UNIVERSITAT DE BARCELONA

FUNDAMENTALS OF DATA SCIENCE MASTER'S THESIS

Man-made Structures Detection from Space

Author:
Peter WEBER

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 26, 2019

UNIVERSITAT DE BARCELONA

Abstract

Facultat de Matemàtiques i Informàtica

Man-made Structures Detection from Space

by Peter WEBER

The Thesis Abstract is written here (and usually kept to just this page). The page is kept centered vertically so can expand into the blank space above the title too...

Acknowledgements

The acknowledgments and the people to thank go here, don't forget to include your project advisor...

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 labelled 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 space, whose weight is 35kg, a minuscul fraction of conventional satellite systems (more than 1000kg). 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 [16]. 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 evalute 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 and ecological cost of acquiring and processing the information? 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.

The detailed outline of the thesis is the following:

- 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 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 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 [17]	Sentinel-2	27,000	10	64	10
UCMerced [13]	USGS	2100	0.3	256	21
DeepSat [18]	USGS	405,000	1	28	6
AID [19]	Google Earth	10,000	0.5 - 8	600	30
PatternNet [20]	Google Earth	30,400	0.06 - 4.69	256	38

TABLE 2.1: **Publicly available remote sensing datasets with labels.**

The USGS National Map Urban Area Imagery collection [21] 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 pixel. 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 and therefore they are very unbalanced, and do not contain sufficient variety of images for the classes without human influence. Therefore, we decided to collect and label images by ourselfes. 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 [22]. 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 [23], 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 [21]. 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 [24], precisely to guarantee larger variety through the selection of data from distinct land use categories.

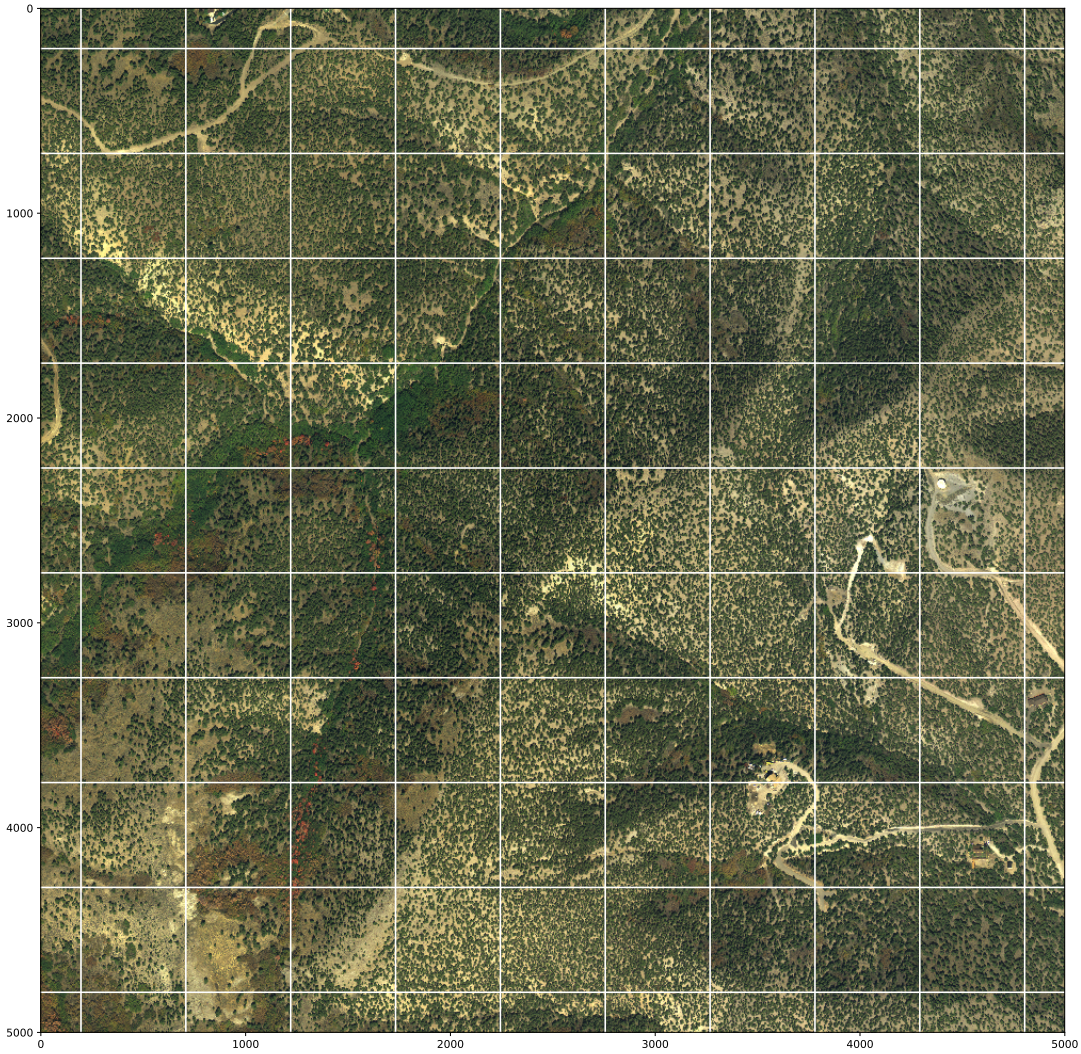

FIGURE 2.1: **Example of unprocessed image.** This image has a size of 5000 $\times$ 5000 pixel. The continuous white lines are guidelines for the eye to demonstrate how we crop smaller images where we only return a cropped image from the white squares with size 512 $\times$ 512.

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, 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, we located it on USGS Earthexplorer and downloaded images from that area. In particular, we constructed two

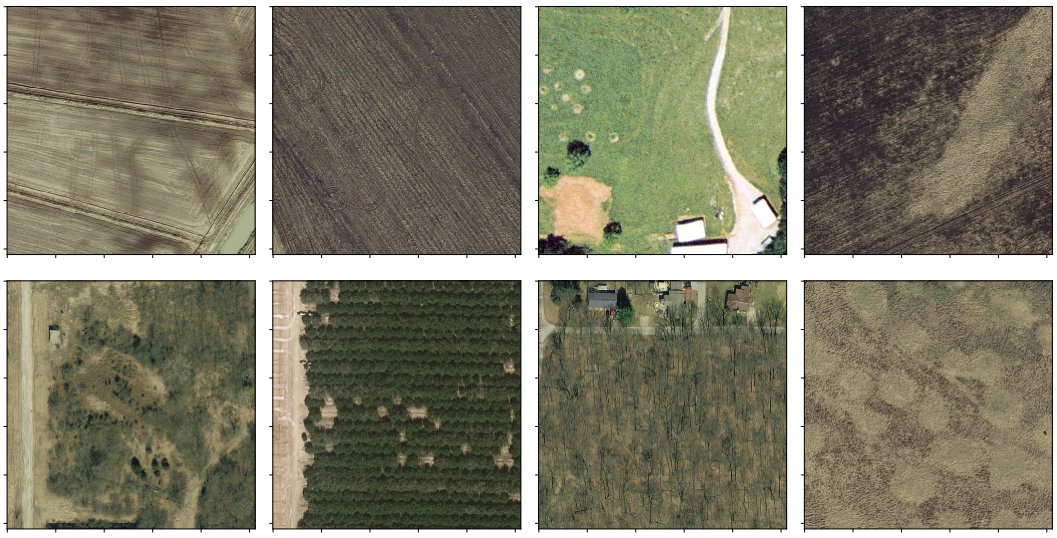

FIGURE 2.2: **Example images of category Agricultue.** All images in this figure show clear signs of human impact. The images have a size of 512 × 512 pixels and a resolution of 0.3m per pixel.

