019). Arctic Vegetation Mapping Using Unsupervised Training Datasets and Convolutional Neural Networks. *Remote Sensing*, *11*(1), 69. https://doi.org/10.3390/rs11010069

Längkvist, M., Kiselev, A., Alirezaie, M., & Loutfi, A. (2016). Classification and Segmentation of Satellite Orthoimagery Using Convolutional Neural Networks. *Remote Sensing*, *8*(4), 329. https://doi.org/10.3390/rs8040329

Li, R., Liu, W., Yang, L., Sun, S., Hu, W., Zhang, F., & Li, W. (2018). DeepUNet: A Deep Fully Convolutional Network for Pixel-Level Sea-Land Segmentation. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, *11*(11), 3954–3962. https://doi.org/10.1109/JSTARS.2018.2833382

Litjens, G., Kooi, T., Bejnordi, B. E., Setio, A. A. A., Ciompi, F., Ghafoorian, M., van der Laak, J. A. W. M., van Ginneken, B., & Sánchez, C. I. (2017). A survey on deep learning in medical image analysis. *Medical Image Analysis*, *42*, 60–88. https://doi.org/10.1016/j.media.2017.07.005

Long, J., Shelhamer, E., & Darrell, T. (2015). Fully Convolutional Networks for Semantic Segmentation. *ArXiv:1411.4038 [Cs]*. http://arxiv.org/abs/1411.4038

Lopatin, J., Dolos, K., Kattenborn, T., & Fassnacht, F. E. (2019). How canopy shadow affects invasive plant species classification in high spatial resolution remote sensing. *Remote Sensing in Ecology and Conservation*, *5*(4), 302–317. https://doi.org/10.1002/rse2.109

Lopatin, J., Fassnacht, F. E., Kattenborn, T., & Schmidtlein, S. (2017). Mapping plant species in mixed grassland communities using close range imaging spectroscopy. *Remote Sensing of Environment*, *201*, 12–23. https://doi.org/10.1016/j.rse.2017.08.031

Lovreglio, R., Meddour-Sahar, O., & Leone, V. (2014). Goat grazing as a wildfire prevention tool: A basic review. *IForest - Biogeosciences and Forestry*, *7*(4), 260–268. https://doi.org/10.3832/ifor1112-007

Mahdianpari, M., Salehi, B., Rezaee, M., Mohammadimanesh, F., & Zhang, Y. (2018). Very Deep Convolutional Neural Networks for Complex Land Cover Mapping Using Multispectral Remote Sensing Imagery. *Remote Sensing*, *10*(7), 1119. https://doi.org/10.3390/rs10071119

Makantasis, K., Karantzalos, K., Doulamis, A., & Doulamis, N. (2015). Deep supervised learning for hyperspectral data classification through convolutional neural networks. *2015 IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*, 4959–4962. https://doi.org/10.1109/IGARSS.2015.7326945

Malenovský, Z., Lucieer, A., King, D. H., Turnbull, J. D., & Robinson, S. A. (2017). Unmanned aircraft system advances health mapping of fragile polar vegetation. *Methods in Ecology and Evolution*, *8*(12), 1842–1857. https://doi.org/10.1111/2041-210X.12833

Maltezos, E., Doulamis, N., Doulamis, A., & Ioannidis, C. (2017). Deep convolutional neural networks for building extraction from orthoimages and dense image matching point clouds. *Journal of Applied Remote Sensing*, *11*(4), 042620. https://doi.org/10.1117/1.JRS.11.042620

Mangewa, L. J., Ndakidemi, P. A., & Munishi, L. K. (2019). Integrating UAV Technology in an Ecological Monitoring System for Community Wildlife Management Areas in Tanzania. *Sustainability*, *11*(21), 6116. https://doi.org/10.3390/su11216116

Marques, A., Martins, I. S., Kastner, T., Plutzar, C., Theurl, M. C., Eisenmenger, N., Huijbregts, M. A. J., Wood, R., Stadler, K., Bruckner, M., Canelas, J., Hilbers, J. P., Tukker, A., Erb, K., & Pereira, H. M. (2019). Increasing impacts of land use on biodiversity and carbon sequestration driven by population and economic growth. *Nature Ecology & Evolution*, *3*(4), 628–637. https://doi.org/10.1038/s41559-019-0824-3

Masters, D., & Luschi, C. (2018). Revisiting Small Batch Training for Deep Neural Networks. *ArXiv:1804.07612 [Cs, Stat]*. http://arxiv.org/abs/1804.07612

Matese, A., Toscano, P., Di Gennaro, S. F., Genesio, L., Vaccari, F. P., Primicerio, J., Belli, C., Zaldei, A., Bianconi, R., & Gioli, B. (2015). Intercomparison of UAV, Aircraft and Satellite Remote Sensing

Platforms for Precision Viticulture. *Remote Sensing*, *7*(3), 2971–2990. https://doi.org/10.3390/rs70302971

Müllerová, J., Brůna, J., Bartaloš, T., Dvořák, P., Vítková, M., & Pyšek, P. (2017). Timing Is Important: Unmanned Aircraft vs. Satellite Imagery in Plant Invasion Monitoring. *Frontiers in Plant Science*, *8*, 887. https://doi.org/10.3389/fpls.2017.00887

Nivaggioli, A., & Randrianarivo, H. (2019). Weakly Supervised Semantic Segmentation of Satellite Images. *ArXiv:1904.03983 [Cs]*. http://arxiv.org/abs/1904.03983

Nogueira, K., Miranda, W. O., & Santos, J. A. D. (2015). Improving Spatial Feature Representation from Aerial Scenes by Using Convolutional Networks. *2015 28th SIBGRAPI Conference on Graphics, Patterns and Images*, 289–296. https://doi.org/10.1109/SIBGRAPI.2015.39

Paisitkriangkrai, S., Sherrah, J., Janney, P., & Van Den Hengel, A. (2016). *Semantic labeling of aerial and satellite imagery*. https://doi.org/10.1109/JSTARS.2016.2582921

Paneque-Gálvez, J., McCall, M. K., Napoletano, B. M., Wich, S. A., & Koh, L. P. (2014). Small Drones for Community-Based Forest Monitoring: An Assessment of Their Feasibility and Potential in Tropical Areas. *Forests*, *5*(6), 1481–1507. https://doi.org/10.3390/f5061481

Pérez-Rodríguez, L. A., Quintano, C., Marcos, E., Suarez-Seoane, S., Calvo, L., & Fernández-Manso, A. (2020). Evaluation of Prescribed Fires from Unmanned Aerial Vehicles (UAVs) Imagery and Machine Learning Algorithms. *Remote Sensing*, *12*(8), 1295. https://doi.org/10.3390/rs12081295

Pohlen, T., Hermans, A., Mathias, M., & Leibe, B. (2016). Full-Resolution Residual Networks for Semantic Segmentation in Street Scenes. *ArXiv:1611.08323 [Cs]*. http://arxiv.org/abs/1611.08323

Proença, V., & Teixeira, C. M. G. L. (2019). Beyond meat: Ecological functions of livestock. *Science*, *366*(6468), 962–962. https://doi.org/10.1126/science.aaz7084

Rakhlin, A., Davydow, A., & Nikolenko, S. (2018). Land Cover Classification from Satellite Imagery with U-Net and Lovász-Softmax Loss. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 257–2574. https://doi.org/10.1109/CVPRW.2018.00048

