UNIVERSITAT DE BARCELONA

FUNDAMENTALS OF DATA SCIENCE MASTER'S THESIS

Man-made Structures Detection from Space

Author:
Peter WEBER and Eduard RIBAS FERNÁNDEZ

Supervisor:
Dr. Jordi VITRIA

A thesis submitted in partial fulfillment of the requirements for the degree of MSc in Fundamentals of Data Science

in the

Facultat de Matemàtiques i Informàtica

June 28, 2019

UNIVERSITAT DE BARCELONA

Abstract

Facultat de Matemàtiques i Informàtica

Man-made Structures Detection from Space

by Peter WEBER and Eduard RIBAS FERNÁNDEZ

With the development of affordable and recurrent remote sensing technology, we can now access frequent geospatial information in different levels of detail, ranging from 100m to 0.01m. The task of detecting various types of man-made structure and man-induced change has become a key problem in remote sensing image analysis.

In this work we focus on providing an answer to the question: what is the optimal tradeoff between resolution and cost when aiming at determining the existence of man-made structures in remote sensing images? Obtaining this value is important not only for designing optimal satellite sensors but also to use optimal data sources when developing data-based remote sensing products. At a global level, this knowledge contributes to understand the impact of our species on the planet.

Our approach is based on developing a deep learning detector to classify human impact on aerial images. In particular, we exploit recent advances of Convolutional Neural Networks (CNN) that were successfully used for object detection and scene classification. We apply transfer learning by integrating a ResNet pre-trained on ImageNet to perform image classification on datasets of few thousand aerial images that we have manually collected and annotated. Using this classification pipeline we are able to determine the existence of man-made structure with an accuracy of 95% at the best resolution.

We study the performance of our detector for resolutions ranging from 0.3m to 16m. We observe a linear decrease of the classification accuracy down to 87% at the lowest resolution. Furthermore, we estimate the cost associated to capturing and processing satellite images. This includes the entire pipeline from building and launching a satellite to predicting human impact on images. We estimate that monitoring the entire land surface of the earth at 1m resolution amounts for about $15 million. This cost increases by about two orders of magnitude at the best resolution studied here, and decreases by about one order of magnitude at a resolution of 10m per pixel. These results could be further improved by training a CNN on a labeled large scale remote sensing dataset. Nevertheless, our results suffice for studying the expansion of human kind using satellite imagery and provide valuable information for designing optimal satellite sensors.

Acknowledgements

First, we want to thank **Santi Seguí**, **Lluis Garido**, and **Eloi Puertas** to serve as experts in our thesis committee. We are very grateful to our thesis advisor **Jordi Vitrià** for close guidance and exceptionally constructive and creative ideas on how to afront the problems we encountered. His scientific instinct was indispensable in critical moments of the project. We also want to thank **Marco Bressan** from Satellogic for fruitful discussions, support with material, and advice about tools and data sources. Further, we want to thank **Javier Marin** and **Aitor Lucas** (both from Satellogic) for help with the datasets.

Contents

Chapter 1

Introduction

1.1 Motivation

Human kind exerts an ever increasing pressure on natural and ecological systems due to the associated consequences of the explosion of human population. The exploitation of the earth manifests itself in extraction of natural ressources, proliferation of human-made infrastructure and waste, and increasing production land use for crop and pastry land [1]. As a logical consequence, we observe widespread declines in biodiversity [2], decrease in natural habitat, attrition of wilderness areas, deforestation, and enhanced emission of greenhouse gases to the atmosphere. This increasing intrusion leads to reduction of benefits that humans receive from natural systems [3] such as the extinction of natural ressources, and ultimately provoke natural disaster induced by effects such as climate change.

An essential prerequisite to mitigate human threat to nature is the access to data that allows for spatial and temporal mapping of human activity [4]. To this end, the last decades have brought about developments of affordable and recurrent remote sensing technology [5]. In particular, we now have public and continuous access to overhead imagery data for earth observation in different levels of detail, ranging from 100m to 0.01m. Overhead imagery data is obtained either by satellites or by airborne sensor systems. Additionally, remote sensing technologies open up the road for applications in agricultue, disaster recovery, urban development, and environmental mapping.

The task of detecting various types of man-made structure and man-induced change has become a key problem in remote sensing image analysis. However, unlike the computer vision community that disposes of datasets with thousands or millions of images containing up to thousands of distinct annotations [6, 7, 8, 9], the remote sensing community is only recently making first steps towards creating standardized labeled large scale datasets. Several approaches into this direction have been focussing either on classifying land cover and land use [10] or annotating overhead images with object categories [11, 12]. These annotations were used to perform object detection or segmentation [13, 9], and to e.g. map roads and buildings [11, 14]. However, the statistics of the images and categories in these

works is heavily biased towards man-made structures so that wilderness areas are strongly underrepresented.

The computer vision community has largely benefited from recent advances in deep learning ultimately leading to the outsourcing of convolutional neural networks pretrained on massive datasets. In the remote sensing community, researchers are recently also starting to follow this pathway [10]. However, pretrained models are not yet widely available, so that many works in the remote sensing field are based on fine-tuning neural networks pre-trained on traditional computer vision tasks.

1.2 Satellogic

This work has been developed in cooperation with Satellogic, a company that provides earth observation data and analytics as a service to enable better decision making for industries, governments, and individuals. Satellogic was founded in 2010 in Buenos Aires, and has expanded since then with offices in Barcelona and China. Satellogic builds, launches and maintains their own satellites.

Satellogic focusses on developing heavily weight and cost optimized satellites. Their first nano satellite, called Capitán Beto, was sent to space in 2013 [15]. Currently, Satellogic has 31 satellites orbiting earth, whose weight is 35kg, a minuscul fraction of conventional satellite systems (more than 1000kg [16]). The satellites feature hyperspectral image aquisition at 1m pixel resolution. Satellogic envisions to have 300 satellites orbiting the earth within a few years providing real time imagery for any geospatial location.

The hyperspectral technology i.e. image aquisition capability in more than 30 spectral bands allows for monitoring the earth with great detail [17]. Every object and every plant has its own spectral fingerprint. Measuring the optical reflectance to the solar radiation for instance allows to distinguish between different kinds of crops, and its status of irrigation and fertilization. Further, it is possible to measure the level of pollution in the air and monitor vegetation below the water surface. Satellogic's clients apply this technology to map land use, monitor infrastructure, track agricultural development, evaluate the health of crops, and evaluate productivity of natural ressources.

At 1m pixel resolution, 10 satellites can remap 1 million square kilometers every 6 weeks. Note that the surface of the earth is roughly 500 million square kilometers of which about 30% is land and 70% is water, which brings us to the goals of this thesis.

1.3 Thesis goals and outline

The motivation for this Master's thesis is to provide an answer to the question: What is the optimal resolution to detect human impact in satellite imagery, having in mind

the economical cost of acquiring and processing the information? Determining this value is important not only for designing optimal satellite sensors but also to use optimal data sources when developing data-based remote sensing products. The goal here is not to build the top performance, state-of-the-art model to detect all sorts of human impact in satellite images, but rather to analyze the feasibility and cost of doing so at different resolutions. Of course, better algorithms could be trained on larger datasets to accurately identify certain types of human impact, but we consider a more general problem.

To address this problem, we divide the work into three parts. First, we develop a detector that is capable of identifying man-made structures on two aerial imagery datasets that we collect and annotate. Next, we study the performance of the detector in terms of classification accuracy as a function of image resolution i.e. the resolution per pixel. In the last part, we provide an approximate estimation of the costs of the entire pipeline. These also include metrics related to building, launching and maintaining a satellite. The estimation is performed for the entire spectrum of resolutions ranging from 0.03m to 16m.

The detailed outline of the thesis is listed below, and the Jupyter notebooks demonstrating the experimental work can be found on our Github repository (see link or reference [18]).

- Chapter 2 provides an overview of existing datasets and a detailed description of the construction of our own datasets. We further discuss the data manipulation pipeline.
- Chapter 3 gives an introduction to deep learning. We discuss theoretical concepts and recent advances in the field.
- Chapter 4 discusses the approach we followed to develop a detector capable of classifying man-made structures in aerial images.
- Chapter 5 presents the final results regarding the performance of the deep learning detector as a function of resolution for multiple image categories. It also provides an estimation of the cost to monitor the entire surface of the earth.
- Chapter 6 concludes the thesis and outlines potential next steps.

Chapter 2

Building datasets

In this chapter, we will give an overview of existing annotated aerial imagery datasets and outline the reasons why none of them is suitable for our investigation. Following this discussion, we will describe two approaches for obtaining our own labelled dataset.

2.1 Requirements and considerations

Before we go into the presentation of labeled datasets we discuss the requirements that the dataset needs to fulfill in order to serve for the investigation in this thesis project. As a refresher, we want to detect human impact on aerial images and determine the dependency on resolution per pixel of a chosen evaluation metric. Ideally, the range for the resolutions should scale from a few tens of centimeters to a few tens of meters, whereas the images with low resolution can be generated from the high resolution images by downsampling. Having in mind previous arguments, we mainly need to consider three aspects.

First, we need to have imagery data with labels that can be used to clearly distinguish between existing and non-existing human impact, respectively. This impact might be classified pixel wise, or as binary classification for the entire image, or as multi-class classification that can be translated into binary labelling. Second, we need a balanced dataset of approximately the same number of images for both classes, and a large variaty of different terrains within each class. Third, the images need to have a resolution per pixel which is equal or better than 1m. Also, the height and width of the images should measure at least 500×500 pixels, so that one has enough room for downsampling.

2.2 Existing datasets

In table 2.1 we have summarized the most relevant remote sensing datasets with ground truth labels that can be found in literature. The datasets were collected using different publicly available data sources. These range from pure low-resolution satellite imagery (Sentinel-2) to high-resolution images taken with an aircraft (USGS) to a mix of different image sources (Google Earth).

The satellite images have a resolution of equal or larger than 10 m and they are collected with the Sentinel-2 satellites of the European earth observation program Copernicus. Although the datasets from this source (BigEarthNet and EuroSat) are comparatively large, they do not suffice for our purpose, because the resolution is not good enough and the images are too small.

name	source	images	resolution (m)	size (pixel)	categories
BigEarthNet [10]	Sentinel-2	590,326	10, 20, 60	120, 60, 20	~ 50
EuroSAT [19]	Sentinel-2	27,000	10	64	10
UCMerced [13]	USGS	2100	0.3	256	21
DeepSat [20]	USGS	405,000	1	28	6
AID [21]	Google Earth	10,000	0.5 - 8	600	30
PatternNet [22]	Google Earth	30,400	0.06 - 4.69	256	38

TABLE 2.1: **Publicly available remote sensing datasets with labels.** The table lists the name of the dataset together with the bibliographic reference. It also details the data source of the images. It contains a description about the number of images, the resolution of the images, the size of the square images in pixel, and the number of categories.

The USGS National Map Urban Area Imagery collection [23] was utilized to collect remote sensing datasets in the two works UCMerced and DeepSat, where the former is the dataset that comes closest to our requirements. It features an image resolution of 0.3m per pixel, and the images have a height and width of 256 pixels. However, out of the 21 categories only 2 belong to images without human impact, while the other 19 show man-made structures. The DeepSat dataset unfortunately consists of image patches which are only 28×28 large, so that we aren't able to study these images as function of resolution.

The datasets using Google Earth as data source are collected using either the Google Earth or the Google Maps application programming interface (API). These images vary in resolution as well as in their original data provider since Google accesses several data sources. Both datasets, the AID and the PatternNet dataset, have about 30 categories with several hundred images in each category. Here, different categories have different pixel resolutions, and again most of the categories relate to urban areas so that we do not have sufficient images without human impact. Even the categories that in principle should not show human influence contain images that break this rule.

Overall, the main issue with these datasets stems from the fact that non of them was collected with the purpose to analyze the human footprint. Therefore they are very unbalanced, and do not contain sufficient variety of images for the classes without human influence. We hence decided to collect and label images by ourselves. In our first approach we used the Google Maps API, and in our final approach we used datasets from the USGS aerial imagery collection.

2.3 Google Maps

Google has a public API that allows for querying images from their service Google Maps [24]. In its most basic form, the API accepts as input parameters a lattitude (lat) and longitude (lon), a zoom, and the image size (in pixels). Given this set of parameters one can calculate the resolution in meter per pixel [25], which is given by

$$resolution\left[\frac{meter}{pixel}\right] = \frac{156543.03392 \cdot cos(\frac{lattitude \cdot \pi}{180})}{2^{zoom}}. \tag{2.1}$$

Equipped with this toolkit, we developed an automated image download pipeline that was based on one of several approaches of selecting and downloading images in a given area. In our first approach we selected images that were Gaussian distributed around a center location defined by a set of lat/lon coordinates. We further provided a precision parameter (standard deviation of the Gaussian), the number of images to download, a list of zooms, the desired image size, and the number of pixels to crop from the border of the image in order to remove e.g. the Google logo. Another approach consisted in downloading randomly sampled locations from within a rectangle defined by its upper left and lower right lat/long coordinates, respectively.

Although any of these two approaches would have served to build a complete dataset in an automated fashion, we finally decided to use a different data source due to the following reasons. According to our advisor from Satellogic, Google Maps images have one major drawback regarding the pixel resolution. The Google Maps image is an interpolation from different spectral bands, where the RGB color bands do not necessarily have the expected resolution. Therefore, the resolution estimated by Eq. 2.1 is not reliable for the three color channels. We did not further investigate into this issue and instead turned to a different solution, which is discussed next.

2.4 USGS land cover

2.4.1 Getting the data

To be able to construct a balanced and representative dataset we were recommended to focus on images of the United States, because of the wealth of available high resolution aerial imagery data from USGS Earthexplorer [23]. A nice side effect of choosing the United States is that a large variety of images of different terrain and topology are available. We combined the aerial imagery datasets from USGS with additional information about land cover and land use from the USGS Land Cover Viewer [26], precisely to guarantee larger variety through the selection of data from distinct land use categories.

For the determination of relevant geographic locations we excluded cities and highly developed urban areas, and instead focussed on unpopulated areas. Specifically, we limited our image search to the four land use categories agriculture,

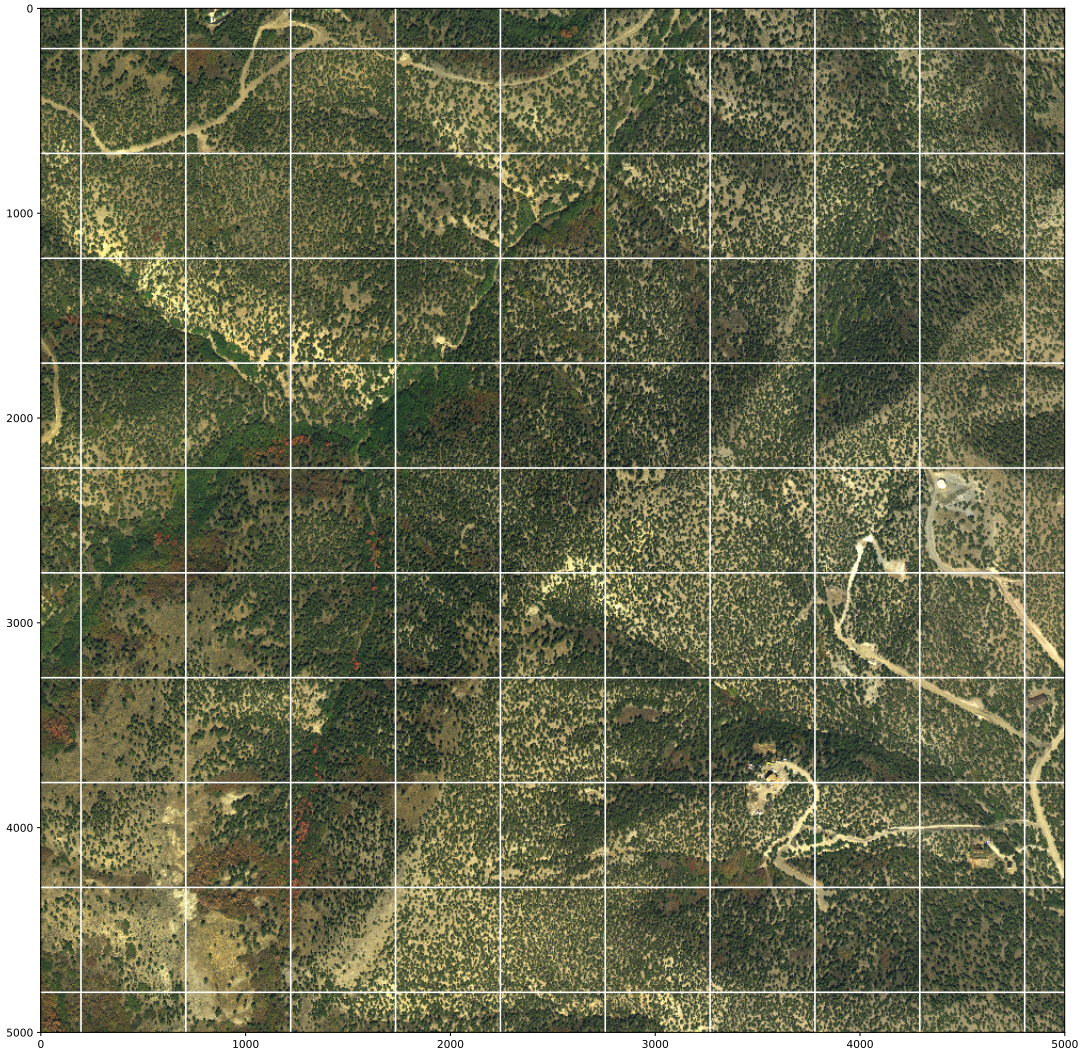

FIGURE 2.1: **Example of unprocessed image.** This image has a size of 5000 $\times$ 5000 pixels. The continuous white lines are guidelines for the eye to demonstrate how we crop smaller images where we only return a cropped image from the white squares with size 512 $\times$ 512 pixels.

shrubland-grassland, semi-desert, and forest-woodland that can be found in the USGS Land Cover Viewer. Note that these categories served as a rough geographic orientation to pin down geolocations of interest. However not all the images could be assigned with absolute certainty to one unique category since we were not able to overlay both maps. Further, we selected many images from national parks because we found that it is significantly harder to find imagery data that does not show human influence. Whenever possible we also tried to collect images from both classes (man-made vs. natural) within a given area/terrain.