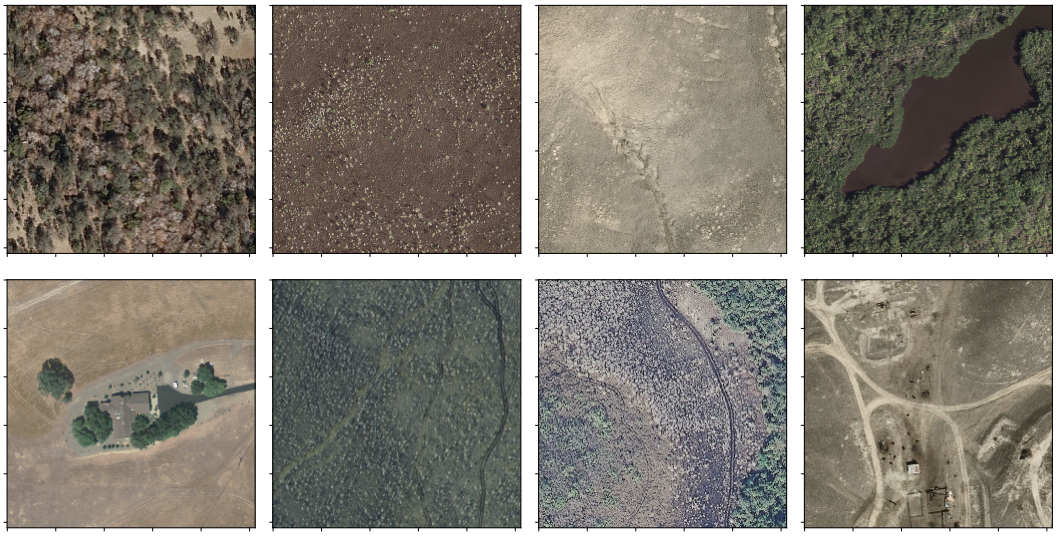

FIGURE 2.3: **Example images of category shrubland-grassland.** The images in the first row do not contain any human influence, while the images in the second row show man-made structures. The images in this figure have a size of 512 × 512 pixels and a resolution of 0.3m per pixel.

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 Program (NAIP). Note that the images in these categories usually have a height and a width of several thousand pixels, and hence oocupy 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.

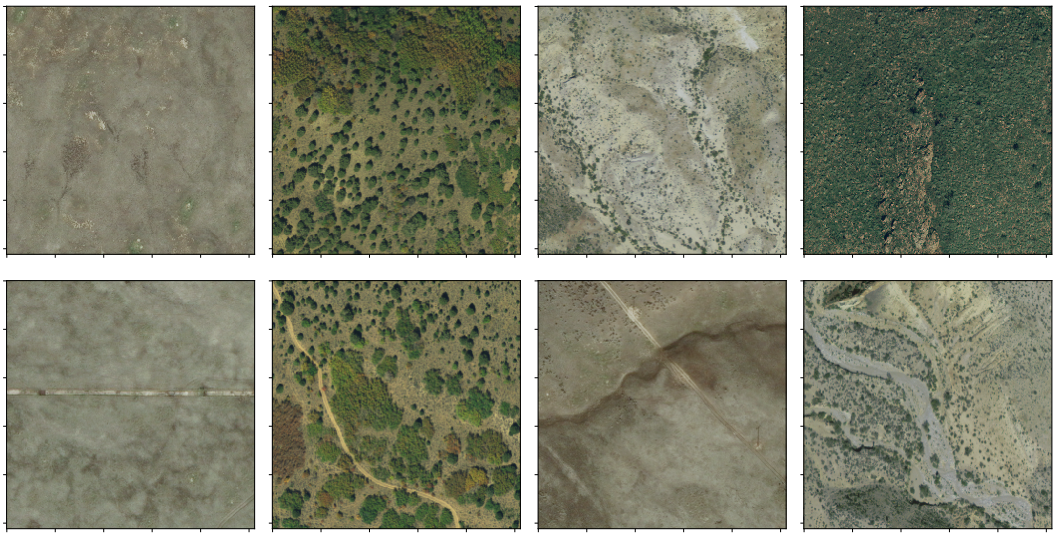

FIGURE 2.5: **Example images of category semi-desert.** The images in the first row do not contain any human influence, while the images in the second row show man-made structures. The images in this figure have a size of 512 × 512 pixels and a resolution of 0.3m per pixel.

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) from the large raw image (5000×5000). 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 [25]. 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 very 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, and were therefore all classified 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.

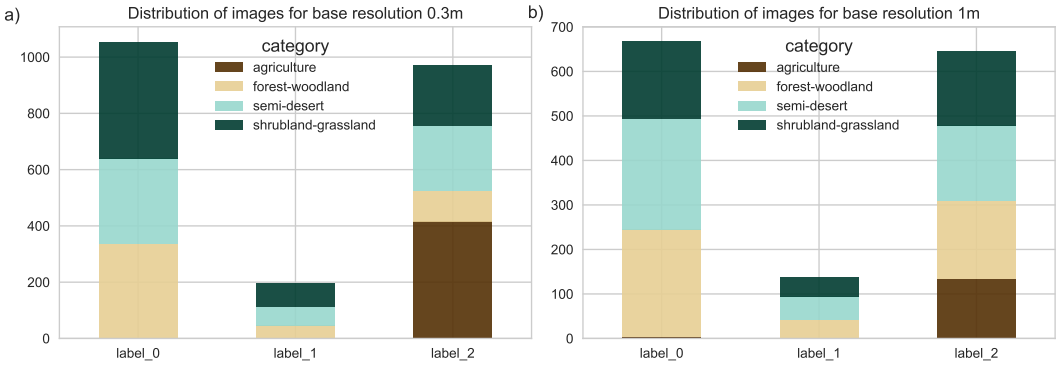

FIGURE 2.6: **Number of images per category and label.** (a) Distribution of images for dataset with resolution of 0.3m per pixel. (b) Distribution of images for dataset with resolution of 1m per pixel.