Ramanath, A., Muthusrinivasan, S., Xie, Y., Shekhar, S., & Ramachandra, B. (2019). NDVI Versus CNN Features in Deep Learning for Land Cover Clasification of Aerial Images. *IGARSS 2019 - 2019 IEEE International Geoscience and Remote Sensing Symposium*, 6483–6486. https://doi.org/10.1109/IGARSS.2019.8900165

Reina, G. A., Panchumarthy, R., Thakur, S. P., Bastidas, A., & Bakas, S. (2020). Systematic Evaluation of Image Tiling Adverse Effects on Deep Learning Semantic Segmentation. *Frontiers in Neuroscience*, *14*, 65. https://doi.org/10.3389/fnins.2020.00065

Rey Benayas, J. (2007). Abandonment of agricultural land: An overview of drivers and consequences. *CAB Reviews: Perspectives in Agriculture, Veterinary Science, Nutrition and Natural Resources*, *2*(057). https://doi.org/10.1079/PAVSNNR20072057

Ripple, W. J., Newsome, T. M., Wolf, C., Dirzo, R., Everatt, K. T., Galetti, M., Hayward, M. W., Kerley, G. I. H., Levi, T., Lindsey, P. A., Macdonald, D. W., Malhi, Y., Painter, L. E., Sandom, C. J., Terborgh, J., & Van Valkenburgh, B. (2015). Collapse of the world's largest herbivores. *Science Advances*, *1*(4), e1400103. https://doi.org/10.1126/sciadv.1400103

Ronneberger, O., Fischer, P., & Brox, T. (2015). U-Net: Convolutional Networks for Biomedical Image Segmentation. *ArXiv:1505.04597 [Cs]*. http://arxiv.org/abs/1505.04597

Sankey, T., Donager, J., McVay, J., & Sankey, J. B. (2017). UAV lidar and hyperspectral fusion for forest monitoring in the southwestern USA. *Remote Sensing of Environment*, *195*, 30–43. https://doi.org/10.1016/j.rse.2017.04.007

Sauder, N. (2014). *Encoded Invariance in Convolutional Neural Networks*. /paper/Encoded-Invariance-in-Convolutional-Neural-Networks-Sauder/02ac327bd8dfd5df31529ce9a9bc87def9e85848

Scott, G. J., England, M. R., Starms, W. A., Marcum, R. A., & Davis, C. H. (2017). Training Deep Convolutional Neural Networks for Land–Cover Classification of High-Resolution Imagery. *IEEE Geoscience and Remote Sensing Letters*, *14*(4), 549–553. https://doi.org/10.1109/LGRS.2017.2657778

Sermanet, P., Eigen, D., Zhang, X., Mathieu, M., Fergus, R., & LeCun, Y. (2014). OverFeat: Integrated Recognition, Localization and Detection using Convolutional Networks. *ArXiv:1312.6229 [Cs]*. http://arxiv.org/abs/1312.6229

Shaban, M., Awan, R., Fraz, M. M., Azam, A., Snead, D., & Rajpoot, N. M. (2019). Context-Aware Convolutional Neural Network for Grading of Colorectal Cancer Histology Images. *ArXiv:1907.09478 [Cs, Eess, Stat]*. http://arxiv.org/abs/1907.09478

Simões, J. P. G. (2019). *Relatório de Estágio Quinta da França – Terra Prima Sociedade Agrícola, lda*. Universidade de Lisboa Instituto Superior de Agronomia.

Sladojevic, S., Arsenovic, M., Anderla, A., Culibrk, D., & Stefanovic, D. (2016, June 22). *Deep Neural Networks Based Recognition of Plant Diseases by Leaf Image Classification* [Research Article]. Computational Intelligence and Neuroscience; Hindawi. https://doi.org/10.1155/2016/3289801

Stoian, A., Poulain, V., Inglada, J., Poughon, V., & Derksen, D. (2019). Land Cover Maps Production with High Resolution Satellite Image Time Series and Convolutional Neural Networks: Adaptations and Limits for Operational Systems. *Remote Sensing*, *11*(17), 1986. https://doi.org/10.3390/rs11171986

Terraprima -Sociedade Agrícola Lda. (2012). *Plano de Gestão Florestal da Quinta da França – Resumo Público*. https://www.terraprima.pt/en/projecto/15

Treml, M., Arjona-Medina, J., Unterthiner, T., Durgesh, R., Friedmann, F., Schuberth, P., Mayr, A., Heusel, M., Hofmarcher, M., Widrich, M., Nessler, B., & Hochreiter, S. (2016). *Speeding up Semantic Segmentation for Autonomous Driving*. https://openreview.net/forum?id=S1uHiFyyg

Turner, W. (2014). Sensing biodiversity. *Science*, *346*(6207), 301–302. https://doi.org/10.1126/science.1256014

Ulmas, P., & Liiv, I. (2020). Segmentation of Satellite Imagery using U-Net Models for Land Cover Classification. *ArXiv:2003.02899 [Cs]*. http://arxiv.org/abs/2003.02899

Vanjare, A., S N, O., & Senthilnath, J. (2014). Satellite Image Processing for Land Use and Land Cover Mapping. *International Journal of Image, Graphics and Signal Processing*, *6*, 18–28. https://doi.org/10.5815/ijigsp.2014.10.03

Volpi, M., & Tuia, D. (2017). Dense Semantic Labeling of Subdecimeter Resolution Images With Convolutional Neural Networks. *IEEE Transactions on Geoscience and Remote Sensing*, *55*(2), 881–893. https://doi.org/10.1109/TGRS.2016.2616585

Wang, S., Chen, W., Xie, S. M., Azzari, G., & Lobell, D. B. (2020). Weakly Supervised Deep Learning for Segmentation of Remote Sensing Imagery. *Remote Sensing*, *12*(2), 207. https://doi.org/10.3390/rs12020207

Watanabe, S., Sumi, K., & Ise, T. (2018). Automatic vegetation identification in Google Earth images using a convolutional neural network: A case study for Japanese bamboo forests. *BioRxiv*, 351643. https://doi.org/10.1101/351643

Wei, Q., & Jr, R. L. D. (2013). The Role of Balanced Training and Testing Data Sets for Binary Classifiers in Bioinformatics. *PLOS ONE*, *8*(7), e67863. https://doi.org/10.1371/journal.pone.0067863

Wen, D., Huang, X., Liu, H., Liao, W., & Zhang, L. (2017). Semantic classification of urban trees using very high resolution satellite imagery. *IEEE Journal of Selected Topics in Earth Observation and Remote Sensing*, *10*(4), 1413–1424. https://doi.org/10.1109/JSTARS.2016.2645798

Willett, W., Rockström, J., Loken, B., Springmann, M., Lang, T., Vermeulen, S., Garnett, T., Tilman, D., DeClerck, F., Wood, A., Jonell, M., Clark, M., Gordon, L. J., Fanzo, J., Hawkes, C., Zurayk, R., Rivera, J. A., Vries, W. D., Sibanda, L. M., … Murray, C. J. L. (2019). Food in the Anthropocene: The EAT–Lancet Commission on healthy diets from sustainable food systems. *The Lancet*, *393*(10170), 447–492. https://doi.org/10.1016/S0140-6736(18)31788-4

Yu, C., Wang, J., Peng, C., Gao, C., Yu, G., & Sang, N. (2018). Learning a Discriminative Feature Network for Semantic Segmentation. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 1857–1866. https://doi.org/10.1109/CVPR.2018.00199