Once an area was pointed out as a region of interest using the USGS Land Cove Viewer, we located it on USGS Earthexplorer and downloaded images from that area. In particular, we constructed two datasets with 0.3m and 1m resolution, respectively. The former was taken from the category High Resolution Orthoimagery and the latter from the category National Agriculture Imagery

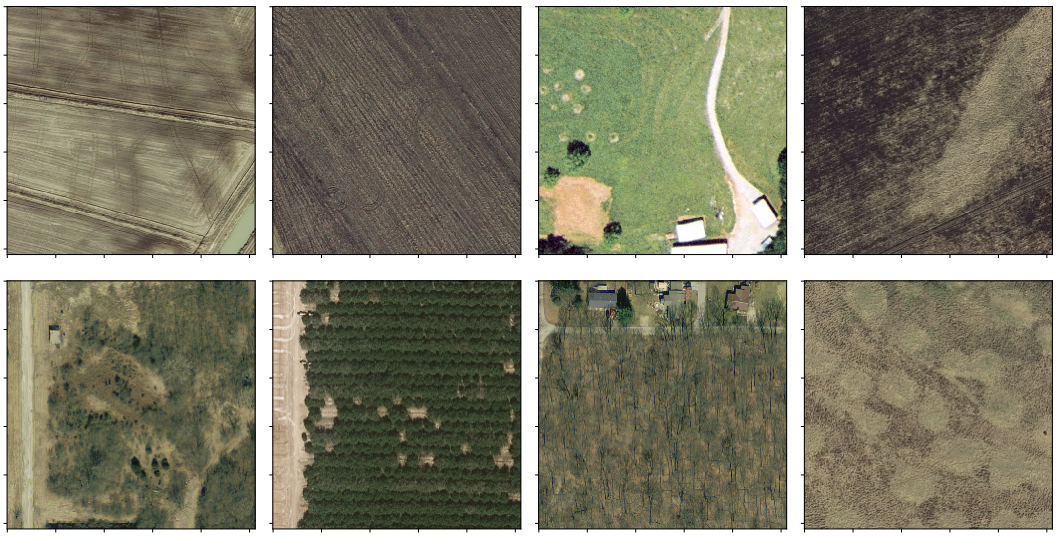

FIGURE 2.2: **Example images of category Agricultue.** All images in this figure show clear signs of human impact. The images have a size of 512 × 512 pixels and a resolution of 0.3m per pixel.

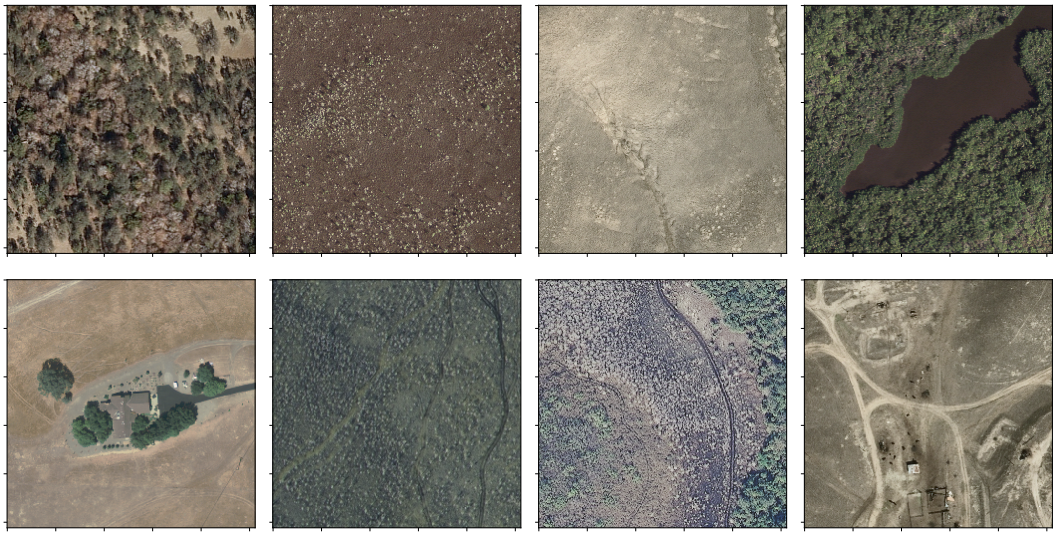

FIGURE 2.3: **Example images of category shrubland-grassland.** The images in the first row do not contain any human influence, while the images in the second row show man-made structures. The images in this figure have a size of 512 × 512 pixels and a resolution of 0.3m per pixel.

Program (NAIP). Note that the images in these categories usually have a height and a width of several thousand pixels, and hence occupy a few hundreds of Megabytes of disk space. We cropped smaller images out of the raw images, which will be discussed in more detail in the following section. Overall, we downloaded about 100 raw images for each dataset. An example is shown in Fig. 2.1.

FIGURE 2.4: **Example images of category forest-woodland.** The images in the first row do not contain any human influence, while the images in the second row show man-made structures. The images in this figure have a size of 512 × 512 pixels and a resolution of 0.3m per pixel.

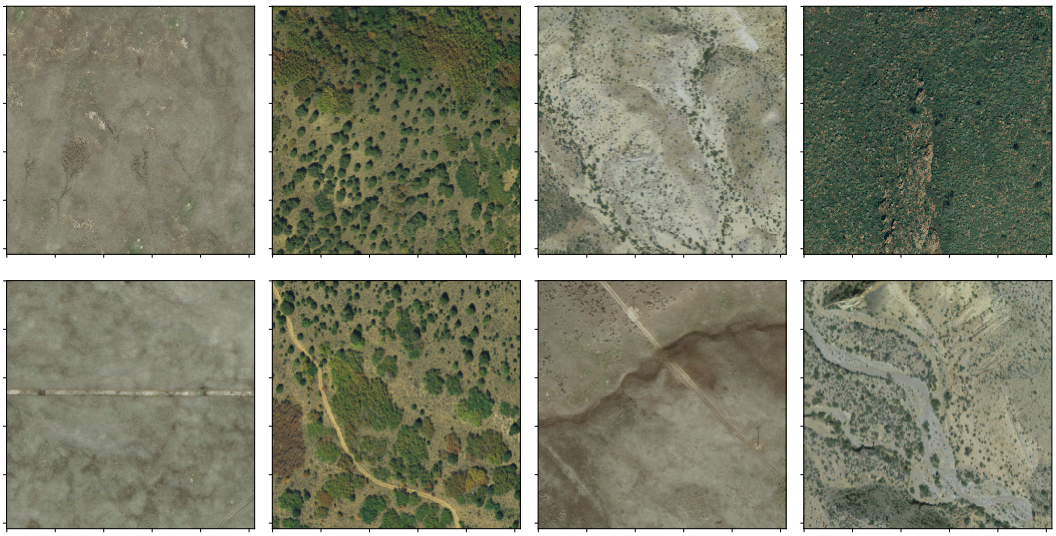

FIGURE 2.5: **Example images of category semi-desert.** The images in the first row do not contain any human influence, while the images in the second row show man-made structures. The images in this figure have a size of 512 × 512 pixels and a resolution of 0.3m per pixel.

2.4.2 Data processing and labeling

Our data processing pipeline consists of the following steps:

1. Download large raw images.

2. Crop images of size 512 × 512 pixel.

3. Label images with either zero (no human impact), one (minimal human impact), two (clear human impact).

4. Degrade images i.e. reduce number of pixels and thereby resolution per pixel.

Let us discuss each of these steps in more detail. An illustration of the first and second step of the image processing pipeline is given in Fig. 2.1. The white lines demonstrate the way we crop smaller images (512×512 pixels) from the large raw image (in this case 5000×5000 pixels). We process all raw images in this manner, which yields approximately 80-150 processed images per raw image. We hence obtain about 10,000 processed images for each dataset. Within each category (USGS Land Cover categories) of the processed images we labeled a selected portion of the images by moving them into the folder with the respective label name.

We have published our datasets via a Google Drive link [27]. The image folder of the published datasets contains the raw images, the processed images, and the labeled images. In this folder we follow a specific folder structure, which is shown below. Here pointy brackets (<parameter>) indicate a parameter and the content in the optional curly braces determines whether it is a folder pertaining to raw images. The first parameter is $pixels = 512$ and the second parameter represents the resolution of the dataset. Note that the label folders only exist in the case of processed images.

```
{raw-images-}usgs-<pixels>-res<resolution>m
├── semi-desert
│   ├── label-0
│   ├── label-1
│   └── label-2
├── agriculture
│   └── label-2
├── shrubland-grassland
│   ├── label-0
│   ├── label-1
│   └── label-2
└── semi-desert
    ├── label-0
    ├── label-1
    └── label-2
```

Annotating the images with labels was performed following certain rules. First, we classified images with no human impact at all into the class with label zero, while we classified images with clear human influence into the class with label two. Ambigious images i.e. images with minimal human traces, such as a small walking path, were classified into class one. Second, we've put major effort into creating datasets that contain images of similar texture spread across all classes. If we for

example classified a set of images of a certain forest type into class zero we classified another set of images with a similar forest type, but containing a building or a street, into the class two. We followed the latter rule for all categories except agriculture. The agriculture images all show human influence. We therefore classified them all with label two. By sticking to these rules, we are able to guarantee that the algorithm learns features that relate to the appearance of man-made structures, and not to image artefacts such as color or texture.

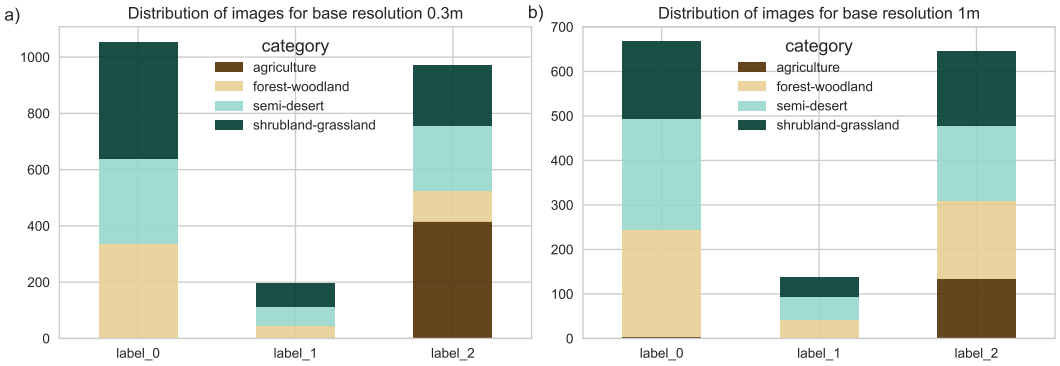

FIGURE 2.6: **Number of images per category and label.** (a) Distribution of images for dataset with resolution of 0.3m per pixel. (b) Distribution of images for dataset with resolution of 1m per pixel.

In Figures 2.2 - 2.5 we display sample images for each of the four categories, repsectively. These images belong to the dataset that has a pixel resolution of 0.3m. The images from the 1m dataset have similar characteristics, but are not shown due to redundancy. Note that in Figs. 2.3 - 2.5 the first row represents images of label zero and the second row shows images that belong to label two. As mentioned above, the images in Fig. 2.2 (agriculture) all contain human influence, and therefore belong to class two.

The distribution of categories and labels is shown in Fig. 2.6. Overall, for the 0.3m dataset we classified about 2200 images, and for the 1m dataset we classified about 1450 images. Our main goal consisted in creating a balanced dataset between label zero and label two as can be seen from the distributions. A minority of images, roughly 10% of all annotated images were assigned to label one. These images were used at random to investigate the behaviour of the Machine Learning classifier, which is discussed in chapter 5.

The last step of the data processing pipeline consisted in downsampling the processed and labeled images, in order to obtain images with a lower resolution. We used a Lanczos filter [28] for the sampling, which is based on a sinusoidal kernel. In Fig. 2.7 we show a few selected resolutions for an example image from the agriculture category. Note that here we only schematically depict an example in order to illustrate the process. However, in our Machine Learning pipeline the images are downsampled on the fly and the result of this process is not stored on disk (see Section 4.2 for further details).

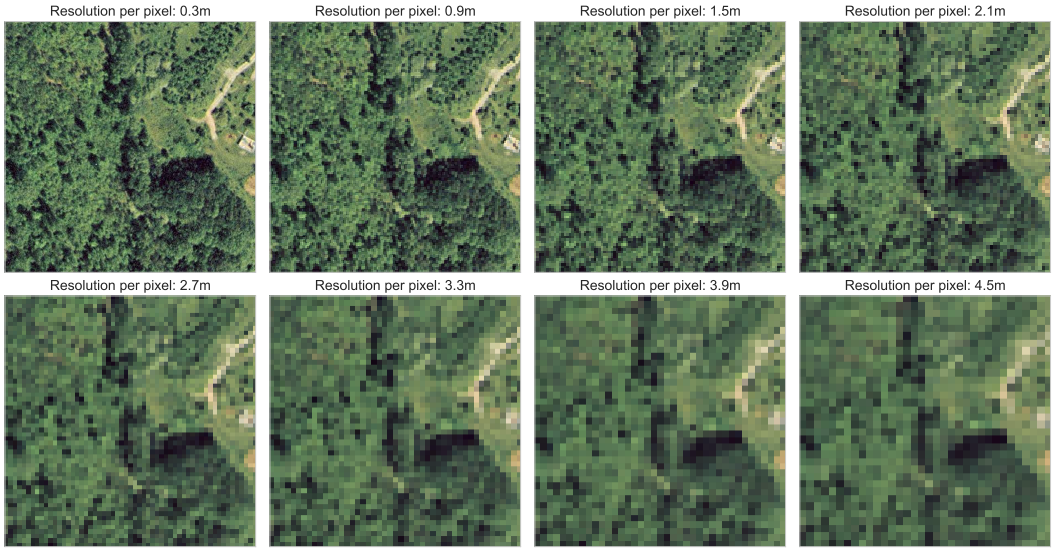

FIGURE 2.7: **Example of image downsampling**. The upper left image has a base resolution of 0.3m per pixel and a size of 512×512 pixels whereas the lower right image has the worst resolution, 4.5m per pixel, and a size of 34×34 pixels. All intermediate images are downsampled by a factor corresponding to the resolution of the actual image divided by the base resolution. For instance, for the lower right image the factor is 15.

For this particular image one can observe how certain image features disappear as the image quality is decreased. Above a resolution of around 3m per pixel one is not able anymore to identify the building close to the right corner of the image. The texture of the track that leads up to the building is blurred above a resolution of around 4m per pixel. This shows how different elements in an image are not recognizable anymore once the resolution approaches their characteristic size.

Chapter 3

Deep Learning

In this chapter, we will provide a short overview of the theoretical concepts and recent advances in the Deep Learning field. We will give a basic introduction to Neural Networks, and discuss convolutional Neural Networks in more detail as these are the type of algorithms utilized in this work. We further will summarize some of the most popular Convolutional Neural Network architectures.

3.1 Introduction to Deep Learning

Deep Learning (DL) models have led to vast performance improvements in a large variety of domains, and therefore have gained substantial popularity over the last decades. These models were initially inspired by the human brain and analogies in neuroscience, which is why this class of algorithms was coined Neural Networks (NN). The two most popular Neural Network architectures are convolutional Neural Networks (CNN) and Recurrent Neural Networks (RNN). CNNs have driven major breakthroughs in visual object recognition [29], and image [30], video [31] and audio [32] processing while RNNs brought about advances in research and applications on sequential data, i.e. in speech and text [33]. However, the superior performance of Neural Networks compared to traditional Machine Learning algorithms is not limited to the aforementioned domains. Other fields in which NNs have advanced the state-of-the-art include, for instance, bioinformatics [34] and the analysis of data from elementary particle physics [35].

Neural Networks define a class of models that are composed of a variable number of processing layers (Hidden units) of simple models, and are generally used to map a fixed size input (e.g. the pixels of an image) to a fixed size output (e.g. a category or a probability). A Hidden unit of a fully connected (FC) Neural Network has connections between all the nodes of the previous layer and the next layer. These connections are fully parametrized by the weights of the network. In analogy to a firing neuron, a non-linear activation function is applied to the output of the nodes of every Hidden unit. Historically, the *sigmoid function*

$$\sigma(x) = 1/(1 + exp(-x))$$

and the *hyperbolic tangent*

$$tanh(x) = 2\sigma(2x) - 1$$

have been used as activation functions. Nowadays, the most popular activation function is the *rectified linear unit* (ReLU)

$$f(x) = max(0, x).$$

We will see the reason for this at the end of the section. In Fig. 3.1(a) we show an example of a fully connected feedforward 3-layer Neural Network.

The strength of Neural Networks lies in their ability to learn arbitrarily complex non-linear input-output mappings [36], and that they can automatically extract features from raw data. The latter is in stark cotrast to traditional Machine Learning algorithms, which require careful feature engineering. For instance, when dealing with images, the multi layer architecture of Neural Networks allows to learn different features at every stage of the network, where the complexity and the abstraction of the learned features increases at every layer [37].