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 4.

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 [26] 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).

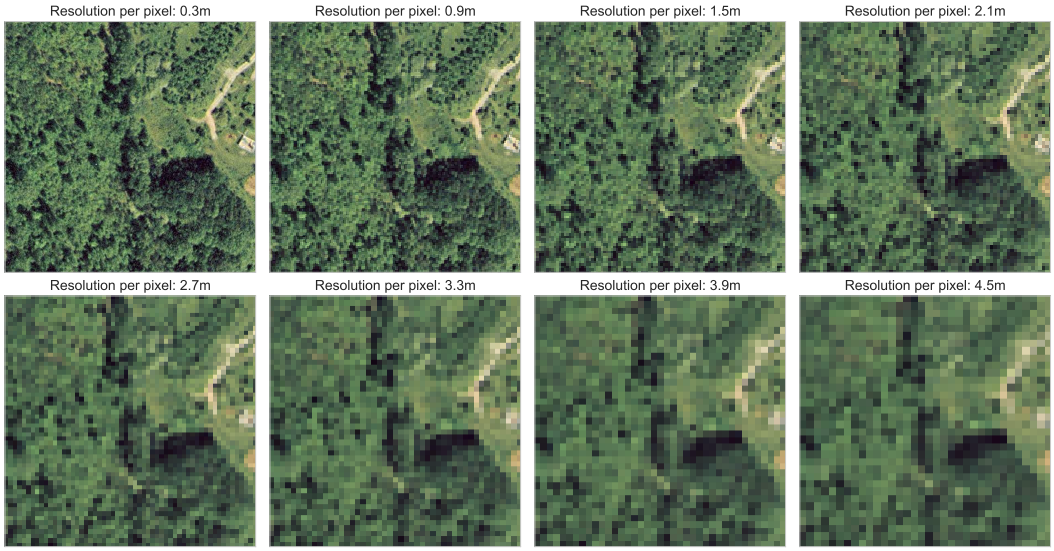

FIGURE 2.7: **Example of image downsampling**. The upper left image has a base resolution of 0.3m per pixel and a size of 512×512 pixels whereas the lower right image has the worst resolution, 4.5m per pixel, and a size of 34×34 pixels. All intermediate images are downsampled by a factor corresponding 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 decade. 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 [27], and image [28], video [29] and audio [30] processing while RNNs brought about advances in research and applications on sequential data, i.e. in speech and text [31]. 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 [32] and the analysis of data from elementary particle physics [33].

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, sigmoid ($\sigma(x) = 1/(1+exp(-x))$) and tanh ($tanh(x) = 2\sigma(2x) - 1$) 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 [34], 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 [35].

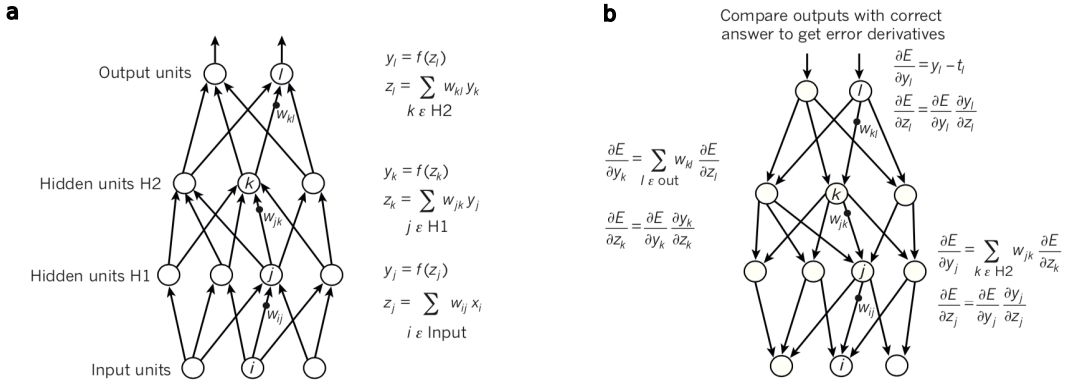

FIGURE 3.1: **Example of 3-Layer Neural Network. (a)** Feed-forward representation of a neural network with two hidden units H1 and H2 and a binary output unit. The inputs to every layer are weighted averages, specified by the weights w, of the outputs of the previous layer. In every layer, the outputs 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 [36].

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. The minimization of the objective function is accomplished by applying gradient descent based methods, in practice, often stochastic gradient descent [37] or variants of it [38]. The computation of the gradient of the objective function with respect to all weights of the network is accomplished using backpropagation [39] (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. MENTION LOSS FUNCTION, FOR INSTANCE SOFTMAX

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 [40]. 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 with the emergence of "Big Data" and new powerful graphics processing units (GPUs) about a decade ago. Also theoretical advances helped to eliviate the difficulties to train deep learning models. These include the application of the ReLU non-linearity [41], 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 [42, 43] 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 [44] 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 [45]. 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 weights, or 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 multiple filters the number of parameters is multiplied by K, which corresponds to having K activation maps. Typically many filters are applied at every convolutional layer, so that every filter learns a set of weights that relate to one particular feature in the image (see Fig. 3.2(a)).

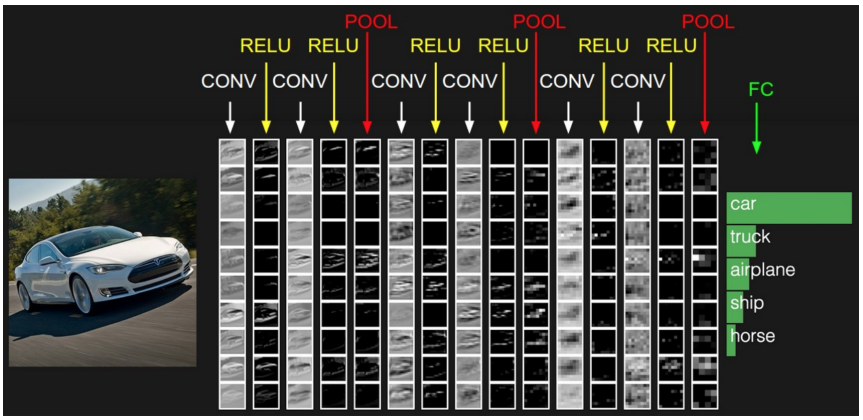

FIGURE 3.2: **Activations of a convolutional neural network used for classification.** The forward pass in this network is computed from left to right. Every column represents a layer of the network and the small images are the filter activations when passing the passing the image through the network. 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 [**cs231_convnets**].

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.