Yue, J., Zhao, W., Mao, S., & Liu, H. (2015). Spectral–spatial classification of hyperspectral images using deep convolutional neural networks. *Remote Sensing Letters*, *6*(6), 468–477. https://doi.org/10.1080/2150704X.2015.1047045

Zhang, F., Du, B., & Zhang, L. (2015). Saliency-Guided Unsupervised Feature Learning for Scene Classification. *IEEE Transactions on Geoscience and Remote Sensing*, *53*(4), 2175–2184. https://doi.org/10.1109/TGRS.2014.2357078

Zhang, P., Ke, Y., Zhang, Z., Wang, M., Li, P., & Zhang, S. (2018). Urban Land Use and Land Cover Classification Using Novel Deep Learning Models Based on High Spatial Resolution Satellite Imagery. *Sensors*, *18*(11), 3717. https://doi.org/10.3390/s18113717

Zhang, W., Tang, P., & Zhao, L. (2019). Remote Sensing Image Scene Classification Using CNN-CapsNet. *Remote Sensing*, *11*(5), 494. https://doi.org/10.3390/rs11050494

Zheng, L., Zhao, Y., Wang, S., Wang, J., & Tian, Q. (2016). Good Practice in CNN Feature Transfer. *ArXiv:1604.00133 [Cs]*. http://arxiv.org/abs/1604.00133

Zhou, Q., Yang, W., Gao, G., Ou, W., Lu, H., Chen, J., & Latecki, L. J. (2019). Multi-scale deep context convolutional neural networks for semantic segmentation. *World Wide Web*, *22*(2), 555–570. https://doi.org/10.1007/s11280-018-0556-3

Zhou, Z., Siddiquee, M. M. R., Tajbakhsh, N., & Liang, J. (2020). UNet++: Redesigning Skip Connections to Exploit Multiscale Features in Image Segmentation. *ArXiv:1912.05074 [Cs, Eess]*. http://arxiv.org/abs/1912.05074

Appendix

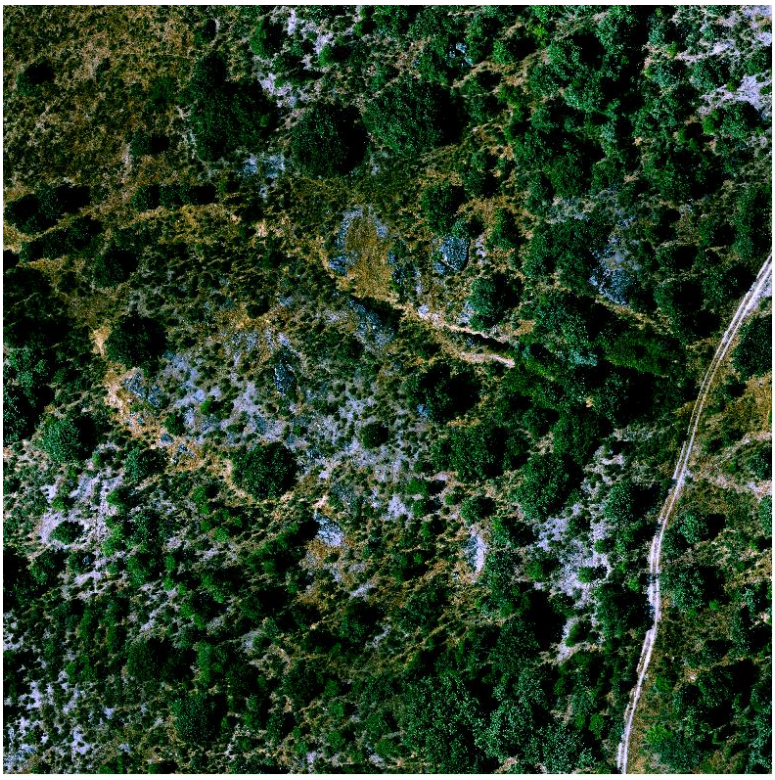


Figure A 1 Easier recognition of rocks and bare soil. RGB image, July 2016. Settings: Red band: Band 1, Min: 150, Max: 255. Green band: Band 2, Min: 100, Max: 255. Blue band: Band 3, Min: 100, Max: 200.

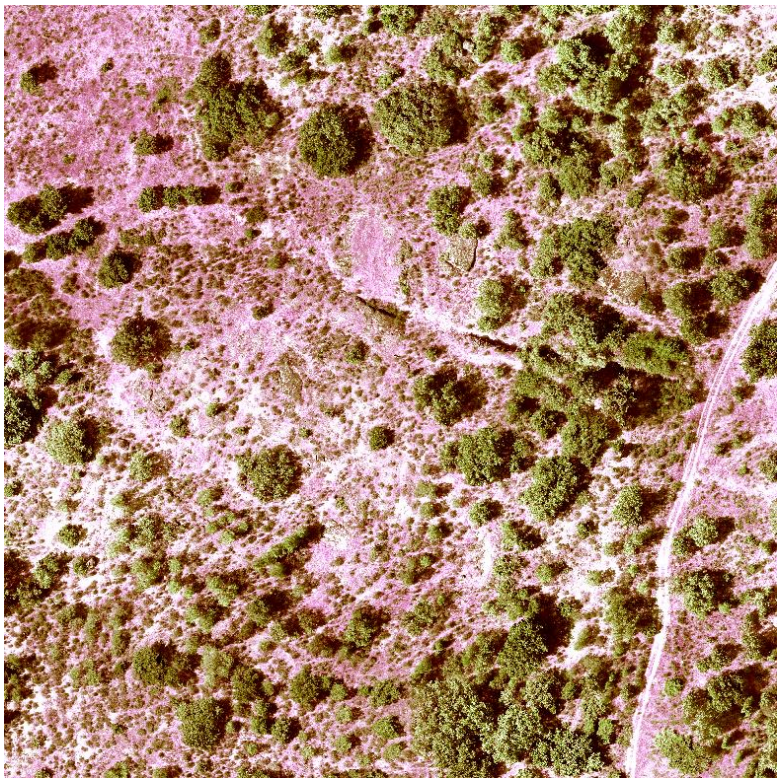


Figure A 2 Easier recognition of vegetation. RGB image, July 2016. Settings: Red band: Band 1, Min: 0, Max: 200. Green band: Band 2, Min: 50, Max: 200. Blue band: Band 1, Min: 0, Max: 200.