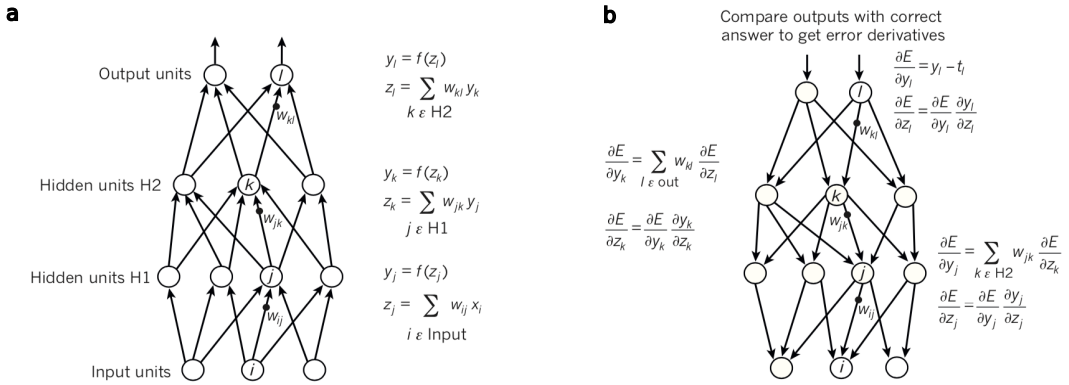

FIGURE 3.1: **Example of 3-Layer Neural Network. (a)** Feed-forward representation of a Neural Network with two hidden units H1 and H2 and a binary output unit. The inputs to every layer are weighted averages, specified by the weights w, of the outputs of the previous layer. In every layer, the outputs are generated by applying a non-linear function to the inputs. The most popular function for this purpose is the ReLu (see text). **(b)** Back-propagation of the error in order to learn the optimal weights of the Neural Network. The error is quantified by a loss function E at the output of the Neural Network, which is a measure of the discrepancy between the desired output and the actual output of the network. During backpropagation the chain-rule is applied recursively to the loss function in a backward manner. Specifically, at every layer the derivative of the error with respect to the inputs is computed by mutlipliying the upstream gradient with the local gradient. The upstream gradient is the derivative of the loss function with respect to the output of each unit, which is a weighted sum of the input derivatives of the layer above. The local gradient is the derivative of the non-linear function $f(z)$ with respect to its inputs. Starting from the output of the network one finally obtains the derivative of the loss function with respect to all weights, so that the network can minimze the loss by adjusting the weights. Panel is adapted from [38].

As all Machine Learning models, Deep Learning models are trained by minimizing an objective function i.e. by finding the optimal set of weights that achieve a specific input-output mapping. A typical objective function (also loss

function) in a classification setting is the cross-entropy loss combined with a softmax. For the i-th training example it is

$$L_i = -log\left(\frac{e^{f_{y_i}}}{\sum_j e^{f_j}}\right)$$

where f_{y_i} represents the class score computed for the real class, and f_j is the j-th element of the vector of class scores. The total loss is the average over all L_i. The minimization of the objective function is accomplished by applying gradient descent based methods, in practice, often stochastic gradient descent [39] or variants of it [40]. The computation of the gradient of the objective function with respect to all weights of the network is accomplished using backpropagation [41] (see Fig. 3.1(b)). Intuitively, the backpropagation algorithm helps to quantify the influence of every weight of the network on the final error, so that one can decrease the error by updating the weights in the direction of the negative gradient.

Although artifical Neural Networks have been known and studied since the 1950s, it was only understood in the 1980s that mutlitlayer networks could be trained by backpropagation and stochastic gradient descent [42]. However, until recently, Neural Networks were still ignored by the computer-vision and speech-recognition communities, because of the believe that the objective function would get trapped in local minima.

The advent of several new methods and technologies shall prove wrong the scepticism towards feasibly training deep Neural Networks. A requirement to reliable train large Neural Networks is the availability of large amounts of labelled data, as well as the necessary processing power. Both became available about a decade ago with the emergence of "Big Data" and new powerful graphics processing units (GPUs). Also theoretical advances helped to eliviate the difficulties to train Deep Learning models. These include the application of the ReLU non-linearity [43], which solved the vanishing gradient problem, as well as the development of a particular type of NNs, the Convolutional Neural Networks. These networks are much easier to train than conventional FC networks, have less parameters, and they generalize better to unseen data.

3.2 Convolutional Neural Networks

Convolutional Neural Networks [44, 45] are specifically designed to process input data that has the shape of multiple arrays, such as the pixel values of a 2-dimensional image with three color channels. This is accomplished by using additional layers to preserve spatial structure. In general, a CNN is composed of several convolutional layers followed by a nonlinearity. These are often followed by a pooling layer, and a fully connected layer is used as the last layer of the network. The architecture of a small VGG convolutional net [46] used for classification is shown in Fig. 3.2.

With this design, CNNs take advantage of the natural properties of images. The central element here is the convolutional layer, which takes into account that local pixel values are highly correlated, and that the local statistics of images are invariant to translation [47]. In particular, in a convolutional layer several small filters are slided spatially over the image computing dot products at every spatial location. The filters always extend the full depth of the input volume. For instance, a typical filter for a 3 color channel image might have dimensions $5 \times 5 \times 3$. Sliding this filter over an image of size, say $32 \times 32 \times 3$, would lead to an activation map with dimensions $28 \times 28 \times 1$.

The activation map is produced with one set of weights that belong to this particular filter. This concept is referred to as shared weights, which means that a comparatively small number of weights is shared across the entire image. As it is the case with conventional Neural Networks the weights, or parameters, are learned by applying gradient descent and backprogagation. The number of parameters per filter is given by the spatial filter size, times the depth of the image plus a bias term. In the usual case of having multiple filters K the number of parameters is multiplied by K, which yields K activation maps. Each of these activation maps relates to one particular feature in the image where a spatial location in the activation map corresponds to the same spatial location in the input image (see Fig. 3.2(a)).

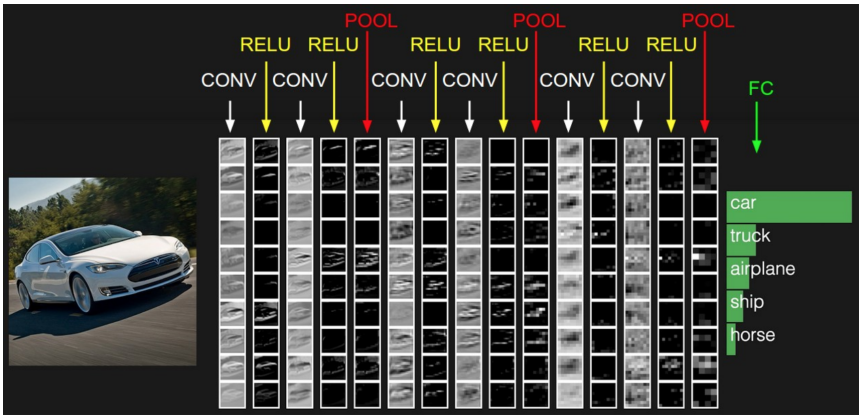

FIGURE 3.2: **Activations of a convolutional Neural Network used for classification.** The forward pass in this network is computed from left to right. Every column represents a layer of the network and the small images are the filter activations when passing the passing the image through the network. The structure shown here is typical for CNNs in that it has convolutional layers followed by a non-linear activation function. Multiple layers of these are followed by a pooling layer. The output of the network are a set of class scores produced by the final fully connected layer. Figure is adapted from [48].

The dimensions of the output of every convolutional layer are controlled by three parameters: the stride S, zero-padding P, and the number of filters with size F. The stride is the interval at which the filter is slided over the image. Zero-padding has the porpuse to increase the image size by adding pixels with zero value at the border, so that the input and output dimensions can be matched (assuming stride is 1). For

an input image of size W the output activation map will have

$$\frac{W - F + 2P}{S} + 1 \tag{3.1}$$

pixels along every dimension.

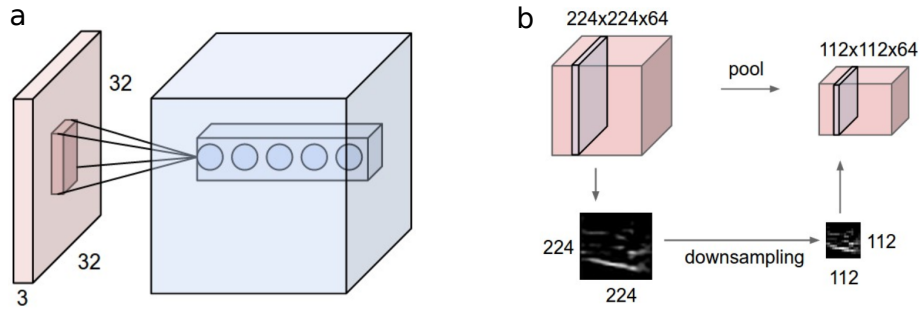

FIGURE 3.3: **Example of a convolutional layer (a) and a pooling layer (b). (a)** In this example five small filters (represented by the 5 dots in the output volume) are applied to the input image of dimensions $32 \times 32 \times 3$. The filters extend over the entire input depth, but look only at a small region spatially, defined by the filter size. Every layer, i.e. every activation map in the output volume is generated by sliding one filter over the entire image. In essence, every filter is sensitive to a specific feature in the input image. **(b)** Schematic representation of the effect of a max-pooling layer with stride 2. Effectively the image is downsampled where in every 2-by-2 area of the input volume the maximum pixel value is passed to the output. Apart from downsampling, pooling introduces an invariance to local variations. Figure is adapted from [48].

Pooling layers introduce coarse-graining in order to create invariance to small shifts and to decrease the number of parameters, which makes the representations smaller and more manageable. A pooling layer is applied to every activation map independently. It downsamples its input, in the most common case, by applying a max operator. This is shown schematically in Fig. 3.3(b). Note that pooling layers don't have any parameters.

When stacking together mutliple combinations of convolutional layers followed by non-linearities and pooling layers, the filters learn a hierarchichal structure. The filters at earlier layers learn simple low-level features such as edges whereas the filters at later stages learn more complex high-level features, which are compositions of lower level features. In conclusion, the deeper a convolutional Neural Network is the more complex compositions of features it can learn.

3.3 CNN architectures

The AlexNet was the first convolutional Neural Network that achieved remarkable results in the ImageNet classification task in 2012 (see Fig. 3.4). It halfed the error in comparison to all competing non Deep Learning based approaches. In this competition deep Convolutional Networks were applied to a dataset with roughly 1 million images and 1000 classes. AlexNet was specifically designed to be trained on two GPUs with each 3GB of memory, which was sufficient to fit all the 60

million parameters inside. AlexNet had 8 layers, and used dropout as regularization technique.

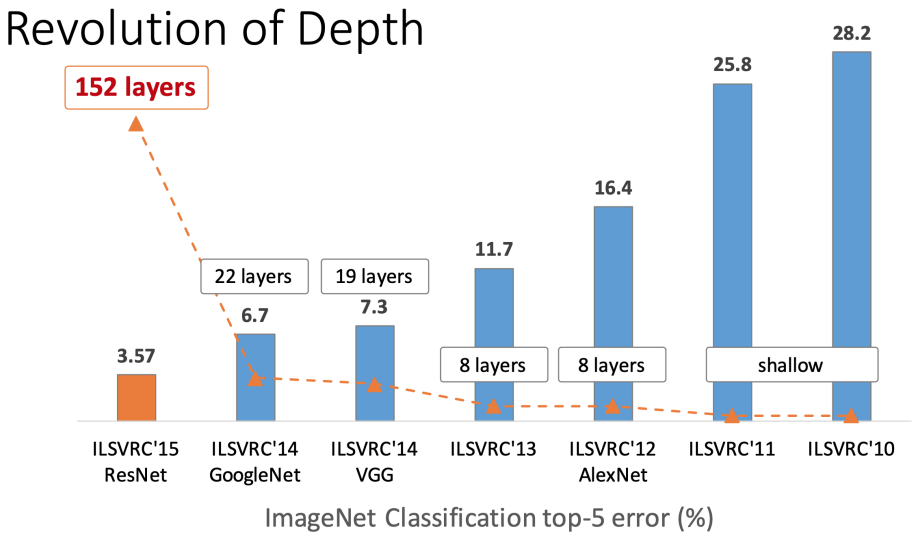

FIGURE 3.4: Top-5 test error of the winning solutions of the Large Scale Visual Recognition Challenge through years 2010 to 2015. Figure is adapted from this link.

The winner in 2013 was the ZFNet [49], which basically had improved hyperparameters compared to the AlexNet. In 2014, two networks were developed that were significantly deeper than previous networks. The VGG network [46] had 19 layers and the GoogleNet had 22 layers [50]. To be able to increase the number of layers, researchers from Oxford used very small filters (3×3) in VGG thereby reducing the number of parameters per layer significantly. The GoogleNet first introduced the Inception module.The idea behind the Inception module is to have several small networks within the network that do multiple convolutions and pooling in parallel. In combination with getting rid of fully connected layers, the GoogleNet has only 5 million parameters, 12 times less than AlexNet.

A significant improvement in the ImageNet challenge was achieved in 2015 by Kaiming He et al. [51], when they submitted ResNet. During the development of this architecture the authors found that very deep networks perform worse than. Having excluded overfitting, they hypothesized that the origin of this observation must have it's roots in wrong optimisation of the objective function. The argument they gave was that a deeper network always must perform at least as good as a shallower network. This can be seen when one replaces some of the modules in the deeper network by identity mappings. Following this interpretation, they introduced residual blocks that contained an identity mapping in parallel to the convolutional layer. Having a so called skip connection, the network just needs to learn the residual denoted as $F(x)$ in Fig. 3.6. Taking advantage of this approach they were able to design networks with a depth of up to 152 layers, which allowed for halfing the error rate of the ILSVRC challenge in 2015.

The architecture of a ResNet with 34 layers is shown in Fig. 3.6. The first

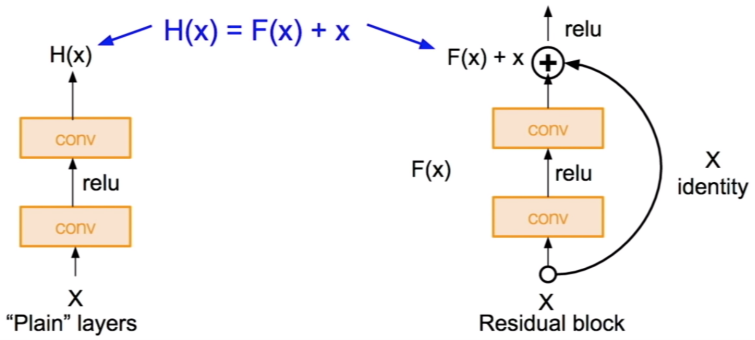

FIGURE 3.5: **Residual block of ResNet architecture.** Comparison between plain convolutional architecture (left) and convolutional architecture using a residual block. For the plain structure the network learns the mapping $H(x)$ while for the residual structure the network learns a residual $F(x) = H(x) - x$. x here is the identity mapping. Figure is adapted from [48].

convolutional layer has a filter size of 7×7 while all consecutive filters are chosen to be as small as possible (3×3). After every convolutional block that contains mutliple convolutional layers and residual blocks the image is downsampled with stride 2 and the number of filters is doubled, so that effectively the spatial size decreases while the depth increases (and therefore the number of features that the network can learn). The network is terminated with an average pooling layer, and a 1000 fully connected layer for the 1000 classes of the ImageNet dataset. Overall, He and coworkers constructed ResNet architectures with 34, 50, 101, and 152 layers.

In recent years there have been developed many networks that go beyond ResNet. Some of them are extensions of ResNet (ResNext [52]), or combinations of ResNet with other architectures (Inception-V4 [53]), or networks that do not make use of a residual block and instead use layer dropout (FractalNet [54]). Nevertheless, ResNet is still a state-of-the-art network, and that is why we have used it for the experiments in this work.