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). Max-pooling with stride 2 would, for instance, transform a 4×4 input image to a 2×2 output image by taking the maximum element at every 2-by-2 location. 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

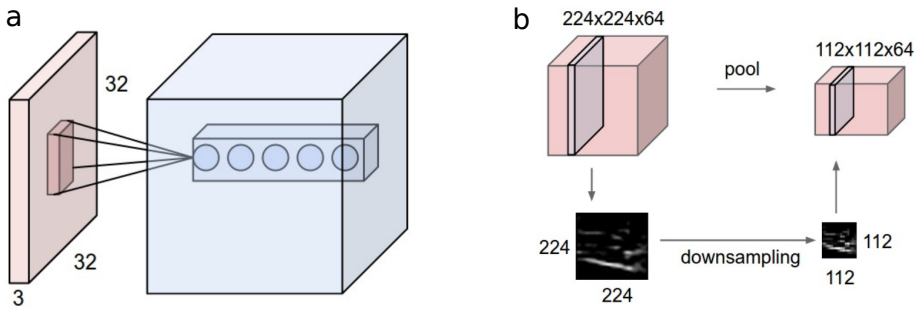

FIGURE 3.3: **Example of a convolutional layer (a) and a pooling layer (b). (a)** In this example a five small filter (represented by the 5 dots in the output volume) are applied to the input image of dimensions $32 \times 32 \times 3$. The filters extend over the entire input depth, but look only at 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 a invariance to local variations. Figure is adapted from [46].

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. It halfed the error in comparison to all competing non deep learning based approaches (see Fig. 3.4). 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.

The winner of the ILSVRC2013 was the ZFNet [47], 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 [44] had 19 layers and the GoogleNet had 22 layers [48]. 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, which is used 9 times.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 the AlexNet.

The researchers that developed ResNet [49], which was the ImageNet classification winner in 2015, wanted to answer the question: What happens when we continue stacking deeper layers on a plain convolutional network? By comparing

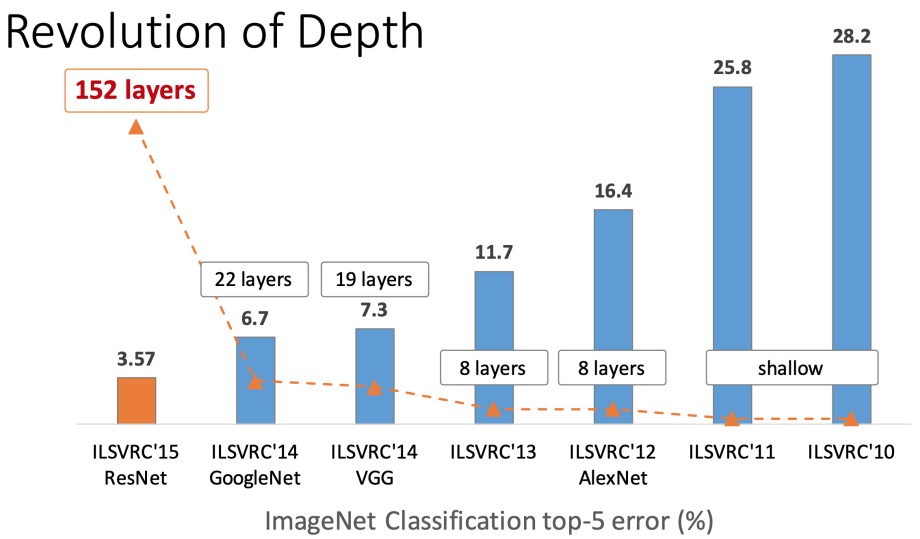

FIGURE 3.4: Top-5 test error of the winning solutions of the Large Scale Visual Recognition Challenge through years 2010 to 2015. Figure is adapted from this link.

a 20 layer network with a 56 layer network they found, that the deeper network had lower performance both on the training set and on the test set. This implies that there is a fundamental problem, since overfitting can be excluded. The authors 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. REWRITE THIS PARAGRAPH TO BE MORE COMPACT, AND MORE PRECISE.

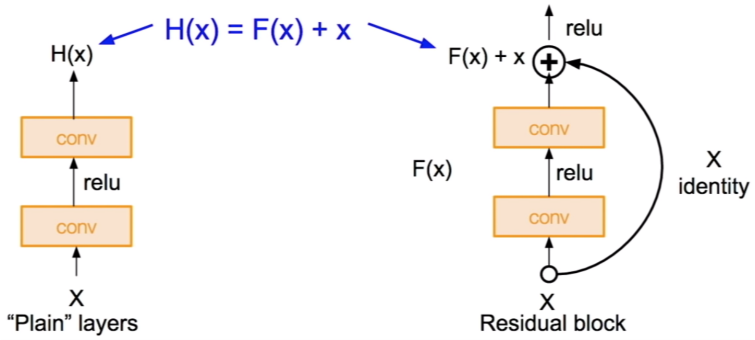

FIGURE 3.5: **Residual block of ResNet architecture.** Comparison between plain convolutional architecture (left) and convolutional architecture using a residual block. For the plain structure the network learns the mapping $H(x)$ while for the residual structure the network learns a residual $F(x) = H(x) - x$. x here is the identity mapping. Figure is adapted from [46].

So instead of simply stacking more and more layers together He et al. developed residual blocks containing an identity mapping additionally to the convolutional layer. When having an identity mapping between input and output the network just needs to learn the residual connections represented 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 ImageNet dataset in the year 2015.