Table A 1 The most impactful parameters for classification results and data visualization

Purpose	Tool	Specification	Conclusion
Visualization for interpretation	Band rendering	Interchanging bands. Adjusting band intensities. Applying Singleband pseudocolor: color ramps, interpolation types, no. of classes.	Due to distinct reflectance of materials, different colour composites can highlight specific land surface features, e.g. bare soil, vegetation, etc. Some of the results can be found in Figure A 1and Figure A 2Table A 5 in the Appendix.
Better classification result	Region Growing Algorithm: Distance	An interval defining the maximum spectral distance between the seed pixel and the surrounding pixels (Congedo, 2016).	Spectral Distance in my experiments varied from 0.005 to 0.08. For easily distinguishable objects, e.g. non-shaded trees, the value could be higher, up to 0.08 (in radiometry units), whereas for the most problematic groups (rocks and shrubs) the interval was 0.005-0.01.
	Minimum Distance	The Euclidean distance between spectral signatures of image pixels and training spectral signatures. If the distance is greater than threshold value, pixels are unclassified.	This classification algorithm yielded better results than Spectral Angle Mapping (it was not possible to use Maximum Likelihood. This problem was most likely related to the pre-processing of study images, that possibly contained NoData values or alpha channels). I used default setting (0).
	Land Cover Signature Classification (LCS)	A classification that can be used as alternative or in combination with the classification algorithm. Pixels belonging to two or more different classes are classified as Class overlap with raster value = -1000 and are left unclassified or are classified according to an additional classification algorithm (Congedo, 2016)	I used LCS in combination with Minimum Distance classification algorithm. Classification results were significantly worse than when using only Minimum Distance, however, it was useful for assessing the proportion of pixels classified as belonging to more than 1 class. According to the generated confusion matrix, 12.5% of pixels (1752807 out of 13980121) were classified as belonging to more than 1 class and 84% were misclassified in this specific trial.
	Number of Regions of interest (ROIs)	I tested 5 – 50 ROIs per class	After certain threshold, higher number of ROI samples does not yield significantly better classification results. I concluded this threshold to be approximately 15 ROI samples per class for my study images.

Table A 2 Names of images (taken in August 2019), tiles produced from them and tiles chosen for labelling

Image number	Image name (identical for TIFF and PNG)	Tiles names	Tiles selected for labelling
1	2019_0830_105101_057	img1tile1… img1tile30	img1tile2.png, img1tile3.png, img1tile4.png, img1tile5.png, img1tile8.png, img1tile11.png, img1tile12.png, img1tile13.png, img1tile14.png, img1tile19.png, img1tile20.png, img1tile21.png, img1tile27.png img1tile26.png – for testing
2	2019_0830_105110_059	img2tile1… img2tile30	-
3	2019_0830_105118_061	img3tile1… img3tile30	img3tile8.png – for testing
4	2019_0830_105127_063	img4tile1… img4tile30	-
5	2019_0830_105135_065	img5tile1… img5tile30	-
6	2019_0830_105144_067	img6tile1… img6tile30	-
7	2019_0830_105153_069	img7tile1… img7tile30	-
8	2019_0830_105315_083	img8tile1… img8tile30	-
9	2019_0830_105324_085	img9tile1… img9tile30	-
10	2019_0830_105332_087	img10tile1… img10tile30	-
11	2019_0830_105340_089	img11tile1… img11tile30	-
12	2019_0830_105349_091	img12tile1… img12tile30	-
13	2019_0830_105357_093	img13tile1… img13tile30	-
14	2019_0830_105405_095	img14tile1… img14tile30	Img14tile20.png – for testing
15	2019_0830_105519_107	img15tile1… img15tile30	-
16	2019_0830_105527_109	img16tile1… img16tile30	-
17	2019_0830_105535_111	img17tile1… img17tile30	-
18	2019_0830_105544_113	img18tile1… img18tile30	-
19	2019_0830_105552_115	img19tile1… img19tile30	-
20	2019_0830_105601_117	img20tile1… img20tile30	-
21	2019_0830_105609_119	img21tile1… img21tile30	-

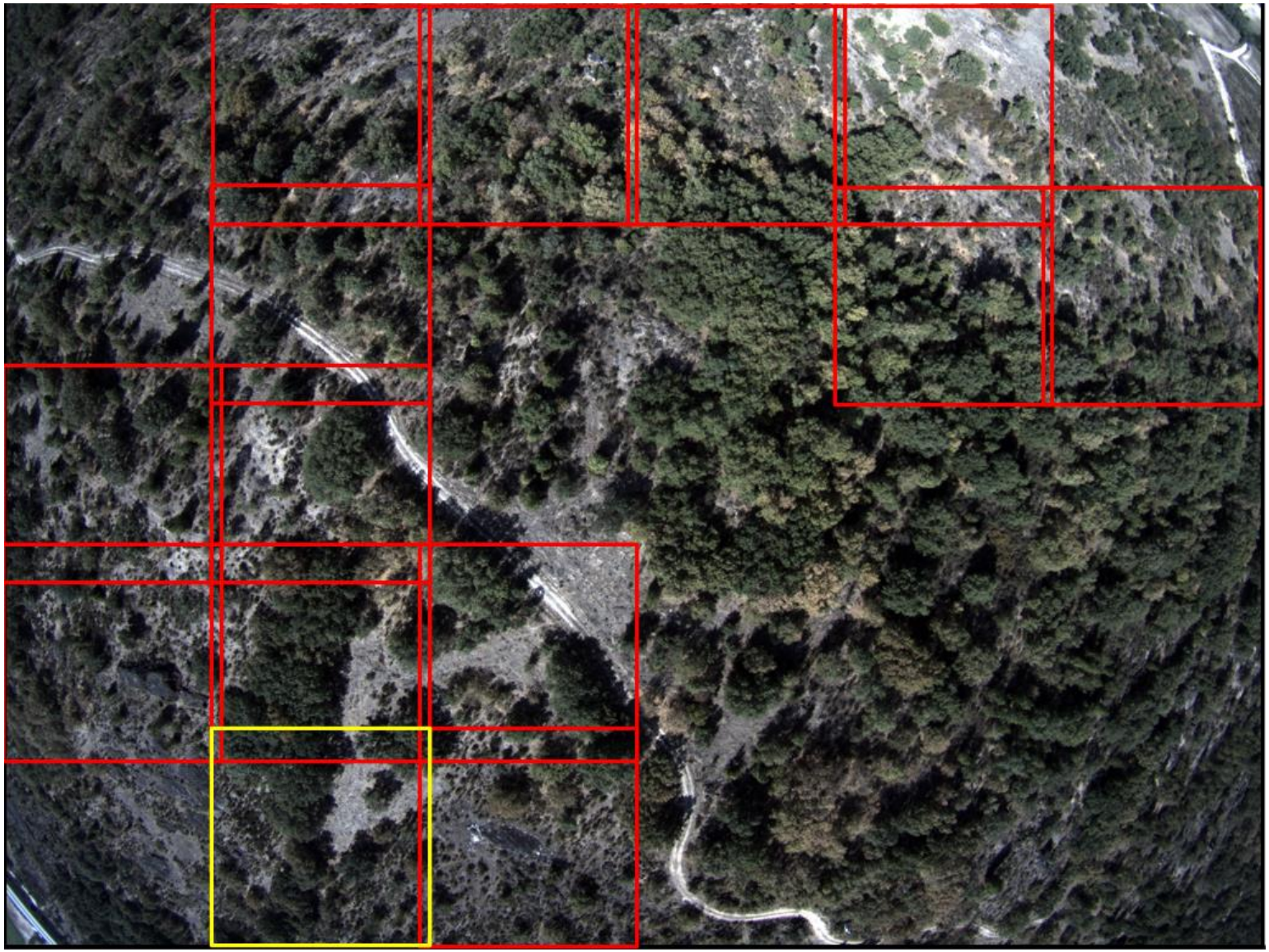


Figure A 3 Image *2019_0830_105101_057* and tiles selected for labelling. Red frame signifies training (and validation) tiles. Yellow is a testing tile for test set 1.