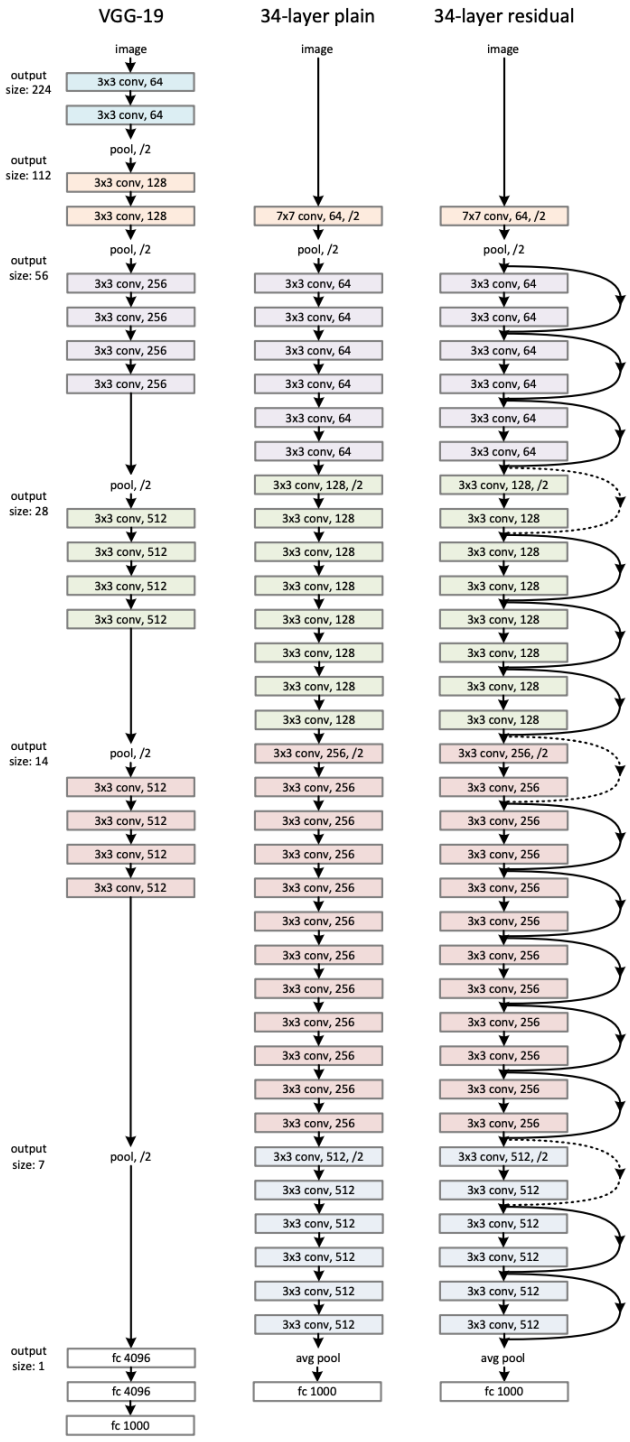

FIGURE 3.6: **Comparison between deep Convolutional Neural Network architectures.** On the left we have a VGG network with 19 layers, in the center a plain ResNet with 34 layers, and on the right a ResNet with 34 layers and residual connections. Figure is adapted from[51].

Chapter 4

Proposed approach

In the previous chapters we have introduced the key components of the approach we followed in our study: on the one hand, we have described existing satellite image datasets and introduced the actual data we will consider, and on the other, we have discussed Deep Learning, how it works and how we can use it for our problem. Now, we are ready to describe our approach: the image feature extraction, the model architecture and the training scenario.

4.1 Image features and transfer learning

In order to train a model based on images, some sort of features need to be extracted. Traditionally, this image feature extraction was based on a set of hand-crafted detectors aimed to detect edges, corners, blobs and other feature descriptors. Some of these detectors are the Sobel filter, Laplacian of Gaussian (LoG), Difference of Gaussians (DoG), Determinant of Hessian (DoH), SIFT [55, 56], SURF [57], Histograms of Oriented Gradients (HOG) [58] and Gabor filters.

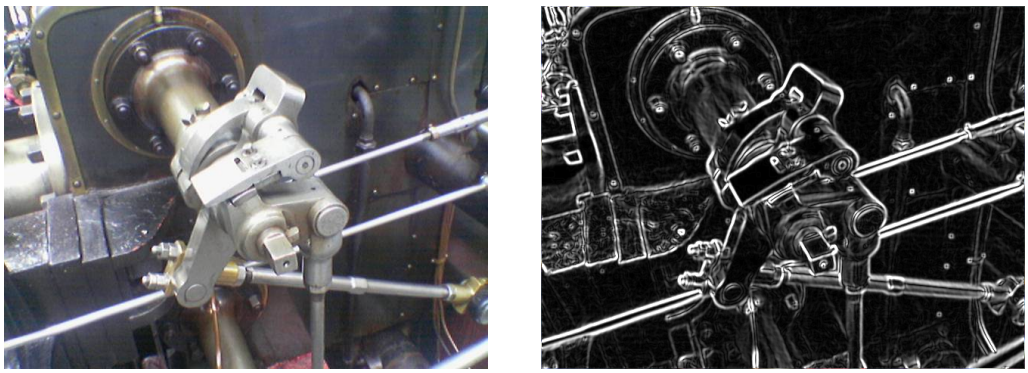

FIGURE 4.1: **Example of the Sobel filter**

More recent approaches to image classification using Neural Networks have benefited from the existing and increasing computational power, and deep Convolutional Neural Networks have been able to achieve higher performances than traditional models.

Yet, training a deep CNN from scratch for a particular problem requires a large and exhaustive dataset along with a huge amount of computational power.

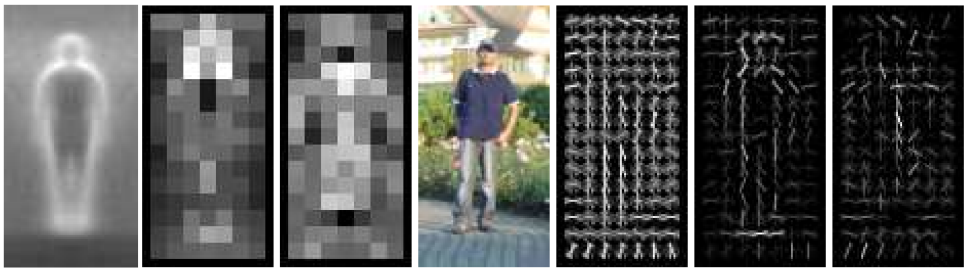

FIGURE 4.2: **Examples of HOG detector [58]**

However, it has been shown that the architectures of pre-trained NN can be reused for other purposes and achieve an equally great performance. This is known as **Transfer Learning**. Figure 4.3 schematizes this idea.

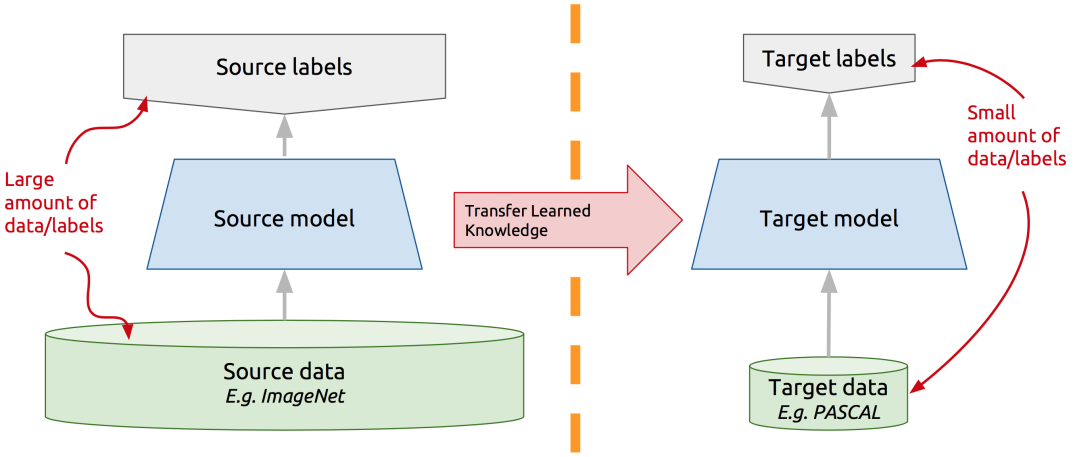

FIGURE 4.3: **Transfer Learning**: a model learned from a large dataset can be transferred and reused for another purpose. [59]

These pre-trained architectures can be re-purposed by reusing the learned weights and either replacing the final layers of the net by some other classifier, or even fine-tuning all the layers for the specific problem. In any case, the initial layers of the Neural Network provide a great image feature extractor.

In the next section we describe our approach using transfer learning from a ResNet architecture.

4.2 Proposed architecture

As described before (Sections 3.3 and 4.1), we can use for our problem a pre-trained ResNet with our own final classification layers. Hence, the architecture we propose for our problem consists on the activation layers of a ResNet, which act as the feature extractors of our images, followed by a shallow classifier made of a single Dense (Fully Connected) layer. Figure 4.4 gives a schema of this approach.

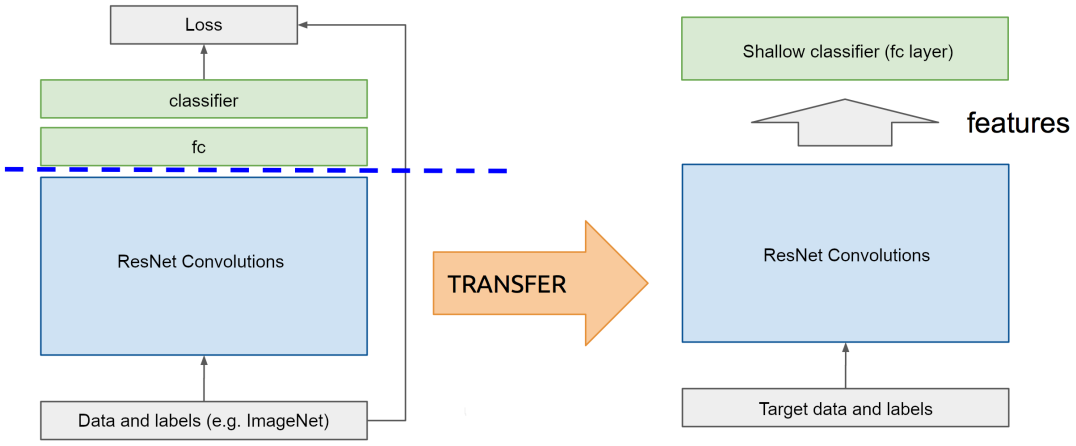

FIGURE 4.4: **Transfer Learning from a ResNet** (figure adapted from [59])

4.2.1 ResNet activations

The ResNet we consider (ResNet50) has a total of 49 activation layers, so the output at each of them is different. Initial layers are able to recognize edges, textures and patterns while keeping an image size similar to the input. On the other hand, deeper activation layers show more convoluted relations and provide much more channels (or filters) by shrinking the image size.

For instance, for an input image of (tensor) size $512 \times 512 \times 3$ (a $512x512$ image with 3 RGB channels), the output of the first activation layer is of size $256 \times 256 \times 64$, the 10^{th} gives a $128 \times 128 \times 256$ tensor, and the last 49^{th} activation layer outputs $16 \times 16 \times 2048$. For our purpose, we will consider the final output of the ResNet (49^{th} activation layer), although this could be further investigated and discussed.

Figures 4.5 - 4.8 show some outputs of the 10^{th} and 49^{th} activation layers for samples of different categories in the dataset. Some of the 10^{th} activations are particularly sensitive to edges, shadows, or textures, which later translate into different outputs at the 49^{th} layer.

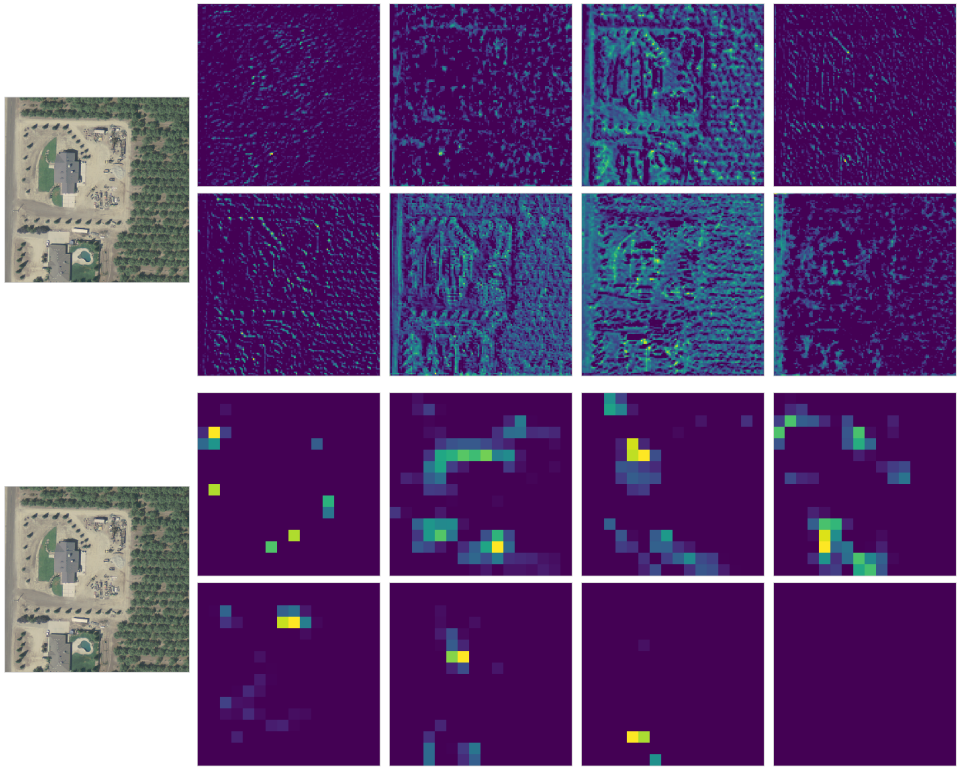

FIGURE 4.5: **ResNet activations of an Agriculture image:** 10^{th} **layer (top) and final layer,** 49^{th} **(bottom).**

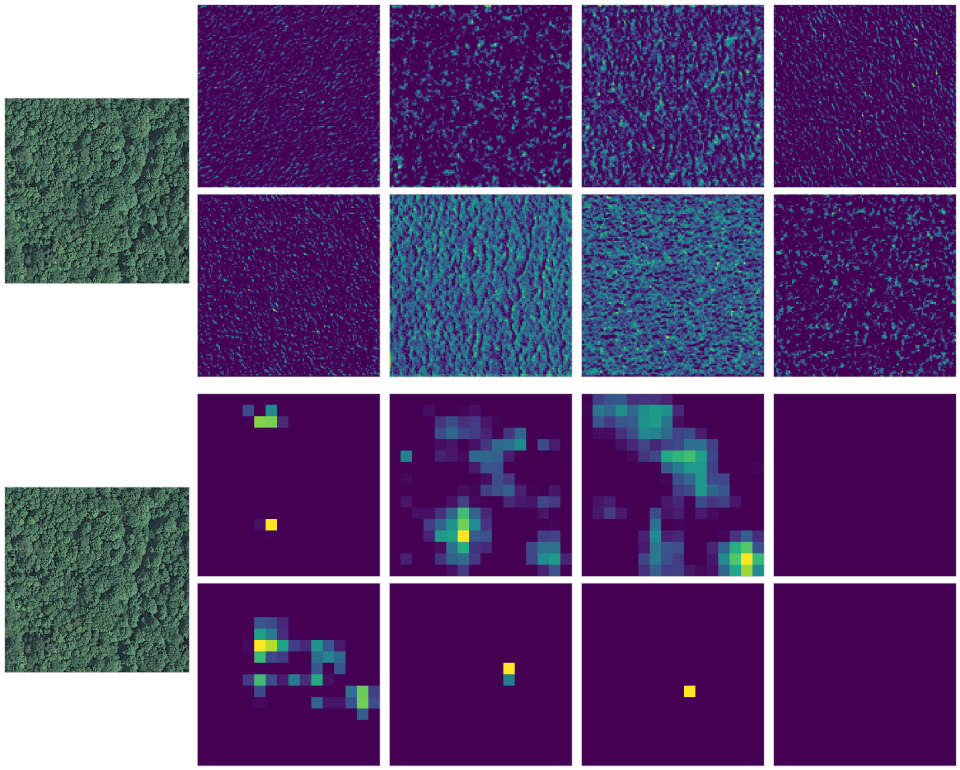

FIGURE 4.6: **ResNet activations of a Forest-woodland image:** 10^{th} **layer (top) and final layer,** 49^{th} **(bottom).**

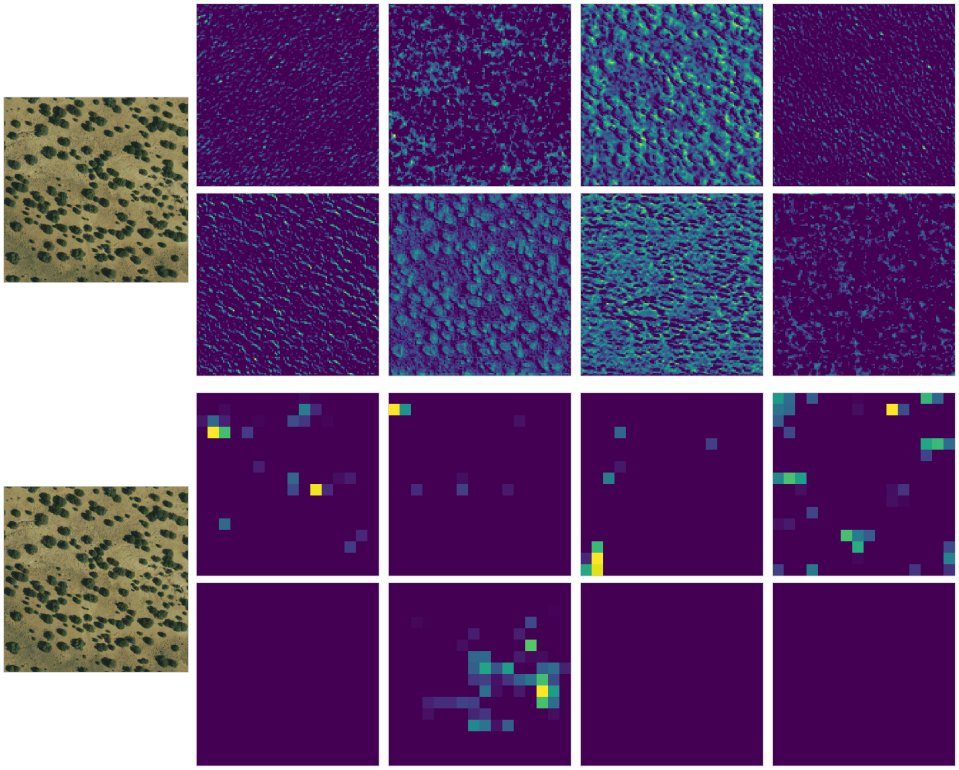

FIGURE 4.7: **ResNet activations of a Semi-desert image: 10^{th} layer (top) and final layer, 49^{th} (bottom).**