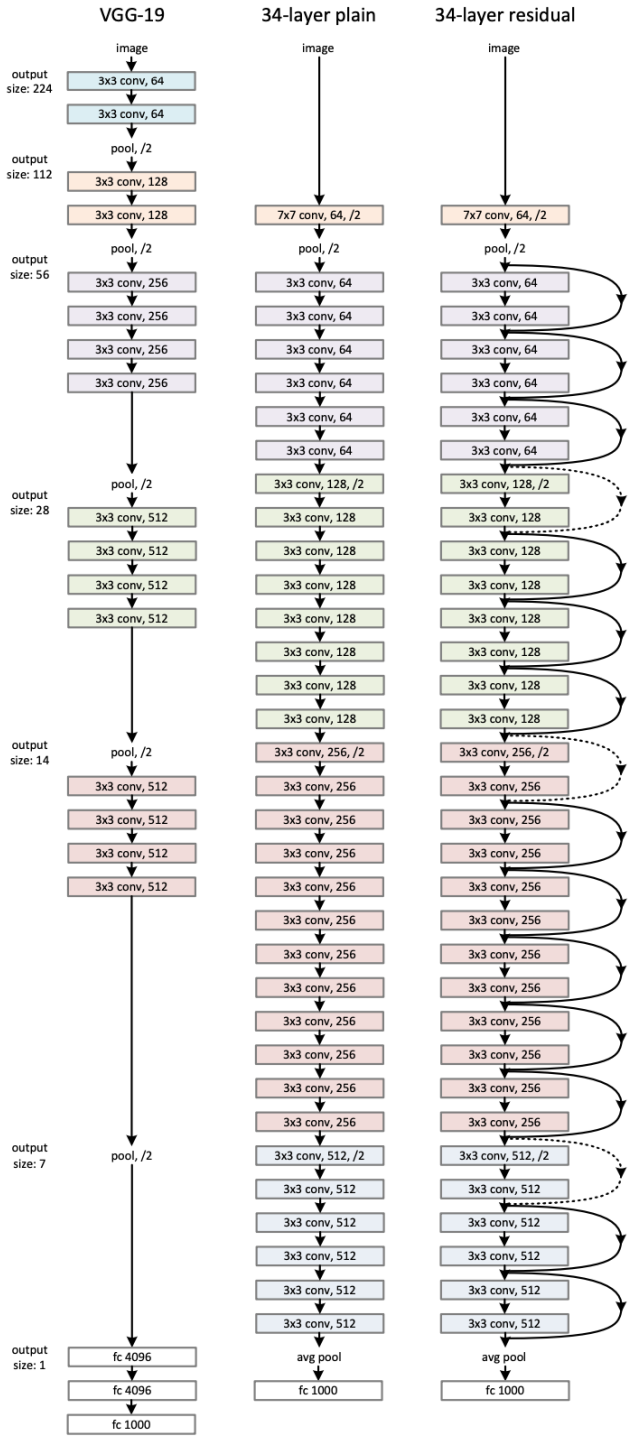

FIGURE 3.6: **Comparison between deep convolutional neural network architectures.** On the left we have a VGG network with 19 layers, in the center a plain ResNet with 34 layers, and on the right a ResNet with 34 layers and residual connections. Figure is adapted from[49].

The architecture of a ResNet with 34 layers is shown in Fig. 3.6. The first convolutional layer has a filter size of 7×7 while all consecutive filters are chosen to be as

small as possible (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. 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.

After the ResNet several other networks have been developed, but they only performed incrementally better and most of them were more sophisticated versions of the ResNet or combinations of the ResNet with other architectures, such as Inception-V4 [50]. We therefore decided to use a ResNet architecture that was pretrained on ImageNet for the experiment in this thesis. MENTION: FRACTAL NET AND LAYER DROPOUT, DENSENT.

Chapter 4

Deep Learning 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 [51, 52], SURF [53], Histograms of Oriented Gradients (HOG) [54] and Gabor filters.

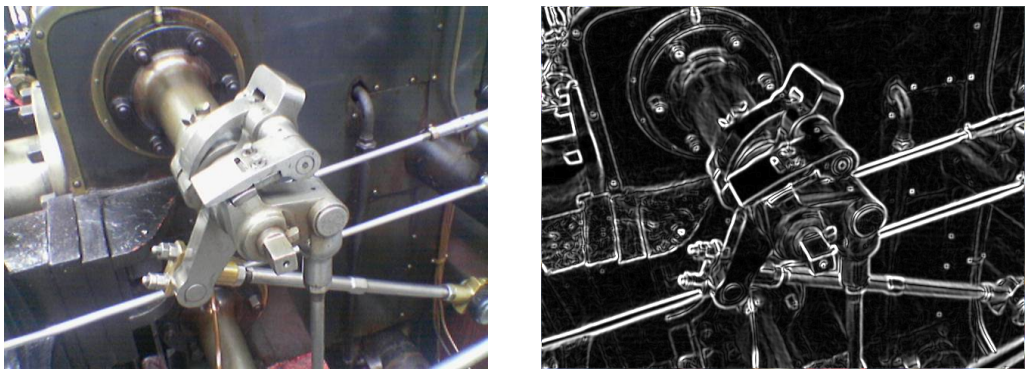

FIGURE 4.1: **Example of the Sobel filter**

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.

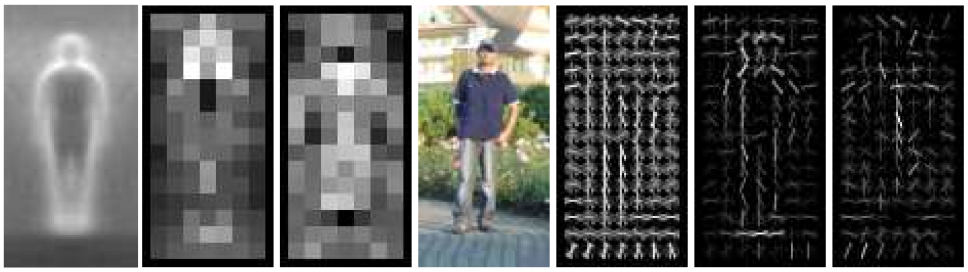

FIGURE 4.2: **Examples of HOG detector [54]**

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. 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.

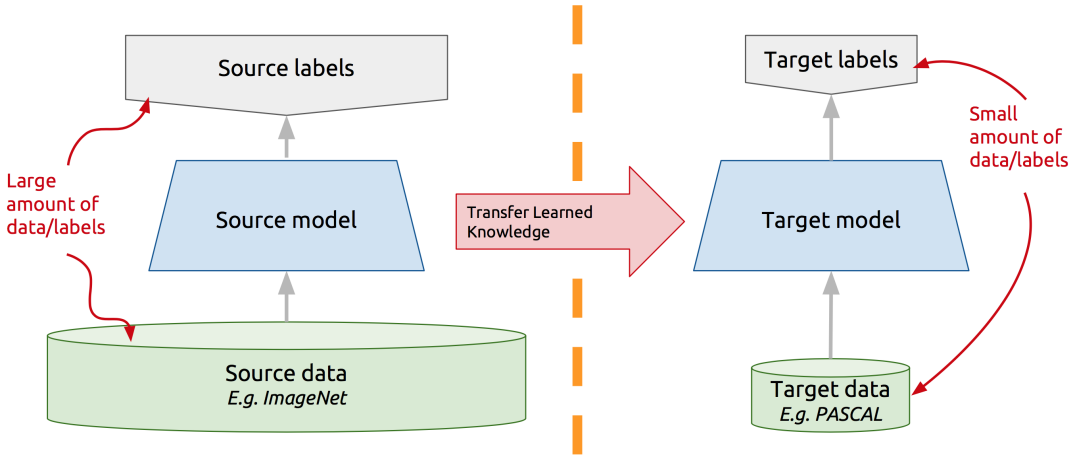

FIGURE 4.3: **Transfer Learning**: a model learned from a large dataset can be transferred and reused for another purpose. [**McGuinness2017**]

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 the 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.