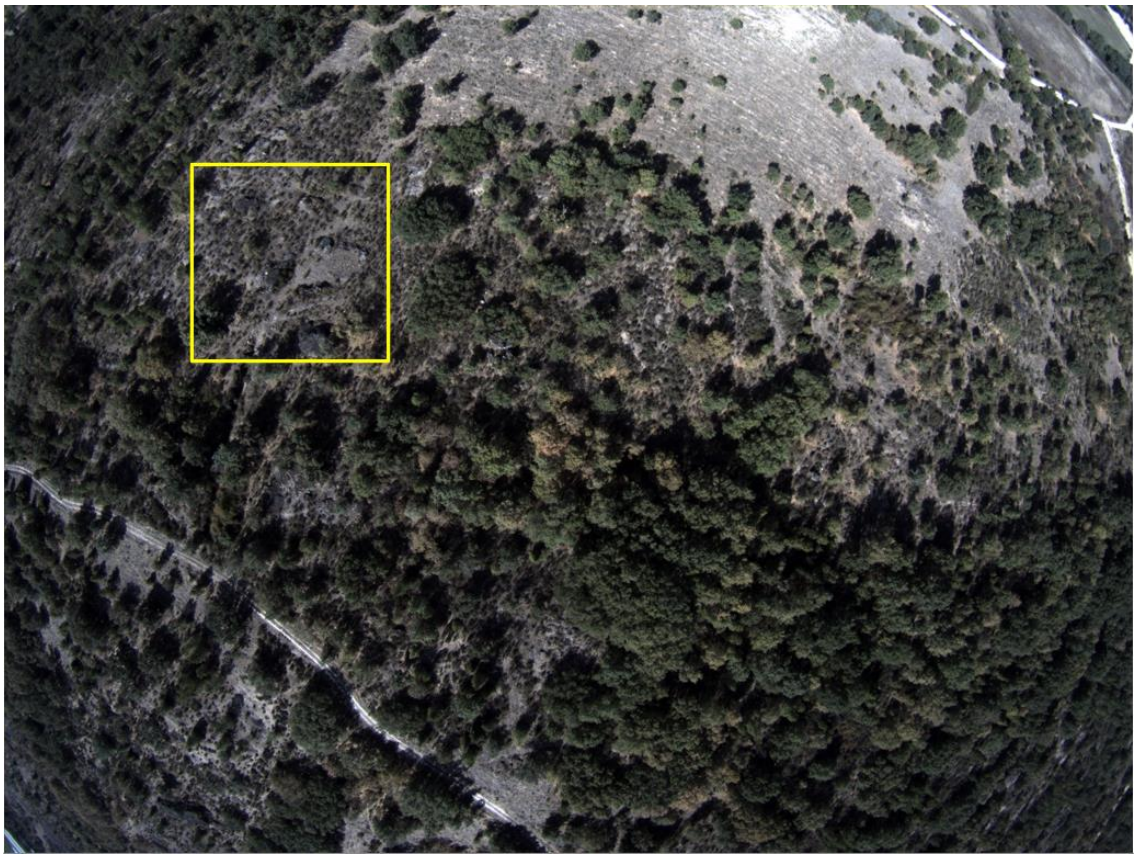


Figure A 4 Image *2019_0830_105101_061* and tile *img3tile8* selected for labelling for test set 2

Figure A 5 Image *2019_0830_105101_095* and tile *img14tile20* selected for labelling for test set 2

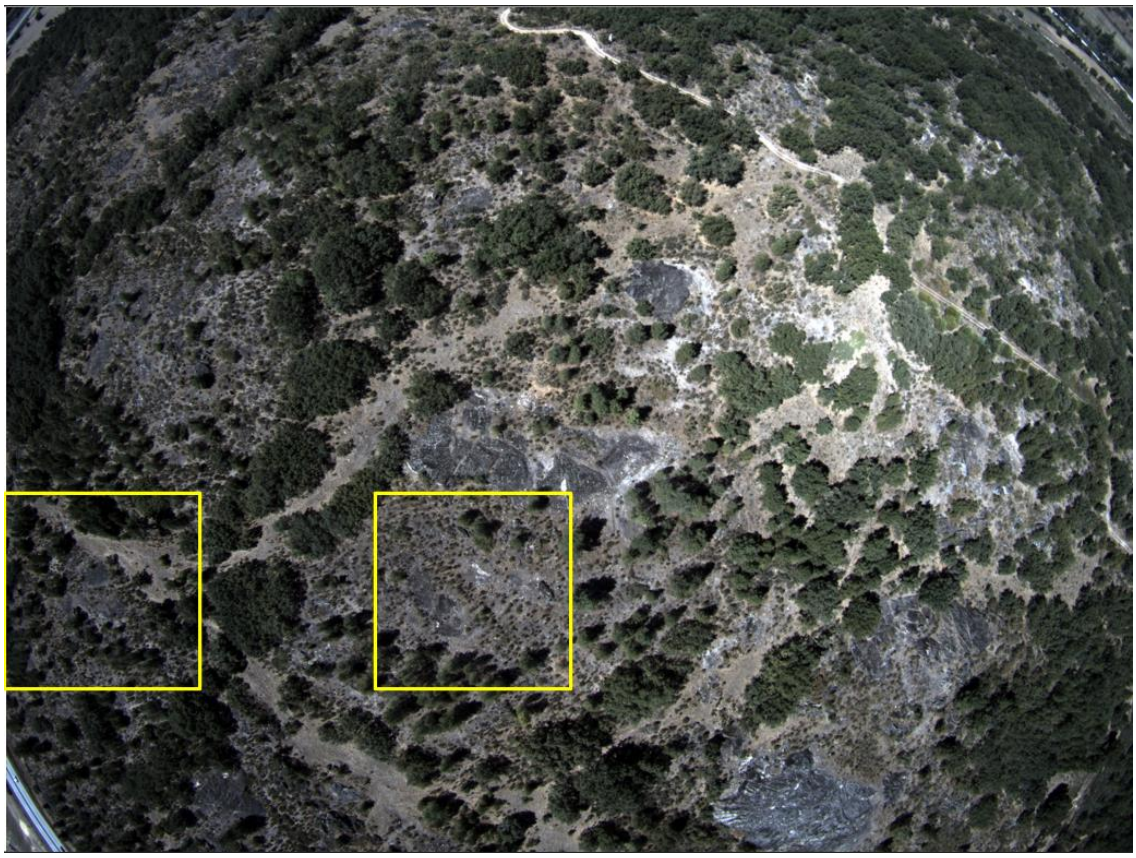


Figure A 6 Image *2020_0826_115241_057* and tiles *img1tile19* and *img1tile21* selected for labelling for test set 3

Figure A 7 Image *2019_1210_141336_069* and tiles *img1tile28* and *img1tile29* selected for labelling for test set 4

Table A 3 Sub-dataset A experiments: confusion matrices & pixel count

		BASELINE - SUB-DATASET A (100x100)px: VAL SET.							
Model input size [px]	***set size &***	***(val) set size &***	***conf.matrix***		***conf.matrix [%]***		***act.shrubs***	***act.non-shrubs***	***total pxs***	***tot shrubs % in the set***
	set pxs: (patch size)^2 * set size	***(val) set pxs: (patch size)^2 *val set size***	*TP*	*FP*	*TP [%]*	*FP [%]*	*TP+FN*	*FP+TN*	*TP+TN+FP+FN*	*(TP+FN)/ tot pxs*
			FN	*TN*	*FN [%]*	*TN [%]*				
128	**832**	**84**	**206996**	**707120**	15.040516	51.379976	215399	1160857	1376256	15.65108526
128	13631488	1376256	**8403**	**453737**	0.6105695	32.968939				
128	**1664**	**167**	**347615**	**139200**	12.704632	5.0874813	623124	2113004	2736128	22.77393455
128	27262976	2736128	**275509**	**1973804**	10.069302	72.138584				
128	**3808**	**384**	**930008**	**349334**	14.782079	5.5525144	1443692	4847764	6291456	22.94686635
128	62390272	6291456	**513684**	**4498430**	8.1647873	71.500619				