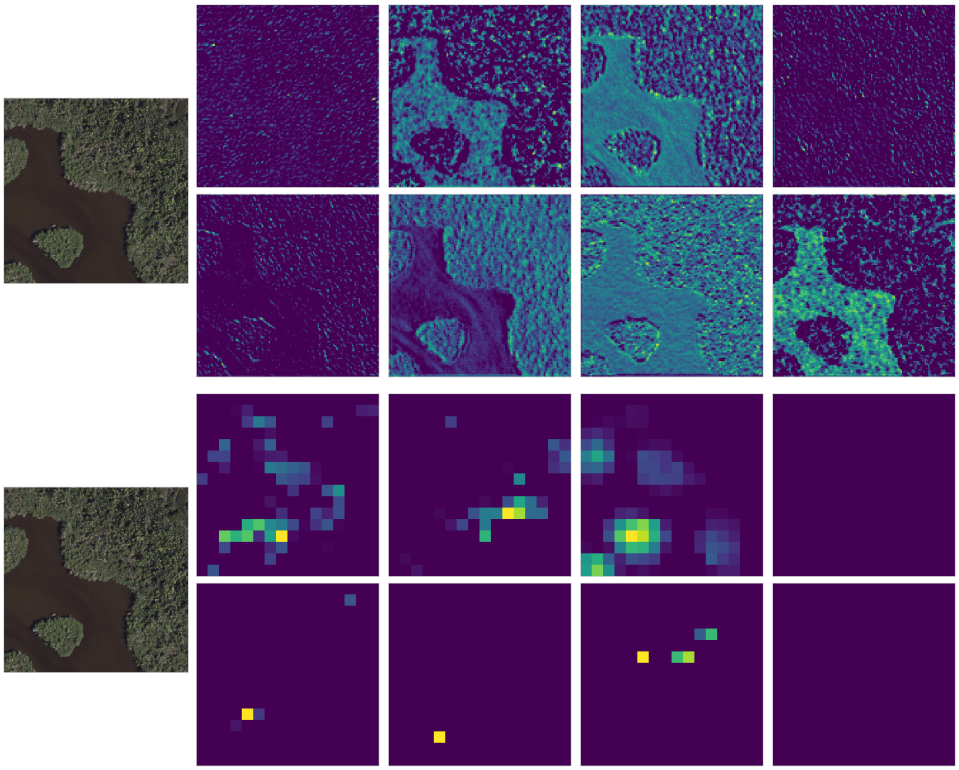

FIGURE 4.8: **ResNet activations of a Shrubland-grassland image: 10^{th} layer (top) and final layer, 49^{th} (bottom).**

4.2.2 Complete architecture

For our purpose we considered the last (49^{th}) activation layer of the ResNet as the features of our images. These features can be extracted and saved on disk in order to speed up the process (as we did), or computed each time, and then passed through a simple Neural Network.

Then, our model consisted on a single dense layer of 100 or 200 neurons with ReLU activation, followed by a single Dense node with a Sigmoid activation acting as the classifier. This model is trained on the dataset with RMSprop optimizer [40] and a binary cross-entropy loss function. The same architecture is used and trained separately for each of the resolutions considered.

This architecture (see Fig. 4.4) has been implemented with Python and Keras. Figure 4.9 shows the model build, while in the following section we describe the complete training pipeline with more detail.