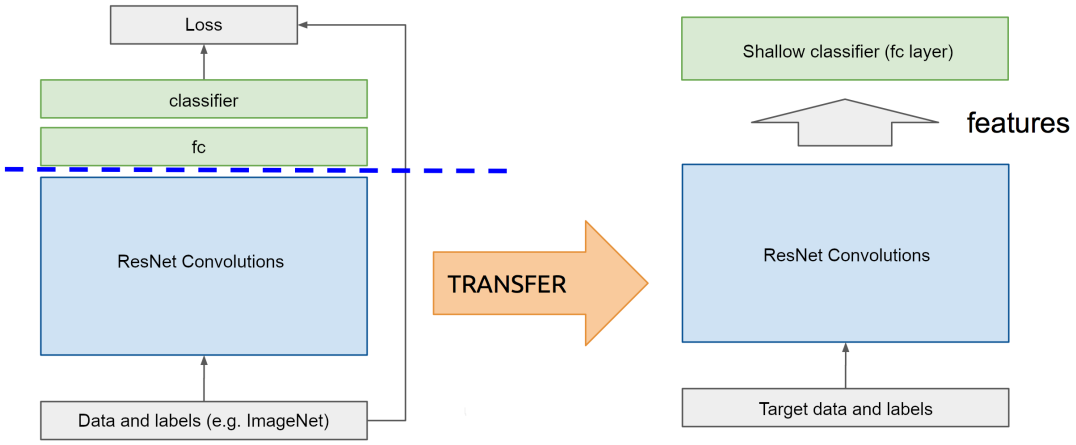

FIGURE 4.4: **Transfer Learning from a ResNet** (figure adapted from **[McGuinness2017]**)

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 and 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, 4.6, 4.7 and 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 of 49^{th} layer.

4.2.2 Complete architecture

As mentioned before, 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 512 neurons (CHECK) with ReLU activation, followed by a single Dense node with a Sigmoid activation acting as the classifier. This model is then trained on the dataset with and RMSprop

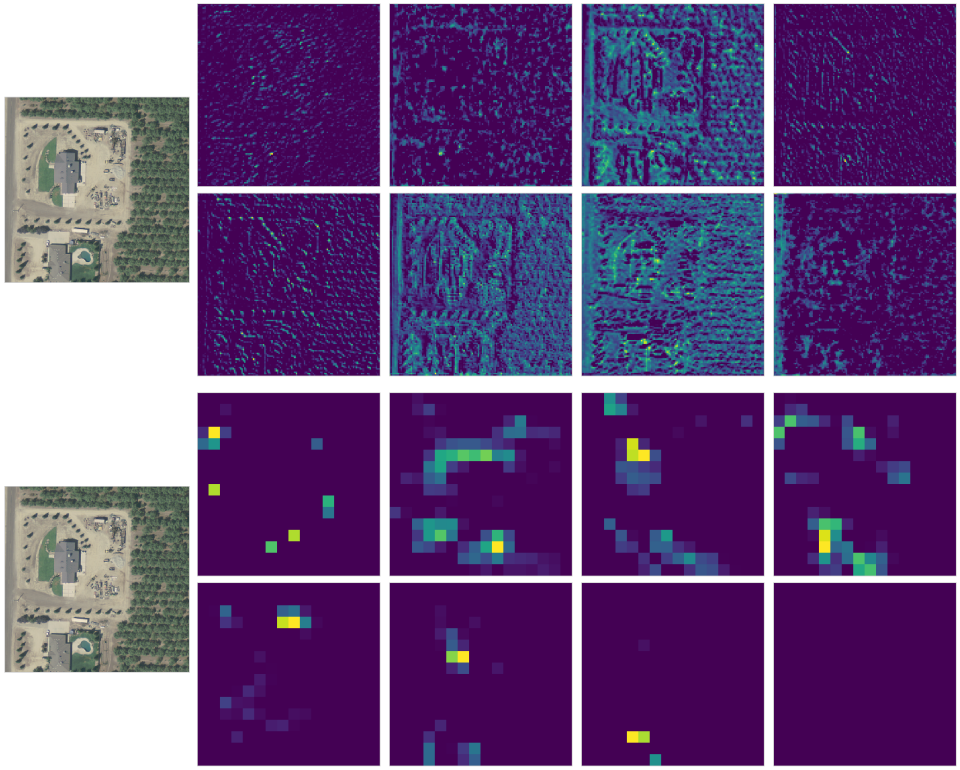

FIGURE 4.5: **ResNet activations of an Agriculture image:** 10^{th} **layer (top) and final layer,** 49^{th} **(bottom).**

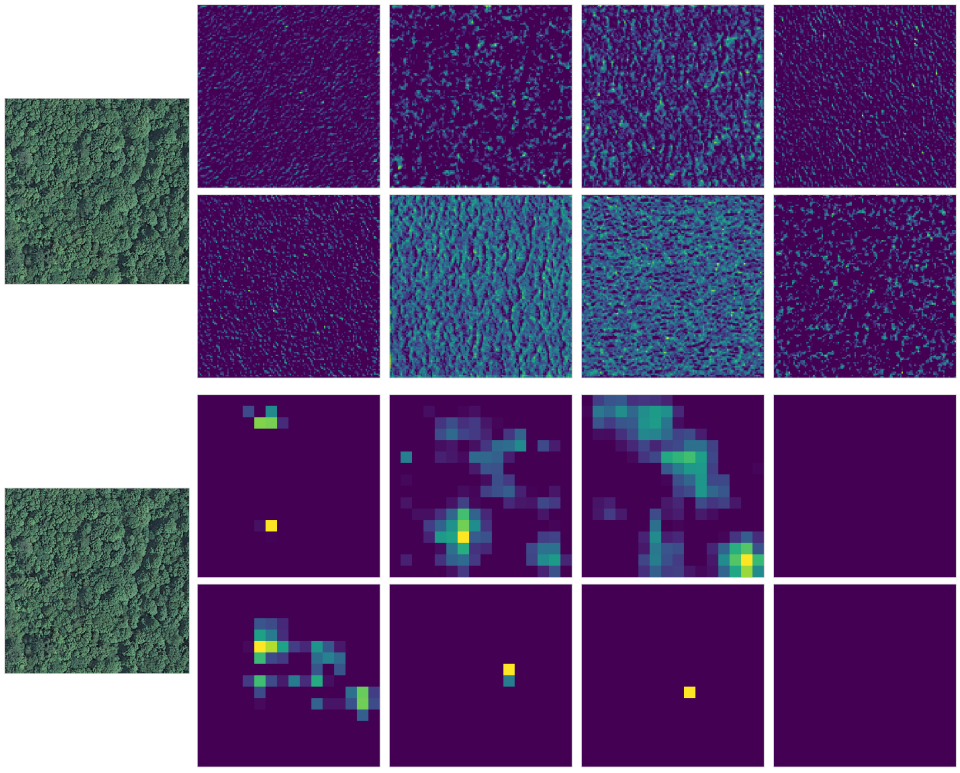

FIGURE 4.6: **ResNet activations of a Forest-woodland image:** 10^{th} **layer (top) and final layer,** 49^{th} **(bottom).**