Table A 4 Sub-dataset B experiments: confusion matrices & pixel count

		SUB-DATASET B (200x200)px: VAL SET.								
Model input size [px]	***set size &***	***(val) set size &***	***conf.matrix***		***conf.matrix [%]***		***act.shrubs***	***act.non-shrubs***	***total pxs***	***tot shrubs % in the set***
	set pxs: (patch size)^2 * set size	***(val) set pxs: (patch size)^2 *val set size***	*TP*	*FP*	*TP [%]*	*FP [%]*	*TP+FN*	*FP+TN*	*TP+TN+FP+FN*	*(TP+FN)/ tot pxs*
			FN	*TN*	*FN [%]*	*TN [%]*				
128	**808**	**81**	**152449**	**38824**	11.487344	2.9254678	309925	1017179	1327104	23.35348247
128	13238272	1327104	**157476**	**978355**	11.866139	73.72105				
128	**1658**	**166**	**442282**	**47008**	16.261898	1.728398	680344	2039400	2719744	25.01500141
128	27164672	2719744	**238062**	**1992392**	8.7531032	73.256601				
192	**1658**	**166**	**1036843**	**175040**	16.943474	2.8603999	1556635	4562789	6119424	25.43760655
192	61120512	6119424	**519792**	**4387749**	8.4941328	71.701994				
128	**3808**	**381**	**1110403**	**66646**	17.788352	1.0676507	1487399	4754905	6242304	23.82772451
128	62390272	6242304	**376996**	**4688259**	6.0393726	75.104625				
192	**3808**	**381**	**2610208**	**290307**	18.584363	2.0669505	3433406	10611778	14045184	24.44543268
192	140378112	14045184	**823198**	**10321471**	5.8610695	73.487617				

Table A 5 Sub-dataset C experiments: confusion matrices & pixel count

		SUB-DATASET C (300x300)px: VAL SET.							
Model input size [px]	***set size &***	***(val) set size &***	***conf.matrix***		***conf.matrix [%]***		***act.shrubs***	***act.non-shrubs***	***total pxs***	***tot shrubs % in the set***
	set pxs: (patch size)^2 * set size	***(val) set pxs: (patch size)^2 *val set size***	*TP*	*FP*	*TP [%]*	*FP [%]*	*TP+FN*	*FP+TN*	*TP+TN+FP+FN*	*(TP+FN)/ tot pxs*
			FN	*TN*	*FN [%]*	*TN [%]*				
128	**808**	**81**	**221328**	**235880**	16.677517	17.77404	347366	979738	1327104	26.17473838
128	13238272	1327104	**126038**	**743858**	9.497221	56.051221				
128	**1664**	**167**	**554954**	**25096**	20.282458	0.9172086	813537	1922591	2736128	29.73314845
128	27262976	2736128	**258583**	**1897495**	9.4506909	69.349643				
144	**1664**	**167**	**627350**	**29334**	18.116256	0.8470905	874407	2588505	3462912	25.25062722
144	34504704	3462912	**247057**	**2559171**	7.1343713	73.902282				
128	**3808**	**381**	**1327801**	**42624**	21.271008	0.6828248	1816802	4425502	6242304	29.10467033
128	62390272	6242304	**489001**	**4382878**	7.8336621	70.212505				
144	**3808**	**381**	**1607796**	**62251**	20.350776	0.7879458	2082710	5817706	7900416	26.36202954
144	78962688	7900416	**474914**	**5755455**	6.0112531	72.850025				
288	**3808**	**381**	**6358099**	**731075**	20.119506	2.3134066	7093538	24508127	31601665	22.44672235
288	315850752	31601665	**735439**	**23777052**	2.327216	75.239871				

Table A 6 Sub-dataset D experiments: confusion matrices & pixel count

		SUB-DATASET D (400x400)px: VAL SET.								
Model input size [px]	***set size &***	***(val) set size &***	***conf.matrix***		***conf.matrix [%]***		***act.shrubs***	***act.non-shrubs***	***total pxs***	***tot shrubs % in the set***
	set pxs: (patch size)^2 * set size	***(val) set pxs: (patch size)^2 *val set size***	*TP*	*FP*	*TP [%]*	*FP [%]*	*TP+FN*	*FP+TN*	*TP+TN+FP+FN*	*(TP+FN)/ tot pxs*
			FN	*TN*	*FN [%]*	*TN [%]*				
128	**808**	**81**	**297124**	**10235**	22.388901	0.7712282	457448	869656	1327104	34.46964217
128	13238272	1327104	**160324**	**859421**	12.080741	64.75913				
128	**1658**	**166**	**565012**	**14354**	20.774455	0.5277703	845771	1873973	2719744	31.09744888
128	27164672	2719744	**280759**	**1859619**	10.322994	68.374781				
192	**1658**	**166**	**1311234**	**37863**	21.427409	0.6187347	1744661	4374763	6119424	28.51021599
192	61120512	6119424	**433427**	**4336900**	7.0828071	70.871049				
400	**1658**	**166**	**4938493**	**452075**	18.593722	1.7020895	5549927	21010075	26560002	20.89580791
400	265280000	26560002	**611434**	**20558000**	2.3020857	77.402103				
128	**3808**	**381**	**1318805**	**12700**	21.126895	0.2034505	1800891	4441413	6242304	28.84978047
128	62390272	6242304	**482086**	**4428713**	7.7228857	70.946769				

Table A 7 Sub-dataset E experiments: confusion matrices & pixel count

		SUB-DATASET E (500x500)px: VAL SET.							
Model input size [px]	***set size &***	***(val) set size &***	***conf.matrix***		***conf.matrix [%]***		***act.shrubs***	***act.non-shrubs***	***total pxs***	***tot shrubs % in the set***
	set pxs: (patch size)^2 * set size	***(val) set pxs: (patch size)^2 *val set size***	*TP*	*FP*	*TP [%]*	*FP [%]*	*TP+FN*	*FP+TN*	*TP+TN+FP+FN*	*(TP+FN)/ tot pxs*
			FN	*TN*	*FN [%]*	*TN [%]*				
128	**808**	**81**	**285**	**0**	0.0214753	0	473181	853923	1327104	35.65515589
128	13238272	1327104	**472896**	**853923**	35.633681	64.344844				
128	**1652**	**166**	**587856**	**10855**	21.614387	0.3991184	865465	1854279	2719744	31.82156115
128	27066368	2719744	**277609**	**1843424**	10.207174	67.77932				
240	**1652**	**166**	**2105657**	**53485**	22.022015	0.5593729	2700721	6860879	9561600	28.24549239
240	95155200	9561600	**595064**	**6807394**	6.2234772	71.195135				
496	**1652**	**166**	**8665087**	**374331**	21.217857	0.9166096	10117422	30721231	40838653	24.77413249
496	406418432	40838653	**1452335**	**30346900**	3.5562755	74.309258				
128	**3808**	**381**	**1454548**	**7387**	23.30146	0.1183377	2076911	4165393	6242304	33.27154525
128	62390272	6242304	**622363**	**4158006**	9.9700848	66.610117				