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_1 (Dense)              (None, 100)               52428900
_________________________________________________________________
dense_2 (Dense)              (None, 1)                 101
=================================================================
Total params: 52,429,001
Trainable params: 52,429,001
Non-trainable params: 0
_________________________________________________________________
```

FIGURE 4.9: **Model build with Keras**

4.2.3 Training pipeline and experiments

As introduced in previous chapters, our goal with this model is to study how feasible it is (technically and costly speaking) to detect different kind of human impact on satellite images, and how this detection behaves for different image resolutions. To do so, we build two datasets of annotated images at base resolutions of $0.3m$ and $1m$ (see Chapter 2), which we later downgrade to a range of resolutions suitable for our study.

Starting from an image at its base resolution and size (512×512 pixels), the downgrade process consists of downsampling (removing) pixels, so the image becomes smaller. Therefore, this imposes a limit on how far a given dataset can be downgraded, as the CNN ResNet model requires a minimum input size of 32×32 pixels [60]. Note that during the downsampling process the physical area displayed by the image is not changed. Tables 4.10 and 4.11 show the resolutions and sizes considered for the two datasets.

resolution (m)	0.3	0.6	0.9	1.2	1.5	1.8	2.1	2.4	2.7	3.0	3.3	3.6	3.9	4.2	4.5	4.8
size (pixels)	512	256	171	128	102	85	73	64	57	51	47	43	39	37	34	32

FIGURE 4.10: **Relation between resolution and size for the** $0.3m$ **dataset**. Size is the width (and height) of a square image.

resolution (m)	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
size (pixels)	512	256	171	128	102	85	73	64	57	51	47	43	39	37	34	32

FIGURE 4.11: **Relation between resolution and size for the** $1m$ **dataset**. Size is the width (and height) of a square image.

Next, for each of the datasets and downgraded resolutions we want to train a separate model and assess its performance. To this end, we consider the following pipeline for each of the datasets:

1. Load the original images (at the original resolution) from disk, along with the human impact labels and categories.
2. Downsample the images to the desired resolution (from the lists in 4.10 and 4.11)
3. Compute the ResNet activations (at the 49^{th} activation layer) of the resulting downgraded images. These activations can be saved to disk for later use if needed.
4. Consider a stratified KFold split of the dataset (with 8 splits) for cross-validation. That means, the dataset is split into 8 sets with labels 0-1 equally distributed. Note that label 1 here represents the class with clear human impact, in contrast to the convention in chapter 2 (where it was label 2). Each cross-validation fold consists of around 300 images for the $0.3m$ dataset, and around 200 samples for the $1m$ dataset.
5. Train the model (Fig. 4.9) separately for each combination of the 7 training sets considering 30 epochs. The remaining set is used as a validation set to assess the accuracy. After training on all folds, the results of the 8 experiments are averaged in order to obtain more consistent measures.
6. Repeat the process for all downgraded resolutions.

Note that the splitting parameters of the cross-validation, the model complexity and the training epochs could be further analyzed in order to find the best combination for each of the resolutions tested. Actually, all the experiments consist of training NN models for two datasets, each with 16 resolutions and 8 folds per

resolution, so every fine-tuning (like changing the size of the NN) implies several executions with some variability on each stage.

Nonetheless, as already mentioned in the introduction, the final goal of the experiments is to have an statistical reference of how well the models can be trained, and not to achieve the highest accuracy for each scenario. In order to do so, a larger dataset would be needed, with more well defined categories and objects, and a clear goal of what needs to be modeled.

The plots in Figure 4.12 (in the next page) show the convergence of some of the models trained, for the different datasets and resolutions (and considering one of the cross-validation folds only). Also, two architectures for the dense layer have been tested: 100 neurons and 200 neurons.

In general, the NN is able to converge and achieve a good accuracy (as shown in the plots), although for some particular splits of the data, it fails to converge and stays in a low accuracy point. This is probably due to the fact of having a relative small dataset. Hence, these particular folds are not going to be considered when computing the final averaged results.

In the next chapter we discuss in much more detail the results obtained from all these experiments.

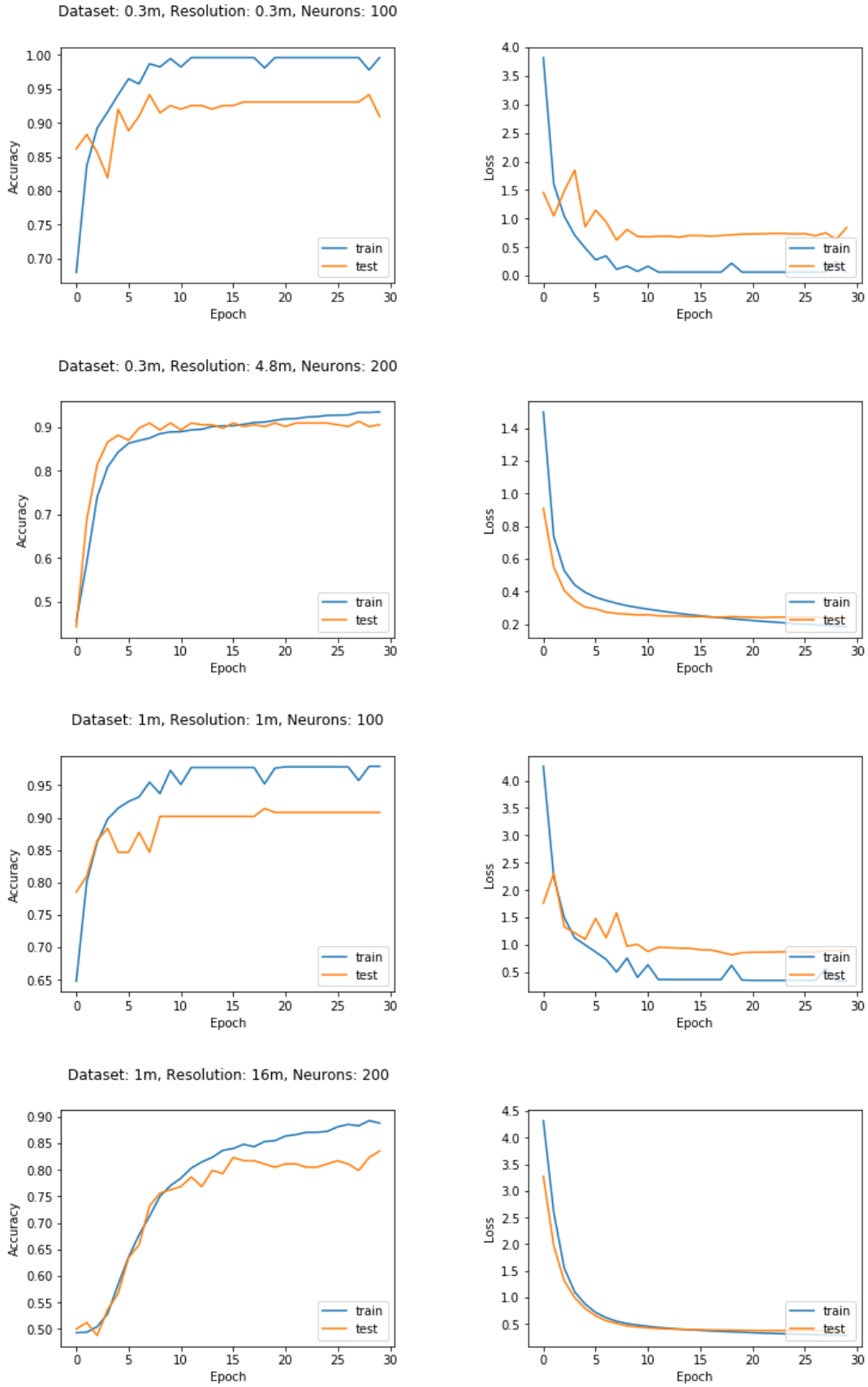

FIGURE 4.12: **Convergence plots some experiments on different datasets, resolutions and architectures.**

Chapter 5

Results

In this chapter we discuss the results obtained with our models. First, we analyze how the model works in a few given resolutions, so that we can be confident about its performance. Then, we consider how the accuracy changes with the resolution and within the categories. Finally, we discuss the cost around Satellite imagery to analyze the world.

5.1 Transfer learning on aerial imagery

The first thing we might want to investigate from our experiments is whether the approach followed is able to achieve good results. That is, we want to evaluate if using a pre-trained ResNet as a feature extractor for aerial images allows the trained model to properly discriminate the existence of human impact.

Tables A.1 - A.4 in Appendix A show that the accuracies obtained with all the experiments are indeed remarkable. All these results are discussed in more detail in the next section, but let us begin by focusing on few cases of the $0.3m$ dataset in order to understand how the models are behaving. In table A.1 we can see that an accuracy of around 0.9226 is achieved on the base resolution ($0.3m$), while it drops to 0.8690 of the last downgraded resolution of the dataset ($4.8m$).

Let us consider first some examples of correctly and wrongly classified images at the base resolution (for one of the cross-validation folds), which are shown in Figures 5.1 and 5.2 respectively. The first set of samples shows that the model accurately detects clear human impact related to agriculture (2nd picture in the second row) and paths. On the other hand, the second set shows that it might fail to detect it when the impact is subtle, covering a small region of the image, or when it can even be confused with natural structures (or vice versa).

Note that, from this point, *label 1* refers to images with clear human impact, which were defined as *label 2* when building the datasets in Chapter 2.

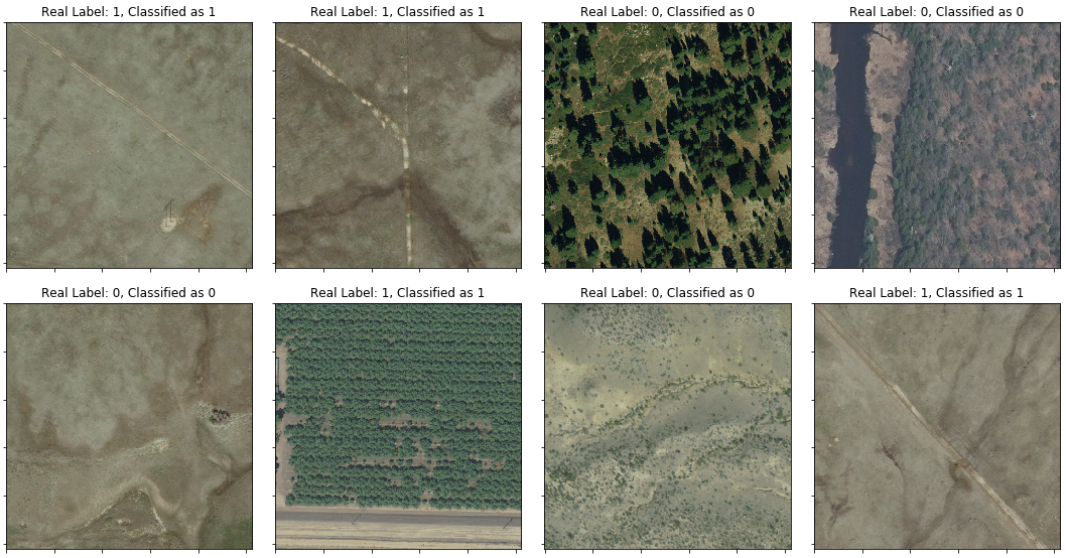

FIGURE 5.1: **Examples of correctly classified images at the base resolution,** $0.3m$**.**

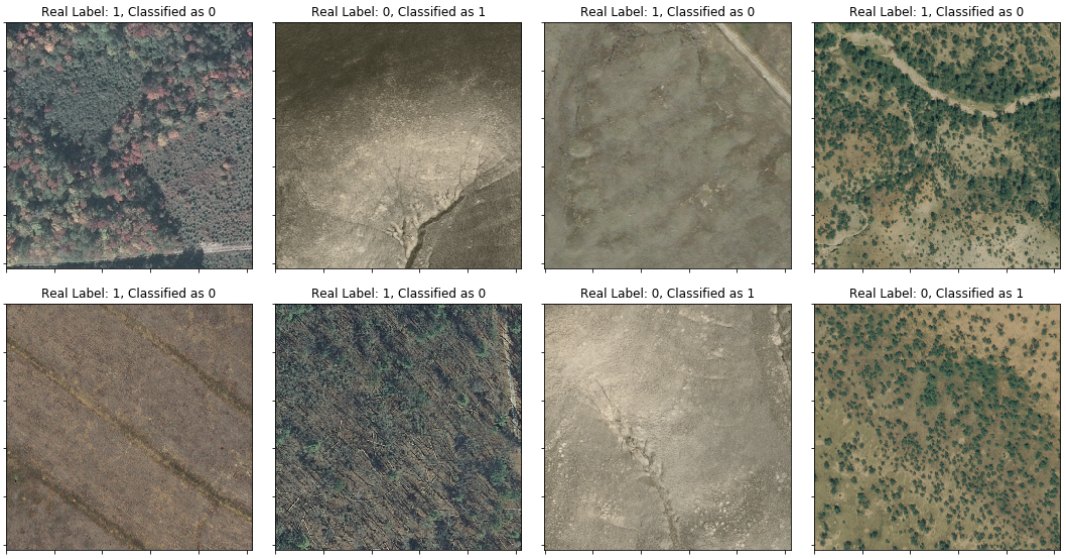

FIGURE 5.2: **Examples of wrongly classified images at the base resolution,** $0.3m$**.**

The same kind of analysis can be done for the last resolution, $4.8m$. Figures 5.3 and 5.4 show correctly and wrongly classified images at this resolution. The first set shows that the model detects human impact when it is still evident, even with the low resolution, but the second set indicates that it commits some mistakes when the human impact evidences are lost with the downgrade process. Similarly, it might classify as man-made structures patterns that are indeed natural.

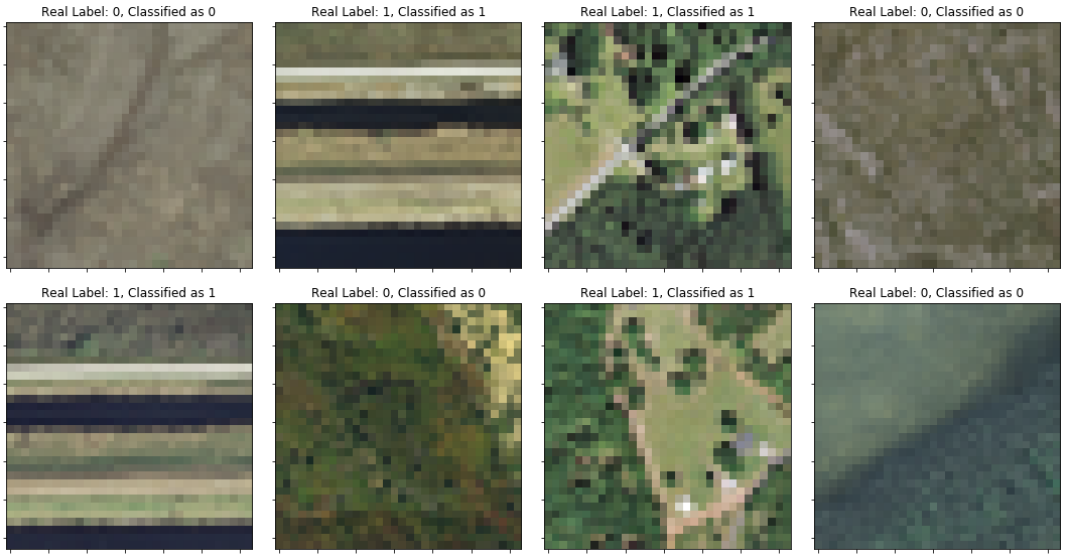

FIGURE 5.3: **Examples of correctly classified images at the last resolution,** $4.8m$**.**

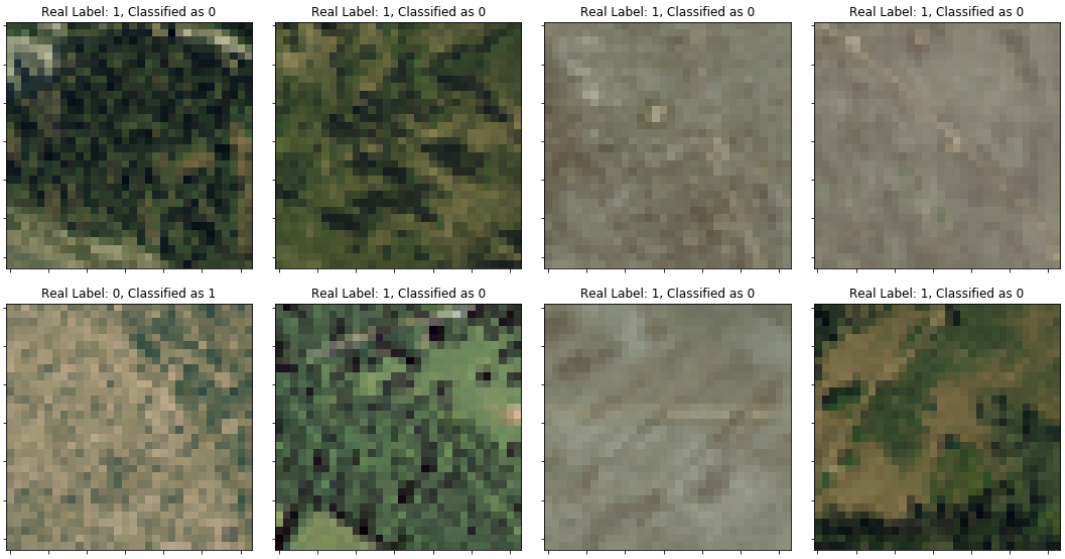

FIGURE 5.4: **Examples of wrongly classified images at the last resolution,** $4.8m$**.**

This can also be seen with Figure 5.5, which shows some correctly classified images at $0.3m$ resolution (top row) that are wrongly classified at $4.8m$. The first and third pair of images demonstrate that, when the human impact is subtle, it missed in the downgraded resolution. Conversely, non human activity can also be misclassified at lower resolutions (second and fourth images).

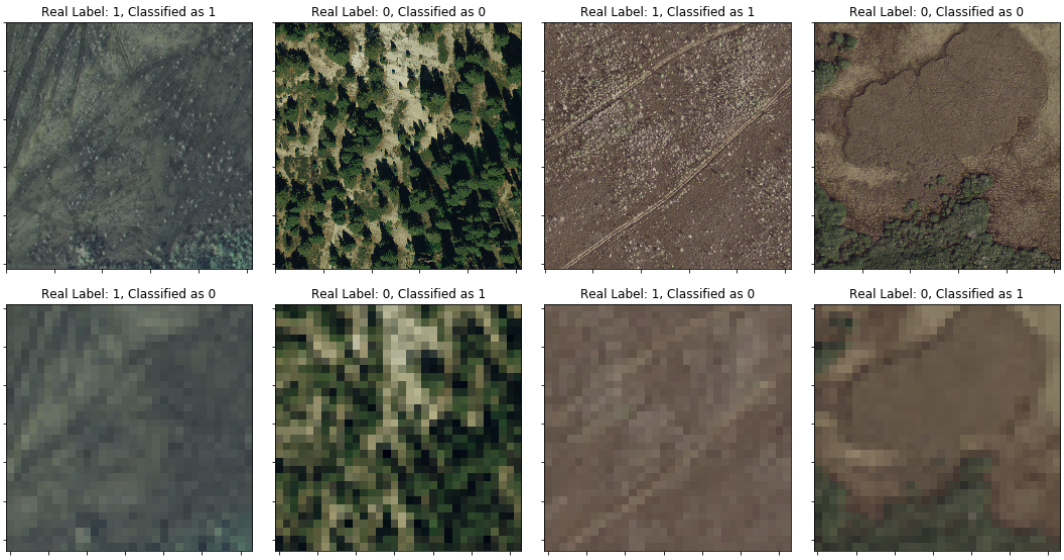

FIGURE 5.5: **Correctly and wrongly classified images at resolutions** $0.3m$ **and** $4.8m$**, respectively.**

Finally, we can investigate how the model behaves with images where human impact is really subtle. For this purpose, we consider the images with the intermediate label (*label 1* in Chapter 2, Figure 2.6). The model has never been faced with these images, so this can give a good perception of whether the models has been able to learn relevant features of human impact. Figure 5.6 shows some of these images with the label predicted by the model in the title. Even if man-made structures in these pictures are small, the model is able to detect straight lines and shapes as human activity.

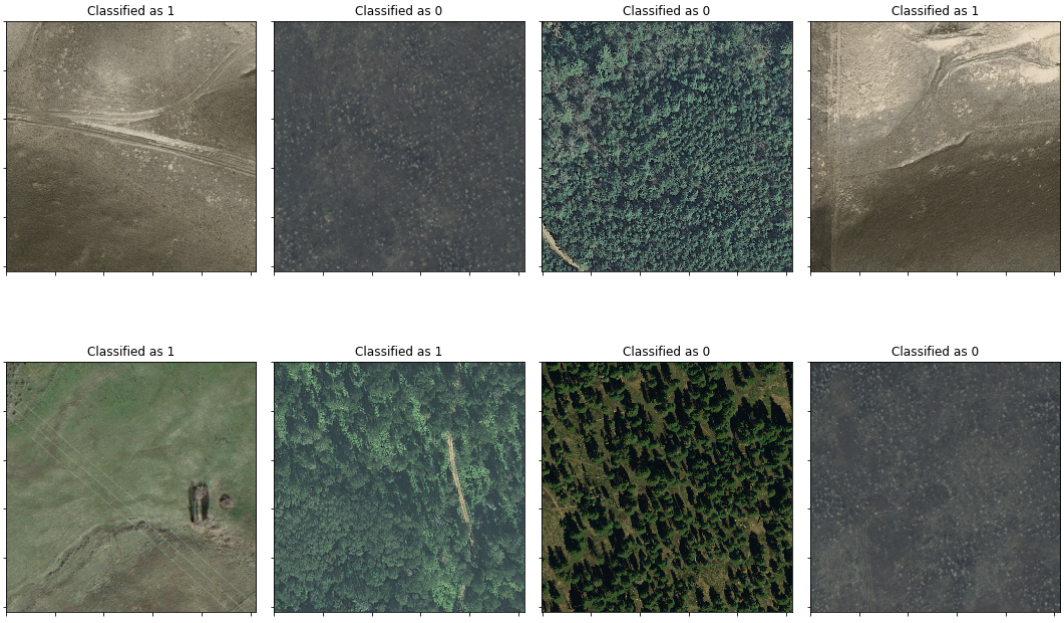

FIGURE 5.6: **Examples of predictions made for images with subtle human impact** ***label 1*** **in Chapter 2.**

All in all, we can conclude that the model is able to achieve a great accuracy, specially on the Agriculture related pictures. It is able to generalize to unseen, subtle images, and still be accurate for lower resolutions, were the amount of pixels and information per images become much smaller.

5.2 Man-made structures detection at different scale

Now that we know the models build are able to correctly detect, with high accuracy, human impact from our datasets, we are ready to analyze the results for different resolutions.

From all the experiments (datasets, architectures, resolutions and cross-validations), the results have been stored and aggregated. As mentioned in the previous chapter, the models were not able to converge for some particular splits of the datasets. Hence, these few folds have been ignored when aggregating the results. Tables A.1 - A.4 in Appendix A summarize the accuracies obtained for each of the datasets ($0.3m$ and $1m$) and downgraded resolutions, both the overall accuracy and by category.

Similarly, Figure 5.7 shows the overall accuracies obtained for all resolutions. From this we can see that similar accuracies are achieved for both datasets on the same degraded resolution, which means that both datasets are comparable and can be considered together. Also, we realize that increasing the size of the model from 100 neurons to 200 does not have a big impact on the accuracy, but tends to perform slightly better. Hence, for other results later we will focus on this architecture only.

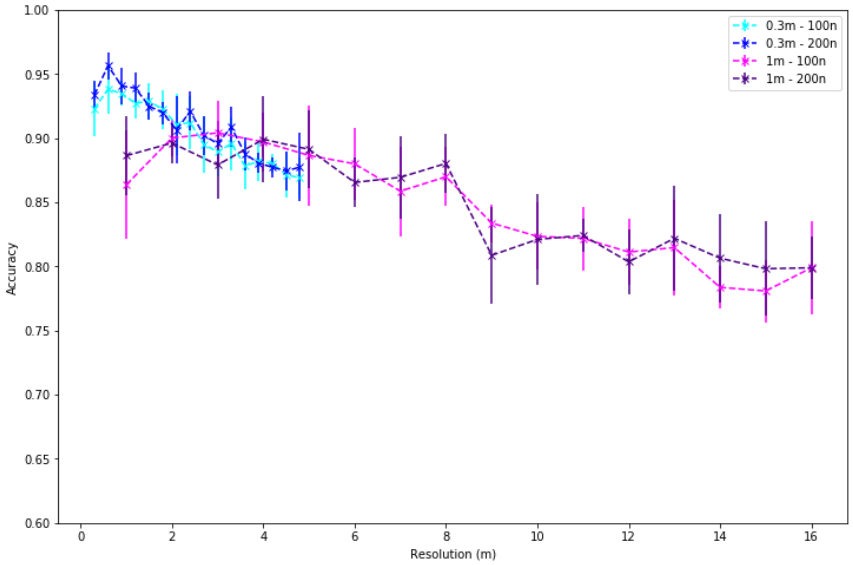

FIGURE 5.7: **Accuracy at each resolution for all datasets and architectures.** Similar results are obtained at the same degraded resolutions. The vertical lines represent the variability (standard deviation) for the different folds of the cross-validations.

On closer inspection, we also detect that the accuracy on the base resolution ($0.3m$ and $1m$) is always slightly worse than the next resolution. This is probably because the input image size at the base resolution is quite large (512×512), which makes the dense layer much more complex to train. More sample images would be required in order to compensate for the complexity and achieve a greater accuracy.

Finally, the overall conclusion from this plot is that, as expected, better resolutions (less than $2m$) allow for greater accuracies, of over 90%, which means a great success considering that the images in the datasets are balanced between having or not human impact. Furthermore, accuracy is still good for resolutions between $2m$ and $8m$, staying between 85% and 90%. From $8m$ onwards, accuracy drops to 80% and the model is not able to detect more subtle ways of man-made structures.

Now let us consider how these accuracies behave for each of the land use categories in the datasets. As discussed in section 2.4.1, these categories are rough approximations of the kind of terrain and human impact, with images that could be exchanged between categories, but overall these can give and idea of the accuracies when analyzing different kind of terrains.

Indeed, Figures 5.8 and 5.9 show that, for the $0.3m$ dataset (and 200 neurons model), accuracy differs substantially between categories. Fig. 5.8 shows the global comparison between categories, while Fig. 5.9 allows for a better comparison of each category with respect to the overall accuracy.

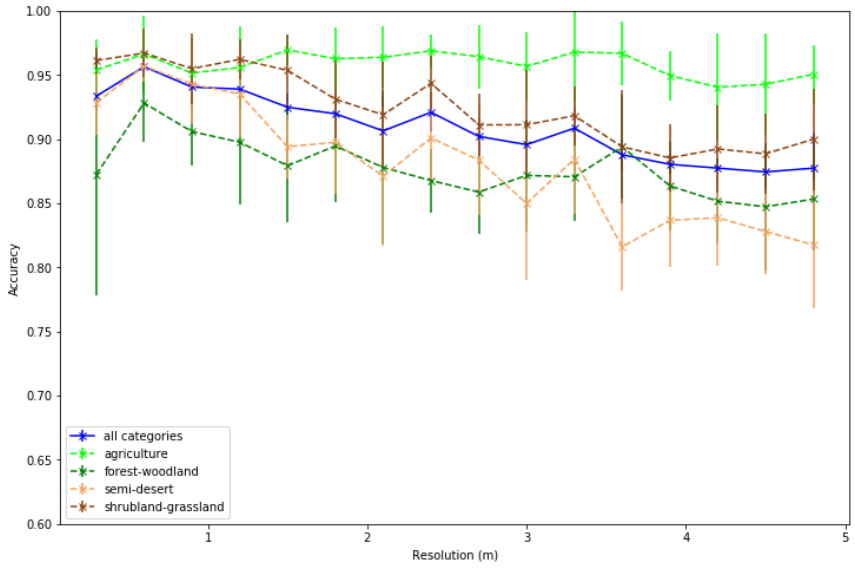

FIGURE 5.8: **Accuracies obtained for each category, with the model trained over all categories, on the $0.3m$ dataset.**