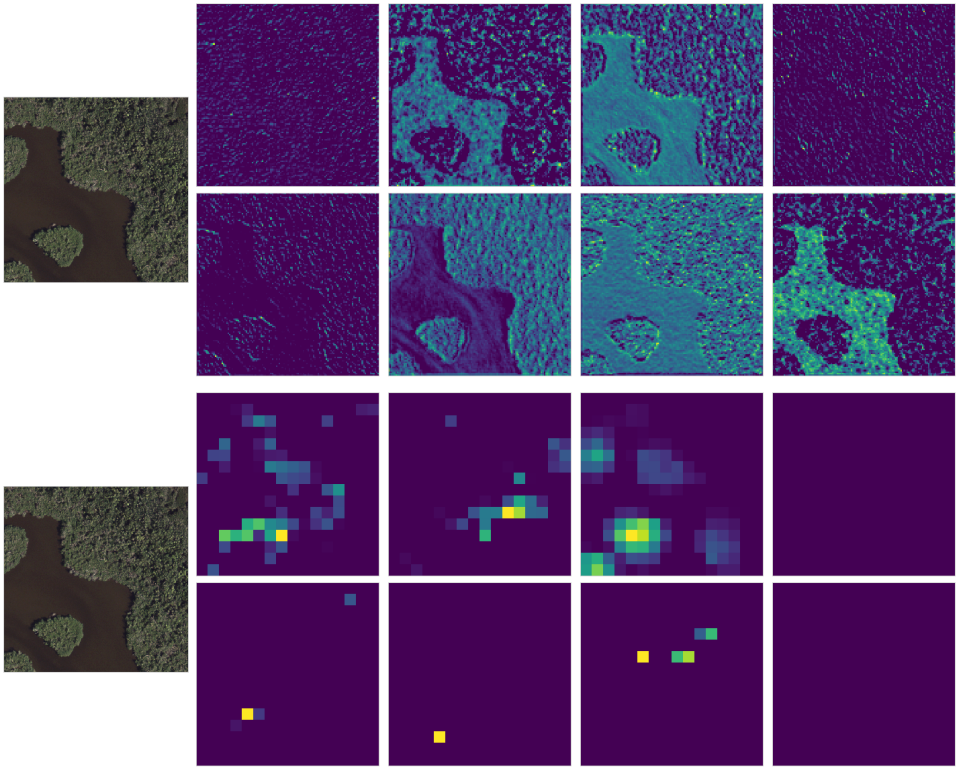

FIGURE 4.7: **ResNet activations of a Shrubland-grassland image: 10^{th} layer (top) and final layer, 49^{th} (bottom).**

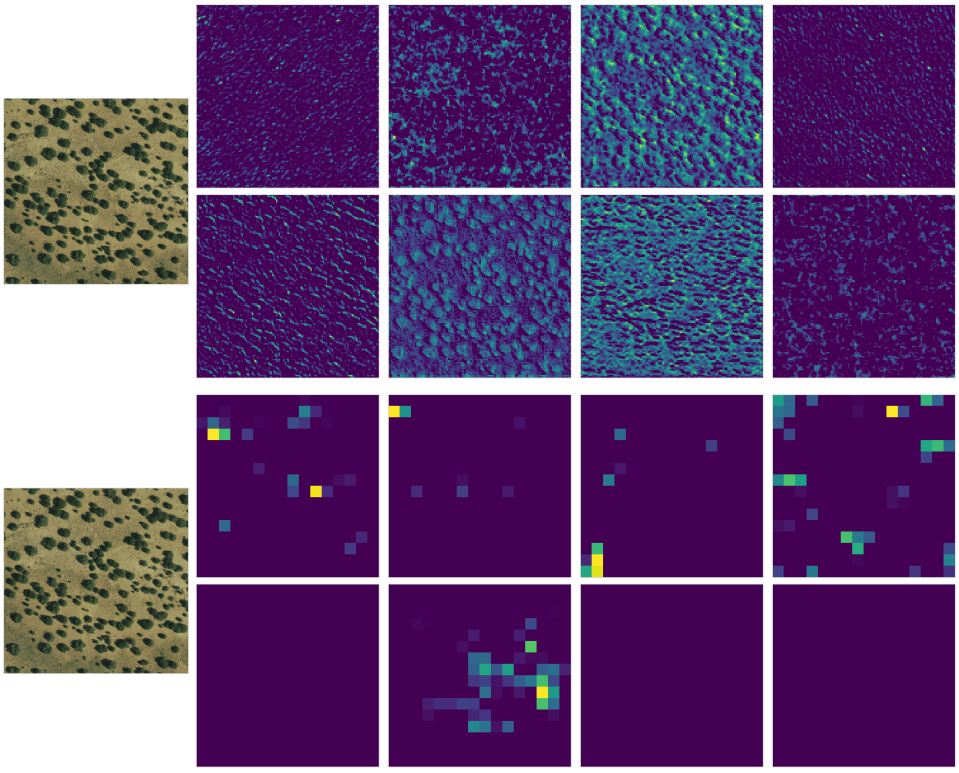

FIGURE 4.8: **ResNet activations of a Semi-desert image: 10^{th} layer (top) and final layer, 49^{th} (bottom).**

optimizer and a binary cross-entropy loss function. This same architecture is then 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, and in the following section we describe the complete 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 and experiments

In order to perform the experiments, we consider the following pipeline for each of the datasets ($0.3m$ and $1m$ resolution):

1. Load the original images (at the original resolution) from disk.
2. Downsample the images to the desired resolution.
3. Compute the ResNet activations (at the 49^{th} activation layer) of the resulting images (and save to disk for later use).
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.
5. Train the model separately for each combination of 7 train sets, with the remaining one as validation. Then results of the 8 experiments are averaged for more consistency.
6. Repeat for all downgraded resolutions needed.

Further experiments in order to fine-tune the model complexity and the splitting parameters could be done, but it is not the goal of this project.