m

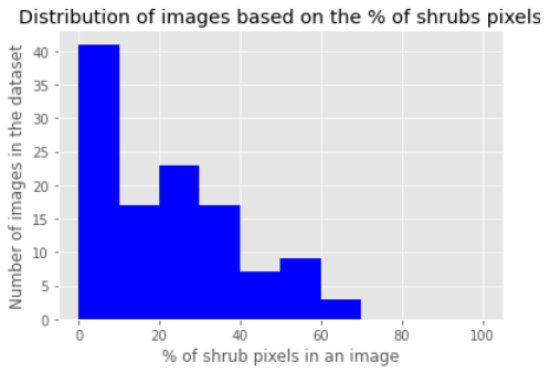
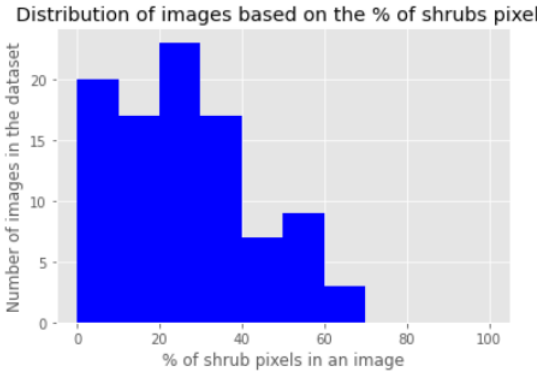
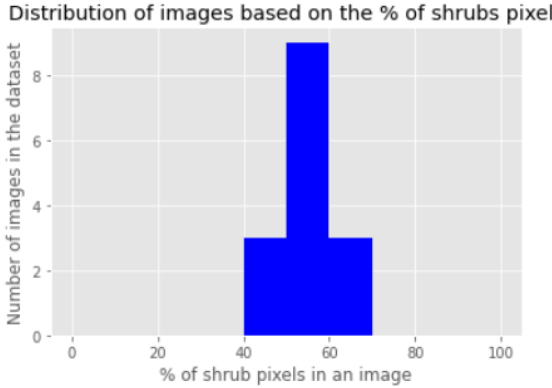
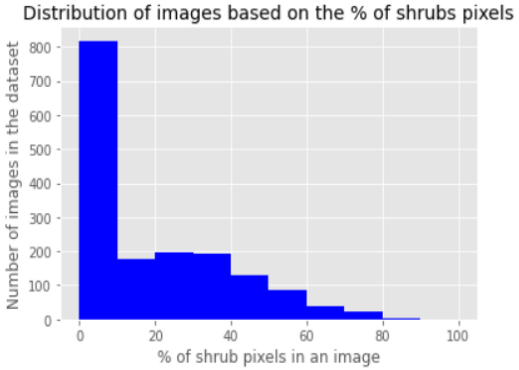
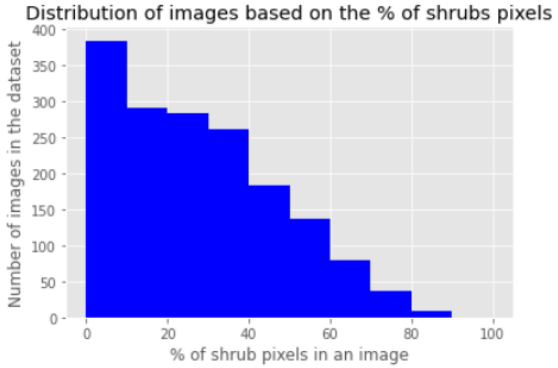
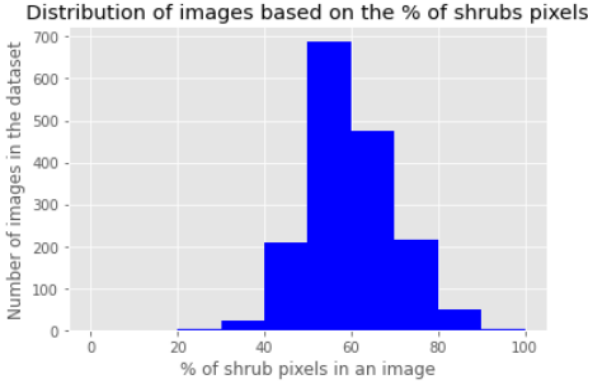
	Non-filtered sub-dataset C (117 instances: 100%)	Filtered sub-dataset C: patches containing more than 1% of shrub pixels (96 instances: 82%)	Filtered sub-dataset C: patches containing more than 45% of shrub pixels (15 instances: 13%)
Shrub class distribution in non-augmented datasets (117 instances)	 (1. ~23% shrubs pixels in the set)	 (2. ~26% shrubs pixels in the set)	 (3. ~55% shrubs pixels in the set)
Shrub class distribution in augmented datasets (1664 instances)	 (4. ~24% shrubs pixels in the set)	 (5. ~29% shrubs pixels in the set)	 (6. ~60% shrubs pixels in the set)

Figure A 8 Overview of shrub distribution in the original and under-sampled versions of sub-dataset C

n

Table A 8 Results on validation and test data. Symbol "-" signifies models that were not evaluated on test data. Symbol "-//-" signifies repeating values.

Sub-dataset	Sub-dataset size	Model input (height x width) [px]	Other specifications	Val. accuracy	Val. precision	Val. recall	Val. F1	Val. IoU	Test set	Test accuracy	Test precision	Test recall	Test F1	Test IoU	Avg. test F1/ model	Avg. test F1/ dataset family	Avg. test F1/ test set in a dataset family
A	832	128x128		0.46	0.23	0.96	0.31	0.22	**1**	0.84	0.39	0.30	0.46	0.30	0.55	*0.58*	test set 1
									2	0.50	0.51	0.91	0.65	0.49			0.52
									3	0.41	0.37	0.91	0.53	0.36			test set 2
	1664	128x128		0.84	0.71	0.56	0.63	0.46	**1**	0.81	0.92	0.32	0.47	0.31	0.56		0.64
									2	0.70	0.83	0.50	0.62	0.45			test set 3
									3	0.69	0.58	0.61	0.60	0.42			0.58
	3832	128x128		0.85	0.73	0.64	0.68	0.52	**1**	0.84	0.88	0.49	0.63	0.46	0.63		
									2	0.71	0.83	0.54	0.65	0.49			
									3	0.72	0.63	0.60	0.61	0.44			
B	808	128x128		0.83	0.80	0.49	0.61	0.44	-	-	-	-	-	-	-	-	-
	1658	128x128		0.87	0.90	0.65	0.76	0.61	-	-	-	-	-	-	-	-	-
		192x192		0.87	0.86	0.67	0.75	0.60	**1**	0.85	0.92	0.47	0.62	0.45	0.61	*0.62*	test set 1
									2	0.70	0.84	0.49	0.62	0.45			0.61
									3	0.72	0.63	0.58	0.60	0.43			test set 2
	3808	128x128		0.90	0.94	0.75	0.83	0.71	-	-	-	-	-	-			0.63
		192x192		0.91	0.90	0.76	0.82	0.70	**1**	0.84	0.88	0.46	0.60	0.43	0.62		test set 3
									2	0.71	0.83	0.53	0.64	0.48			0.62
									3	0.72	0.63	0.63	0.63	0.46			
									4	0.63	0.68	0.00	0.00	0.00			
C	808	128x128		0.66	0.48	0.64	0.55	0.38	-	-	-	-	-	-	-	-	-
	1664	128x128		0.84	0.86	0.68	0.80	0.66	-	-	-	-	-	-	-	-	-