These plots have been obtained once the model was trained for all the categories. Then, the accuracy on the validation set was computed for each individual category and over all images in the set. The validation set in each iteration of the cross-validation was small, consisting on few hundred images, and an homogeneous

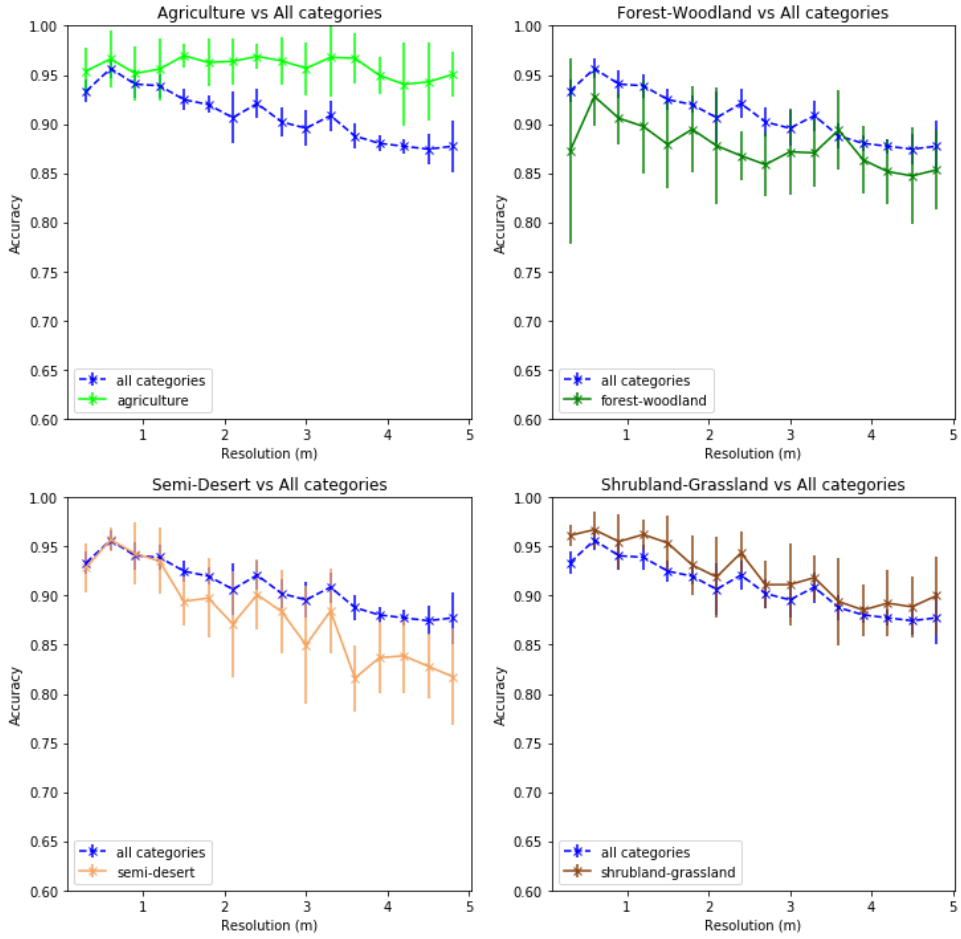

FIGURE 5.9: **Comparsion of accuracies between each category and overall, with the model trained over all categories, on the** $0.3m$ **dataset.**

representation of the categories was not imposed (validation samples were picked randomly, only preserving proportion between having or not human impact), which explains the large variability for each experiment (vertical lines in the plots).

Although the category is not taken into account when training, we can see that the models are capable of detecting agriculture-related human impact with high accuracy (over 95%), without really being affected by the drop in accuracy. Shrubland-grassland category also has a good accuracy, while the model performs worse in semi-desert and forest-woodland images. That means that the models are able to capture textures and patterns related to agriculture quite easily, while the other categories have more variable features.

Similar results are obtained for the $1m$ dataset, which are shown in Figures 5.10 and 5.11. The models are able to achieve a great accuracy when detecting agriculture-related human impact (over 95%), independently of the resolution considered.

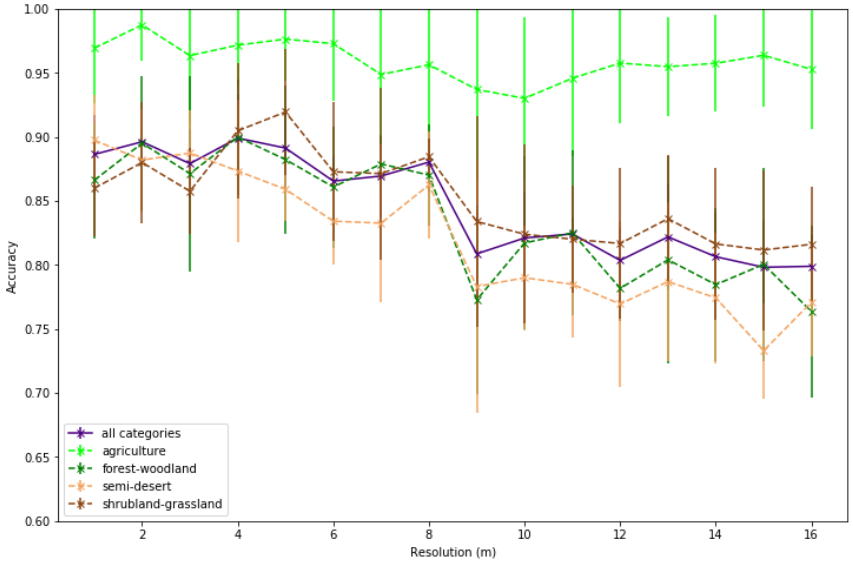

FIGURE 5.10: **Accuracies obtained for each category, with the model trained over all categories, on the $1m$ dataset.**

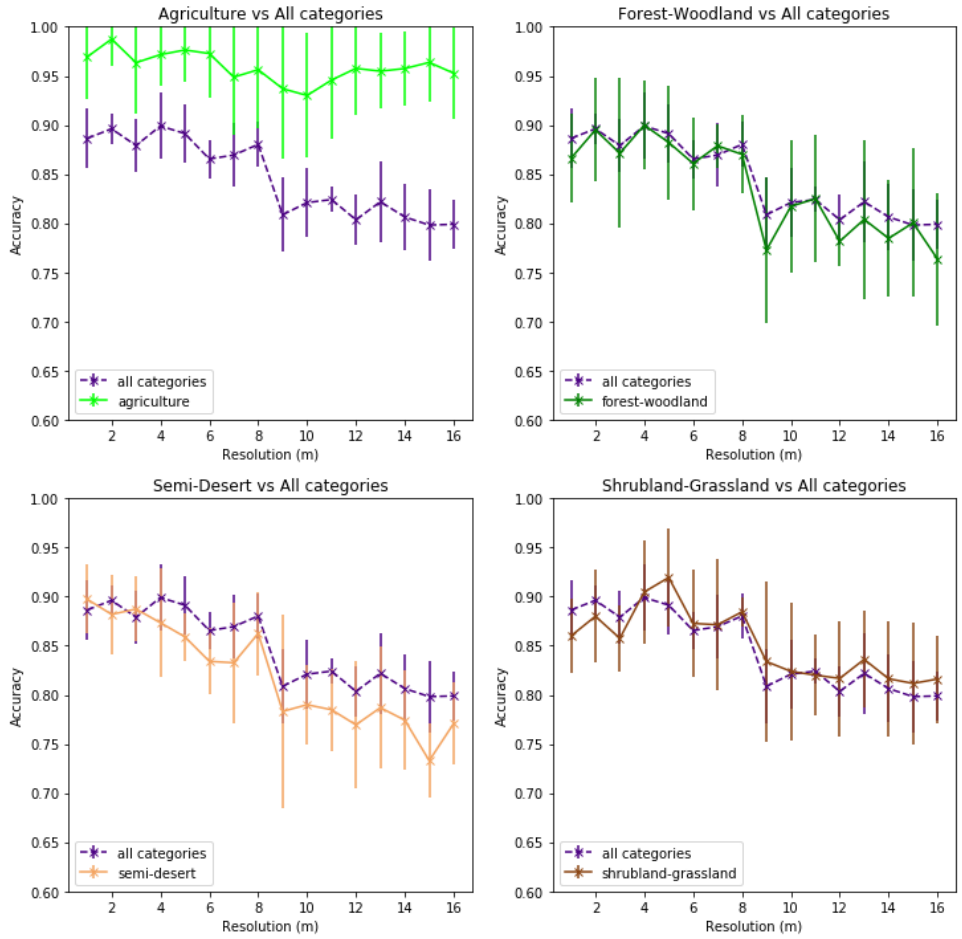

FIGURE 5.11: **Comparsion of accuracies between each category and overall, with the model trained over all categories, on the $1m$ dataset.**

5.3 Cost estimation

We discuss the financial cost associated to building and launching a satellite, and to renting infrastructure for performing the entire image processing pipeline. We further study the cost as a function of pixel resolution. However, our estimates are very rough approximations because many factors are involved and large variations occur between them. To give an example, choosing one material over the other might change the cost of manufacturing and launching a satellite by one order of magnitude. It is also completely different to have a satellite for 3 years in space, or to target a lifespan of 20 years.

Having this in mind, we follow laws from physics to estimate the dependency of the satellite cost on resolution. First, the cost of launching a satellite into the orbit scales linearly with its mass, which is given by the amount of fuel needed. Second, the mass of the satellite scales quadratic with resolution so that overall we obtain a cubic dependency of financial cost on resolution. The latter increase in cost is associated with the optical instruments used. As a reference for the satellite cost we use a Skysat satellite from Planet **[skysat_planet]** that has a resolution of about 1m and a value of $30 million. This amount was provided to us by Satellogic and includes construction, launch and maintenance during the satellite's lifespan.

Our final goal is to give an estimation of the expenditure to monitor once the entire surface of the earth (about 149 million km^2). To this end, we multiply the satellite cost by the ratio: time needed to scan the earth over the satellite's lifespan. Further, a satellite can map 1 million km^2 at 1m resolution in 4.2 days [16]. We hence can calculate the satellite cost per km^2. With $area\ per\ lifespan = 10^6 \times \frac{10 \cdot 365}{4.2}\ km^2$ we obtain $cost\ satellite\ per\ km^2 = cost\ satellite/area \approx 0.035\ \$/km^2$.

description	cost	unit	cost ($/km^2)	cost ($/pixel)
process raw data			0.004	4×10^{-9}
hot storage	72×10^{-6}	$/(km^2/month)	0.000864	8.64×10^{-10}
cold storage	36×10^{-6}	$/(km^2/month)	0.000432	4.32×10^{-10}
archive storage	9×10^{-6}	$/(km^2/month)	0.000108	1.08×10^{-10}
download data	8	$/Gb	0.021	2.1×10^{-8}
serving to final client	0.09	$/Gb	0.000236	4.7232×10^{-10}
prediction (AWS)	0.05 & $\sim$6	$/h & s/km^2	0.00145	1.45×10^{-9}

TABLE 5.1: **Costs for image data processing.** All costs except the prediction are provided by Satellogic.

Another cost intensive block when capturing satellite imagery involves image data processing for which the cost scales quadratic with resolution. For example, an operation that costs 100$/km2 at 1m resolution will cost only 1$/km^2 at 10m resolution. The data processing step consists of multiple parts: transformation of raw data into image pixels, storing data in a hot, cold, and archive storage, downloading data from the satellite, serving it to the final client, and in our case predicting human impact. These costs are summarized for 1m resolution in table

5.1. Note that we used the conversion factor 0.002624 for an image to convert from Gigabytes to km^2 (2$\times$ compressed) and we assume 12 months of data storage. The prediction step is estimated by loading 4 images that each have an area of about $500 \times 500m^2$, calculating the ResNet activations of the final layer, and predicting the class using the models trained in chapter 4 in an ensemble fashion. This part amounts to a processing time of 6s for an area of 1km^2, which can be converted into costs per km^2 assuming 0.05\$/h of AWS EC2 compute [61].

To finally obtain the dependence of the resolution on the total financial cost we sum the data cost per km^2 and the satellite cost per km^2 at 1m resolution, and convert to cost per pixel ($\times 10^{-6}$). We then multiply with the number of pixels necessary to cover the entire surface of the earth. Here the satellite cost per km^2 is a cubic function and the earth surface in pixel is a quadratic function in resolution. The result is shown in Fig. 5.12. We obtain a cost of about \$15 million dollars at 1m resolution with a very step slope towards better resolutions. At 0.3m resolution the cost is two orders of magnitude higher than at 1m while for worse resolutions the cost decreases by two orders of magninute when the cost is a factor 10 larger. We conclude that for worse resolutions the data processing cost is the dominating cost whereas for very good resolutions the satellite cost dominates.

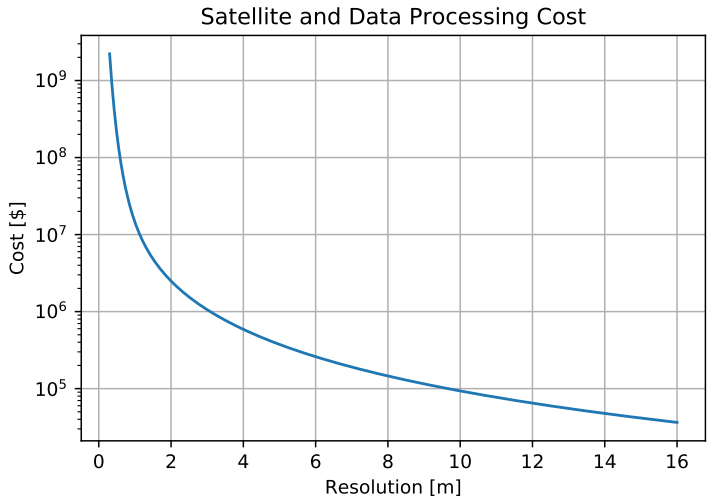

FIGURE 5.12: **Satellite and data processing cost.** The total financial cost to capture images with a satellite and process the data as function of resolution.

Chapter 6

Conclusions

6.1 The Problem

Lorem ipsum

6.2 The Datasets

Lorem ipsum

6.3 The Models

Lorem ipsum

6.4 Further work

- Dataset: improve it, enlarge with more and more variate images, better classify them, even anotate with more categories of actual objects appearing, consider annotating areas, and train a model to detect this particular patterns. Yet, all this would diverge from the original scope of the project,
- Model: CNN feature extraction, other architectures and number of activations, layers, neurons, training epochs. Make a more robust pipeline, load images on batches, so a larger dataset can be considered. Implement some sort of data augmentation if needed
- Results: further study on which images the models fail, what kind of information is learning (patters, colors, shapes, etc)
- Conclusions: further analysis costs of such implementations, environment impact, etc

Appendix A

Tables

The following tables contain the aggregated results of the cross-validations performed for each dataset and downgraded resolution. Folds where the model was not able to converge have been removed.

resolution (m)	All categories		agriculture		forest-woodland		semi-desert		shrubland-grassland	
	mean acc.	std	mean acc.	std	mean acc.	std	mean acc.	std	mean acc.	std
0.3	0.9226	0.0211	0.9850	0.0248	0.8630	0.0624	0.8882	0.0453	0.9561	0.0228
0.6	0.9383	0.0190	0.9699	0.0216	0.8779	0.0269	0.9231	0.0410	0.9748	0.0198
0.9	0.9347	0.0096	0.9439	0.0244	0.8944	0.0561	0.9327	0.0216	0.9578	0.0212
1.2	0.9271	0.0120	0.9488	0.0374	0.9061	0.0373	0.9150	0.0328	0.9399	0.0231
1.5	0.9288	0.0139	0.9622	0.0285	0.8679	0.0383	0.9144	0.0309	0.9604	0.0215
1.8	0.9224	0.0154	0.9678	0.0311	0.8864	0.0650	0.9014	0.0363	0.9371	0.0221
2.1	0.9108	0.0241	0.9618	0.0168	0.8749	0.0454	0.8885	0.0571	0.9216	0.0371
2.4	0.9120	0.0206	0.9573	0.0280	0.8686	0.0590	0.8867	0.0358	0.9352	0.0263
2.7	0.8952	0.0223	0.9620	0.0216	0.8631	0.0191	0.8510	0.0350	0.9117	0.0496
3.0	0.8893	0.0189	0.9571	0.0067	0.8717	0.0197	0.8340	0.0472	0.9055	0.0455
3.3	0.8957	0.0209	0.9539	0.0311	0.8723	0.0479	0.8529	0.0341	0.9121	0.0425
3.6	0.8784	0.0184	0.9415	0.0394	0.8727	0.0442	0.8199	0.0348	0.8935	0.0299
3.9	0.8819	0.0153	0.9456	0.0186	0.8738	0.0330	0.8397	0.0364	0.8796	0.0178
4.2	0.8804	0.0070	0.9389	0.0305	0.8525	0.0267	0.8444	0.0533	0.8961	0.0222
4.5	0.8715	0.0179	0.9383	0.0390	0.8445	0.0308	0.8301	0.0340	0.8821	0.0206
4.8	0.8690	0.0160	0.9508	0.0157	0.8569	0.0517	0.7798	0.0456	0.9035	0.0310

TABLE A.1: **Aggregated cross-validation results for** $0.3m$ **dataset,** 100 **neurons**

resolution (m)	All categories		agriculture		forest-woodland		semi-desert		shrubland-grassland	
	mean acc.	std	mean acc.	std	mean acc.	std	mean acc.	std	mean acc.	std
0.3	0.9335	0.0113	0.9540	0.0234	0.8725	0.0946	0.9286	0.0253	0.9613	0.0105
0.6	0.9565	0.0104	0.9663	0.0296	0.9281	0.0304	0.9569	0.0120	0.9671	0.0189
0.9	0.9406	0.0139	0.9515	0.0273	0.9059	0.0264	0.9430	0.0314	0.9550	0.0273
1.2	0.9390	0.0122	0.9559	0.0317	0.8976	0.0483	0.9351	0.0336	0.9622	0.0153
1.5	0.9249	0.0106	0.9696	0.0116	0.8793	0.0443	0.8942	0.0250	0.9536	0.0280
1.8	0.9198	0.0087	0.9627	0.0243	0.8947	0.0439	0.8975	0.0404	0.9312	0.0304
2.1	0.9065	0.0263	0.9638	0.0238	0.8780	0.0597	0.8711	0.0541	0.9190	0.0415
2.4	0.9209	0.0152	0.9689	0.0129	0.8677	0.0247	0.9009	0.0354	0.9435	0.0211
2.7	0.9021	0.0150	0.9644	0.0248	0.8587	0.0327	0.8837	0.0426	0.9111	0.0247
3.0	0.8957	0.0181	0.9568	0.0269	0.8717	0.0440	0.8499	0.0597	0.9112	0.0424
3.3	0.9086	0.0156	0.9679	0.0397	0.8707	0.0345	0.8844	0.0426	0.9183	0.0230
3.6	0.8878	0.0125	0.9669	0.0250	0.8940	0.0402	0.8160	0.0339	0.8940	0.0442
3.9	0.8804	0.0078	0.9495	0.0191	0.8635	0.0345	0.8368	0.0365	0.8853	0.0264
4.2	0.8774	0.0078	0.9405	0.0422	0.8516	0.0330	0.8387	0.0374	0.8923	0.0343
4.5	0.8745	0.0150	0.9428	0.0397	0.8473	0.0493	0.8280	0.0328	0.8886	0.0309
4.8	0.8774	0.0265	0.9505	0.0228	0.8533	0.0404	0.8176	0.0497	0.9000	0.0396

TABLE A.2: **Aggregated cross-validation results for** $0.3m$ **dataset,** 200 **neurons**

resolution (m)	All categories		agriculture		forest-woodland		semi-desert		shrubland-grassland	
	mean acc.	std	mean acc.	std	mean acc.	std	mean acc.	std	mean acc.	std
1	0.8636	0.0425	1.0000	0.0000	0.8387	0.0489	0.8561	0.0432	0.8541	0.0646
2	0.9001	0.0138	0.9803	0.0306	0.8991	0.0634	0.8843	0.0413	0.8881	0.0430
3	0.9041	0.0246	0.9846	0.0246	0.9245	0.0280	0.8742	0.0537	0.8805	0.0315
4	0.8970	0.0227	0.9846	0.0246	0.8743	0.0280	0.8786	0.0481	0.9129	0.0222
5	0.8865	0.0391	0.9809	0.0328	0.8692	0.0577	0.8821	0.0328	0.8797	0.0472
6	0.8800	0.0282	0.9673	0.0352	0.8685	0.0467	0.8419	0.0469	0.9093	0.0508
7	0.8587	0.0350	0.9584	0.0482	0.8360	0.0653	0.8468	0.0583	0.8682	0.0598
8	0.8699	0.0230	0.9831	0.0289	0.8788	0.0442	0.8432	0.0303	0.8498	0.0335
9	0.8338	0.0148	0.9341	0.0562	0.8330	0.0565	0.8086	0.0431	0.8245	0.0543
10	0.8234	0.0263	0.9911	0.0253	0.8159	0.0553	0.7815	0.0420	0.8199	0.0700
11	0.8219	0.0249	0.9703	0.0645	0.8052	0.0507	0.7901	0.0538	0.8193	0.0702
12	0.8112	0.0257	0.9490	0.0333	0.8178	0.0362	0.7606	0.0363	0.8118	0.0814
13	0.8147	0.0369	0.9666	0.0488	0.7988	0.0921	0.7719	0.0723	0.8244	0.0572
14	0.7837	0.0167	0.9351	0.0690	0.7682	0.0403	0.7398	0.0543	0.7999	0.0397
15	0.7808	0.0245	0.9457	0.0289	0.7663	0.0673	0.7257	0.0472	0.8017	0.0439
16	0.7990	0.0361	0.9403	0.0635	0.7680	0.0700	0.7713	0.0857	0.8141	0.0764

TABLE A.3: **Aggregated cross-validation results for** $1m$ **dataset,** 100 **neurons**

resolution (m)	All categories		agriculture		forest-woodland		semi-desert		shrubland-grassland	
	mean acc.	std	mean acc.	std	mean acc.	std	mean acc.	std	mean acc.	std
1	0.8863	0.0305	0.9693	0.0432	0.8661	0.0452	0.8974	0.0352	0.8599	0.0377
2	0.8962	0.0156	0.9875	0.0280	0.8951	0.0526	0.8819	0.0401	0.8800	0.0472
3	0.8791	0.0267	0.9635	0.0519	0.8713	0.0761	0.8873	0.0331	0.8576	0.0336
4	0.8991	0.0339	0.9717	0.0317	0.8999	0.0453	0.8734	0.0555	0.9047	0.0529
5	0.8913	0.0301	0.9764	0.0324	0.8823	0.0577	0.8591	0.0243	0.9195	0.0495
6	0.8655	0.0194	0.9728	0.0450	0.8609	0.0472	0.8341	0.0339	0.8728	0.0542
7	0.8695	0.0321	0.9487	0.0588	0.8789	0.0221	0.8327	0.0616	0.8713	0.0669
8	0.8803	0.0230	0.9563	0.0592	0.8702	0.0399	0.8625	0.0422	0.8845	0.0144
9	0.8087	0.0378	0.9370	0.0706	0.7727	0.0738	0.7832	0.0990	0.8339	0.0819
10	0.8211	0.0352	0.9301	0.0634	0.8172	0.0676	0.7900	0.0409	0.8239	0.0698
11	0.8242	0.0130	0.9458	0.0604	0.8250	0.0646	0.7849	0.0416	0.8201	0.0414
12	0.8036	0.0252	0.9577	0.0473	0.7817	0.0252	0.7695	0.0649	0.8168	0.0588
13	0.8219	0.0408	0.9548	0.0384	0.8040	0.0809	0.7870	0.0621	0.8361	0.0494