o

		144x144		0.88	0.96	0.72	0.82	0.69	**1**	0.83	0.90	0.52	0.66	0.49	0.67		test set 1
									2	0.63	0.88	0.63	0.74	0.58			0.68
									3	0.63	0.78	0.52	0.62	0.45			test set 2
	3808	128x128		0.85	0.97	0.73	0.83	0.71	-	-	-	-	-	-			0.72
		144x144		0.88	0.96	0.77	0.86	0.75	**1**	0.82	0.92	0.47	0.62	0.45	0.61	*0.67*	test set 3
									2	0.60	0.88	0.52	0.65	0.49			0.60
									3	0.62	0.78	0.43	0.55	0.38			
		288x288		0.94	0.90	0.90	0.90	0.81	**1**	0.89	0.84	0.72	0.77	0.63	0.72		
									2	0.74	0.76	0.76	**0.76**	0.61			
									3	0.67	0.56	0.71	0.62	0.45			
									4	1.00	0.00	0.00	0.00	0.00			
C	1664	144x144	unfiltered (orig. model; orig.set.)	0.88	0.96	0.72	0.82	0.69	-\\-	-\\-	-\\-	-\\-	-\\-	-\\-			
		144x144	1% (orig. model)	0.86	0.86	0.77	0.81	0.73	**1**	0.80	0.80	0.52	0.63	0.46	0.65	*0.68*	test set 1
									2	0.63	0.88	0.62	0.73	0.57			0.66
									3	0.63	0.78	0.48	0.59	0.42			test set 2
									4	0.57	0.64	0.03	0.06	0.03			0.74
		144x144	45% (orig. model)	0.74	0.73	0.78	0.76	0.75	**1**	0.74	0.59	0.80	0.68	0.52	0.71		test set 3
									2	0.61	0.74	0.71	**0.76**	0.61			0.63
									3	0.56	0.64	0.71	0.68	0.51			
C	1664	144x144	Batch size = 15 (model sel.45%)	0.73	0.76	0.78	0.77	0.75	**1**	0.74	0.59	0.79	0.68	0.51	0.70	*0.68*	test set 1
									2	0.60	0.74	0.73	**0.76**	0.61			0.66
									3	0.56	0.63	0.72	0.67	0.51			test set 2
		144x144	Batch size = 32 (model sel.45%; orig.set.)	0.74	0.73	0.78	0.76	0.75	-//-	-//-	-//-	-//-	-//-	-//-			0.74
		144x144		0.72	0.76	0.78	0.77	0.73	**1**	0.71	0.54	0.71	0.61	0.44	0.64		test set 3

			Batch size = 50 (model sel.45%)						**2**	0.57	0.75	0.64	0.70	0.54			0.65
									3	0.56	0.63	0.58	0.61	0.43			
		144x144	dropout = 0.05 (model sel.45%; orig.set.)	0.74	0.73	0.78	0.76	0.75	- //-	-//-	-//-	-//-	-//-	-//-			
		144x144	dropout = 0.2 (model sel.45%)	0.71	0.73	0.75	0.74	0.82	**1**	0.57	0.40	0.78	0.53	0.36			test set 1
									2	0.60	0.71	0.68	0.74	0.58	0.64		0.50
									3	0.51	0.58	0.71	0.64	0.47			test set 2
									4	0.32	0.45	0.97	0.61	0.44			0.73
		144x144	dropout = 0.5 (model sel.45%)	0.41	0.61	0.71	0.66	0.69	**1**	0.33	0.25	0.62	0.35	0.22		*0.61*	test set 3
									2	0.39	0.58	0.70	0.69	0.52	0.52		0.61
									3	0.32	0.43	0.67	0.53	0.36			
									4	0.34	0.44	0.70	0.57	0.40			
		144x144	dropout = 0.75 (model sel.45%)	0.41	0.60	1.00	0.75	0.60	**1**	0.31	0.30	0.91	0.45	0.29			
									2	0.39	0.59	0.90	0.71	0.55	0.59		
									3	0.26	0.44	0.92	0.60	0.43			
									4	0.31	0.44	0.94	0.60	0.42			
		144x144	filters = 16 (orig. model; orig.set)	0.88	0.96	0.72	0.82	0.69	- \\-	-\\-	-\\-	-\\-	-\\-	-\\-			test set 1
		144x144	filters = 32 (orig. model)	0.89	0.97	0.75	0.84	0.73	**1**	0.81	0.92	0.41	0.57	0.40			0.62
									2	0.61	0.90	0.54	0.67	0.51	0.60		test set 2
									3	0.63	0.81	0.42	0.55	0.38		*0.66*	0.71
		144x144	filters = 64 (orig. model)	0.87	0.97	0.62	0.76	0.61	**1**	0.82	0.86	0.51	0.64	0.47			test set 3
									2	0.64	0.90	0.62	0.73	0.58	0.70		0.63
									3	0.66	0.80	0.65	0.72	0.56			
D	808	128x128		0.79	0.97	0.65	0.78	0.64	-	-	-	-	-	-			
	1658	128x128		0.81	0.98	0.67	0.79	0.66	-	-	-	-	-	-			

q

		192x192		0.87	0.97	0.75	0.85	0.74	**1**	0.81	0.97	0.42	0.58	0.41	0.62	*0.64*	test set 1
									2	0.63	0.88	0.61	0.72	0.57			0.62
									3	0.62	0.74	0.47	0.57	0.40			test set 2
									4	0.58	0.89	0.00	0.00	0.00			0.71
		400x400		0.96	0.92	0.89	0.90	0.82	**1**	0.87	0.88	0.56	0.69	0.52	0.66		test set 3
									2	0.75	0.76	0.61	0.68	0.51			0.59
									3	0.75	0.60	0.62	0.61	0.44			
	3808	128x128		0.84	0.99	0.73	0.84	0.73	**1**	0.78	0.93	0.42	0.58	0.41	0.64		
									2	0.52	0.90	0.63	0.74	0.59			
									3	0.52	0.76	0.48	0.59	0.42			
E	808	128x128		0.64	1.00	0.00	0.00	0.00	-	-	-	-	-	-	-	-	-
	1658	128x128		0.79	0.98	0.68	0.80	0.67	-	-	-	-	-	-	-	-	-
		240x240		0.89	0.98	0.78	0.87	0.76	**1**	0.84	0.91	0.51	0.65	0.49	0.67	*0.66*	test set 1
									2	0.62	0.85	0.69	**0.76**	0.61			0.66
									3	0.61	0.71	0.52	0.60	0.43			test set 2
		496x496		0.95	0.96	0.86	0.90	0.83	**1**	0.87	0.92	0.53	0.67	0.50	0.64		0.72
									2	0.72	0.83	0.56	0.67	0.50			test set 3
									3	0.75	0.70	0.49	0.58	0.41			0.60
									4	0.63	0.34	0.00	0.01	0.01			
	3808	128x128		0.79	0.99	0.70	0.82	0.70	**1**	0.78	0.93	0.52	0.66	0.50	0.67		
									2	0.42	0.93	0.61	0.73	0.58			
									3	0.46	0.83	0.48	0.61	0.43			

Table A 9 Total average F1 score per a test set

Test set no.	**1**	**2**	**3**
Total average F1 score of the test set	0.61	0.70	0.61

r