14	0.8064	0.0343	0.9575	0.0374	0.7846	0.0595	0.7743	0.0508	0.8164	0.0592
15	0.7982	0.0368	0.9638	0.0397	0.8004	0.0756	0.7330	0.0374	0.8117	0.0625
16	0.7989	0.0247	0.9528	0.0466	0.7636	0.0673	0.7708	0.0419	0.8161	0.0446

TABLE A.4: **Aggregated cross-validation results for** $1m$ **dataset,** 200 **neurons**

Appendix B

Files and Code

All the files used in this project, including the image datasets build and code generated, are available online:

- The images datasets are published in Google Drive (see link or reference [27]).
- All the code produced and used in these analysis is available in a GitHub repository (see link or reference [18]). It includes Python libraries generated, scripts and Jupyter Notebooks.

Appendix C

Author contributions

This thesis is a group project between Eduard Ribas Fernández and Peter Weber. Here we will describe the individual contributions of each author. Overall, both authors have contributed to all parts in this project with different weights in each part.

In the first major block, the generation of the dataset, the distribution is as follows. The image search and download of the raw images was performed by P. Weber, while programming the image processing pipeline was done by both authors with similar weight. The labeling of the processed images was also done by both authors.

In the second major block, the data analysis pipeline, P. Weber has higher contribution at the beginning of the pipeline i.e. prototyping first solutions using transfer learning. E. Ribas has higher contribution towards the end of the pipeline. This includes optimizing the code for Colab, tuning the hyperparamters, performing analysis per category and producing the final figures. Regarding estimation of the cost, both authors have equal contribution.

The same applies for preparing all documents related to this project (Github repository, high-level overview, thesis document), both authors have equally contributed.

Bibliography

[1] Peter Kareiva et al. “Domesticated Nature: Shaping Landscapes and Ecosystems for Human Welfare”. In: *Science* 316.5833 (2007), pp. 1866–1869. ISSN: 0036-8075. DOI: 10.1126/science.1140170. eprint: https://science.sciencemag.org/content/316/5833/1866.full.pdf. URL: https://science.sciencemag.org/content/316/5833/1866.

[2] Tim Newbold et al. “Global effects of land use on local terrestrial biodiversity”. In: *Nature* 520 (Apr. 2015), 45 EP –. URL: https://doi.org/10.1038/nature14324.

[3] Robert Costanza et al. “Changes in the global value of ecosystem services”. In: *Global Environmental Change* 26 (2014), pp. 152 –158. ISSN: 0959-3780. DOI: https://doi.org/10.1016/j.gloenvcha.2014.04.002. URL: http://www.sciencedirect.com/science/article/pii/S0959378014000685.

[4] Keren G. Raiter et al. “Under the radar: mitigating enigmatic ecological impacts”. In: *Trends in Ecology & Evolution* 29.11 (2014), pp. 635–644. DOI: 10.1016/j.tree.2014.09.003. URL: https://doi.org/10.1016/j.tree.2014.09.003.

[5] M. C. Hansen et al. “High-Resolution Global Maps of 21st-Century Forest Cover Change”. In: *Science* 342.6160 (2013), pp. 850–853. ISSN: 0036-8075. DOI: 10.1126/science.1244693. eprint: https://science.sciencemag.org/content/342/6160/850.full.pdf. URL: https://science.sciencemag.org/content/342/6160/850.

[6] Mark Everingham et al. “The Pascal Visual Object Classes (VOC) Challenge”. In: *International Journal of Computer Vision* 88.2 (2010), pp. 303–338. DOI: 10.1007/s11263-009-0275-4. URL: https://doi.org/10.1007/s11263-009-0275-4.

[7] J. Deng et al. “ImageNet: A Large-Scale Hierarchical Image Database”. In: *CVPR09*. 2009.

[8] Tsung-Yi Lin et al. “Microsoft COCO: Common Objects in Context”. In: *CoRR* abs/1405.0312 (2014). arXiv: 1405.0312. URL: http://arxiv.org/abs/1405.0312.

[9] Ivan Krasin et al. *OpenImages: A public dataset for large-scale multi-label and multi-class image classification.* Jan. 2016. URL: https://github.com/openimages.

[10] Gencer Sumbul et al. “BigEarthNet: A Large-Scale Benchmark Archive For Remote Sensing Image Understanding”. In: *CoRR* abs/1902.06148 (2019). arXiv: 1902.06148. URL: http://arxiv.org/abs/1902.06148.

[11] Adam Van Etten, Dave Lindenbaum, and Todd M. Bacastow. “SpaceNet: A Remote Sensing Dataset and Challenge Series”. In: *CoRR* abs/1807.01232 (2018). arXiv: 1807.01232. URL: http://arxiv.org/abs/1807.01232.

[12] Darius Lam et al. “xView: Objects in Context in Overhead Imagery”. In: *CoRR* abs/1802.07856 (2018). arXiv: 1802.07856. URL: http://arxiv.org/abs/1802.07856.

[13] Yi Yang and Shawn Newsam. “Bag-of-visual-words and spatial extensions for land-use classification”. In: Jan. 2010, pp. 270–279. DOI: 10.1145/1869790.1869829.

[14] Adam Van Etten. “City-scale Road Extraction from Satellite Imagery”. In: *CoRR* abs/1904.09901 (2019). arXiv: 1904.09901. URL: http://arxiv.org/abs/1904.09901.

[15] *Satellogic Article on Wikipedia*. URL: https://en.wikipedia.org/wiki/Satellogic.

[16] *Youtube Channel Satellogic*. URL: https://www.youtube.com/watch?v=_kE7FC8yWGs.

[17] *Satellogic Website*. URL: https://satellogic.com.

[18] *Github repository of this project*. URL: https://github.com/peterweber85/MasterThesis (visited on 06/26/2019).

[19] Patrick Helber et al. “EuroSAT: A Novel Dataset and Deep Learning Benchmark for Land Use and Land Cover Classification”. In: *CoRR* abs/1709.00029 (2017). arXiv: 1709.00029. URL: http://arxiv.org/abs/1709.00029.

[20] Saikat Basu et al. “DeepSat - A Learning framework for Satellite Imagery”. In: *CoRR* abs/1509.03602 (2015). arXiv: 1509.03602. URL: http://arxiv.org/abs/1509.03602.

[21] Gui-Song Xia et al. “AID: A Benchmark Dataset for Performance Evaluation of Aerial Scene Classification”. In: *CoRR* abs/1608.05167 (2016). arXiv: 1608.05167. URL: http://arxiv.org/abs/1608.05167.

[22] Weixun Zhou et al. “PatternNet: A Benchmark Dataset for Performance Evaluation of Remote Sensing Image Retrieval”. In: *CoRR* abs/1706.03424 (2017). arXiv: 1706.03424. URL: http://arxiv.org/abs/1706.03424.

[23] *Earthexplorer USGS*. URL: https://earthexplorer.usgs.gov/.

[24] *Google Maps static API*. URL: https://developers.google.com/maps/documentation/maps-static/intro.

[25] *Google Maps: Conversion from zoom levels to distance.* https://gis.stackexchange.com/questions/7430/what-ratio-scales-do-google-maps-zoom-levels-correspond-to. accessed 17.06.2019.

[26] *USGS Land Cover Viewer*. URL: https://gis1.usgs.gov/csas/gap/viewer/land_cover/Map.aspx.

[27] *Image folder of published datasets*. URL: https://drive.google.com/open?id=1Hjod1ZTuSIW3VNO2IuGoq_iagI3imnJQ.

[28] Claude E. Duchon. "Lanczos Filtering in One and Two Dimensions". In: *Journal of Applied Meteorology* 18, 1016-1022 (1979). URL: https://icess.eri.ucsb.edu/gem/Duchon_1979_JAM_Lanczos.pdf.

[29] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. "ImageNet Classification with Deep Convolutional Neural Networks". In: *Commun. ACM* 60 (2012), pp. 84–90.

[30] Kaiming He et al. "Deep Residual Learning for Image Recognition". In: *CoRR* abs/1512.03385 (2015). arXiv: 1512.03385. URL: http://arxiv.org/abs/1512.03385.

[31] Jonathan Tompson et al. "Joint Training of a Convolutional Network and a Graphical Model for Human Pose Estimation". In: *CoRR* abs/1406.2984 (2014). arXiv: 1406.2984. URL: http://arxiv.org/abs/1406.2984.

[32] G. Hinton et al. "Deep Neural Networks for Acoustic Modeling in Speech Recognition: The Shared Views of Four Research Groups". In: *IEEE Signal Processing Magazine* 29.6 (2012), pp. 82–97. ISSN: 1053-5888. DOI: 10.1109/MSP.2012.2205597.

[33] Ronan Collobert et al. "Natural Language Processing (almost) from Scratch". In: *CoRR* abs/1103.0398 (2011). arXiv: 1103.0398. URL: http://arxiv.org/abs/1103.0398.

[34] Junshui Ma et al. "Deep Neural Nets as a Method for Quantitative Structure: Activity Relationships". In: *Journal of Chemical Information and Modeling* 55.2 (2015). PMID: 25635324, pp. 263–274. DOI: 10.1021/ci500747n. eprint: https://doi.org/10.1021/ci500747n. URL: https://doi.org/10.1021/ci500747n.

[35] T Ciodaro et al. "Online particle detection with Neural Networks based on topological calorimetry information". In: *Journal of Physics: Conference Series* 368 (2012), p. 012030. DOI: 10.1088/1742-6596/368/1/012030. URL: https://doi.org/10.1088%2F1742-6596%2F368%2F1%2F012030.

[36] G. Cybenko. "Approximation by superpositions of a sigmoidal function". In: *Mathematics of Control, Signals and Systems* 2.4 (1989), pp. 303–314. DOI: 10.1007/BF02551274. URL: https://doi.org/10.1007/BF02551274.

[37] C. Farabet et al. “Learning Hierarchical Features for Scene Labeling”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 35.8 (2013), pp. 1915–1929. ISSN: 0162-8828. DOI: 10.1109/TPAMI.2012.231.

[38] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. “Deep Learning”. In: *Nature* 521 (2015), 436 EP. URL: https://doi.org/10.1038/nature14539.

[39] Leon Bottou and Olivier Bousquet. “The Tradeoffs of Large Scale Learning”. In: (2008), pp. 161–168.

[40] Sebastian Ruder. “An overview of gradient descent optimization algorithms”. In: *CoRR* abs/1609.04747 (2016). arXiv: 1609.04747. URL: http://arxiv.org/abs/1609.04747.

[41] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. “Learning representations by back-propagating errors”. In: *Nature* 323.6088 (1986), pp. 533–536. DOI: 10.1038/323533a0. URL: https://doi.org/10.1038/323533a0.

[42] Yann LeCun et al. “Handwritten Digit Recognition with a Back-Propagation Network”. In: *NIPS*. 1989.

[43] Xavier Glorot, Antoine Bordes, and Y Bengio. “Deep Sparse Rectifier Neural Networks”. In: *Proceedings of the 14th International Conference on Artificial Intelligence and Statisitics (AISTATS) 2011* 15 (Jan. 2011), pp. 315–323.

[44] R. Vaillant, C. Monrocq, and Yann LeCun. “Original approach for the localization of objects in images”. English (US). In: *IEE Conference Publication*. Publ by IEE, 1993, pp. 26–29.

[45] Y. Lecun et al. “Gradient-based learning applied to document recognition”. In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324. ISSN: 0018-9219. DOI: 10.1109/5.726791.

[46] K. Simonyan and A. Zisserman. “Very Deep Convolutional Networks for Large-Scale Image Recognition”. In: *CoRR* abs/1409.1556 (2014).

[47] S. Lawrence et al. “Face recognition: a convolutional neural-network approach”. In: *IEEE Transactions on Neural Networks* 8.1 (1997), pp. 98–113. ISSN: 1045-9227. DOI: 10.1109/72.554195.

[48] *Stanford CS231: Convolutional Neural Networks for Visual Recognition*. URL: http://cs231n.stanford.edu.

[49] Matthew D. Zeiler and Rob Fergus. “Visualizing and Understanding Convolutional Networks”. In: *CoRR* abs/1311.2901 (2013). arXiv: 1311.2901. URL: http://arxiv.org/abs/1311.2901.

[50] Christian Szegedy et al. “Going Deeper with Convolutions”. In: *CoRR* abs/1409.4842 (2014). arXiv: 1409.4842. URL: http://arxiv.org/abs/1409.4842.

[51] Kaiming He et al. "Deep Residual Learning for Image Recognition". In: *CoRR* abs/1512.03385 (2015). arXiv: 1512.03385. URL: http://arxiv.org/abs/1512.03385.

[52] Saining Xie et al. "Aggregated Residual Transformations for Deep Neural Networks". In: *CoRR* abs/1611.05431 (2016). arXiv: 1611.05431. URL: http://arxiv.org/abs/1611.05431.

[53] Christian Szegedy, Sergey Ioffe, and Vincent Vanhoucke. "Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning". In: *CoRR* abs/1602.07261 (2016). arXiv: 1602.07261. URL: http://arxiv.org/abs/1602.07261.

[54] Gustav Larsson, Michael Maire, and Gregory Shakhnarovich. "FractalNet: Ultra-Deep Neural Networks without Residuals". In: *CoRR* abs/1605.07648 (2016). arXiv: 1605.07648. URL: http://arxiv.org/abs/1605.07648.

[55] D. G. Lowe. "Object recognition from local scale-invariant features". In: *Proceedings of the Seventh IEEE International Conference on Computer Vision*. Vol. 2. 1999, 1150–1157 vol.2. DOI: 10.1109/ICCV.1999.790410.

[56] David G. Lowe. "Distinctive Image Features from Scale-Invariant Keypoints". In: *International Journal of Computer Vision* 60.2 (2004), pp. 91–110. ISSN: 1573-1405. DOI: 10.1023/B:VISI.0000029664.99615.94. URL: https://doi.org/10.1023/B:VISI.0000029664.99615.94.

[57] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. "SURF: Speeded Up Robust Features". In: *Computer Vision – ECCV 2006*. Ed. by Aleš Leonardis, Horst Bischof, and Axel Pinz. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 404–417. ISBN: 978-3-540-33833-8.

[58] N. Dalal and B. Triggs. "Histograms of oriented gradients for human detection". In: *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*. Vol. 1. 2005, 886–893 vol. 1. DOI: 10.1109/CVPR.2005.177.

[59] Kevin McGuinness. *Transfer Learning (D2L4 Insight at DCU Machine Learning Workshop 2017)*. 2017. URL: https://www.slideshare.net/xavigiro/transfer-learning-d2l4-insightdcu-machine-learning-workshop-2017.

[60] *ResNet - Applications - Keras Documentation*. URL: https://keras.io/applications/{\#}resnet (visited on 06/26/2019).

[61] *AWS Elastic Cloud Compute Pricing*. URL: https://aws.amazon.com/de/ec2/pricing/on-demand/.