- discussion about resolutions considered

Transfer learning:

- https://www.youtube.com/playlist?list=PLC1qU-LWwrF64f4QKQT-Vg5Wr4qEE1Zxk

- http://cs231n.github.io/transfer-learning/
- VGG: https://arxiv.org/abs/1409.1556
- https://iopscience.iop.org/article/10.1088/1742-6596/1087/6/062032/pdf
- https://arxiv.org/abs/1805.02294
- https://cs.nyu.edu/~fergus/papers/zeilerECCV2014.pdf
- https://towardsdatascience.com/transfer-learning-from-pre-trained-models-f2393f124751
- https://www.mitpressjournals.org/doi/pdf/10.1162/neco_a_00990
- https://towardsdatascience.com/a-comprehensive-hands-on-guide-to-transfer-learning-wi

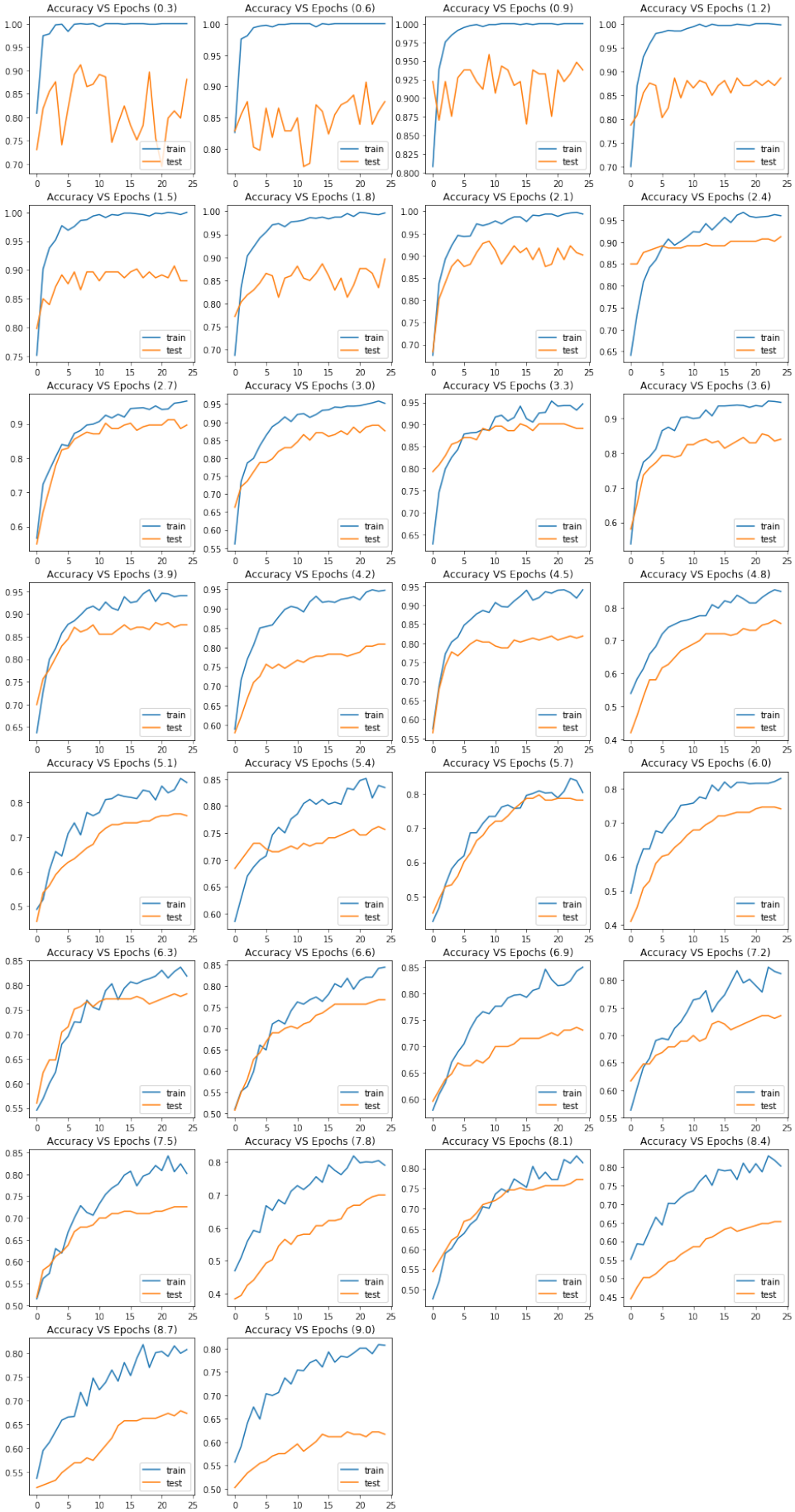

FIGURE 4.10: **Convergence plots for all downgraded resolutions of the** $0.3m$ **dataset.**

Chapter 5

Results

In this chapter we discuss the results obtained with our model. First, we analyze how the model works in a given (original) resolution, 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 en environment impact around Satellite imagery to analyze the world.

5.1 Transfer Learning on Aerial Imagery

5.2 Man-made Structures Detection at Different Scale

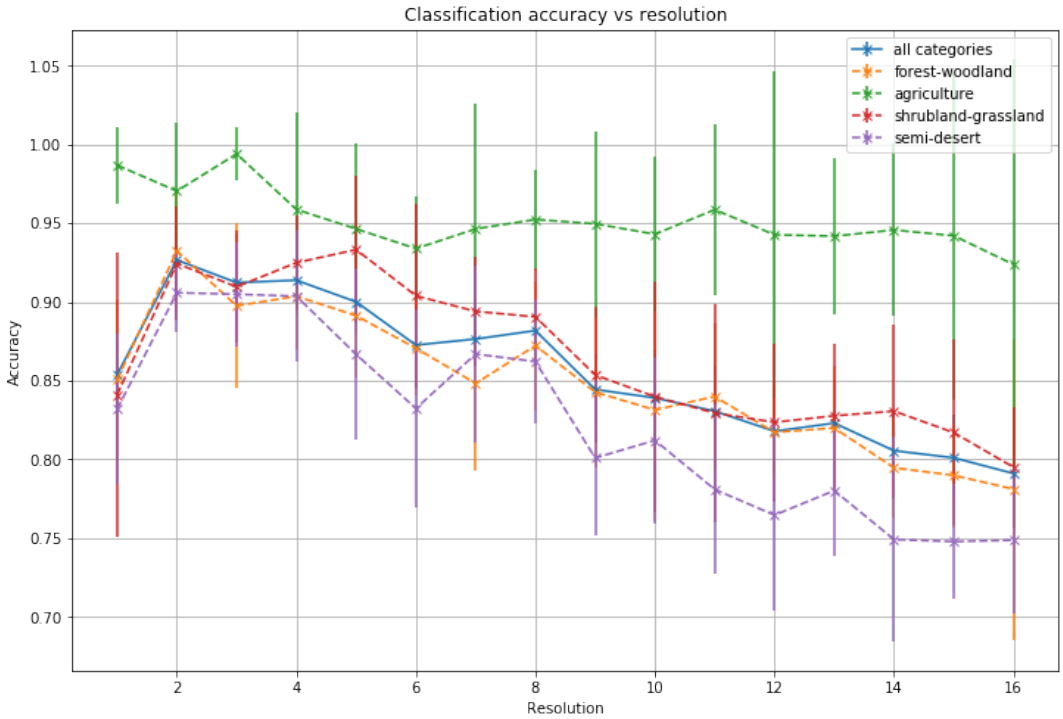

FIGURE 5.1: 1m **dataset**

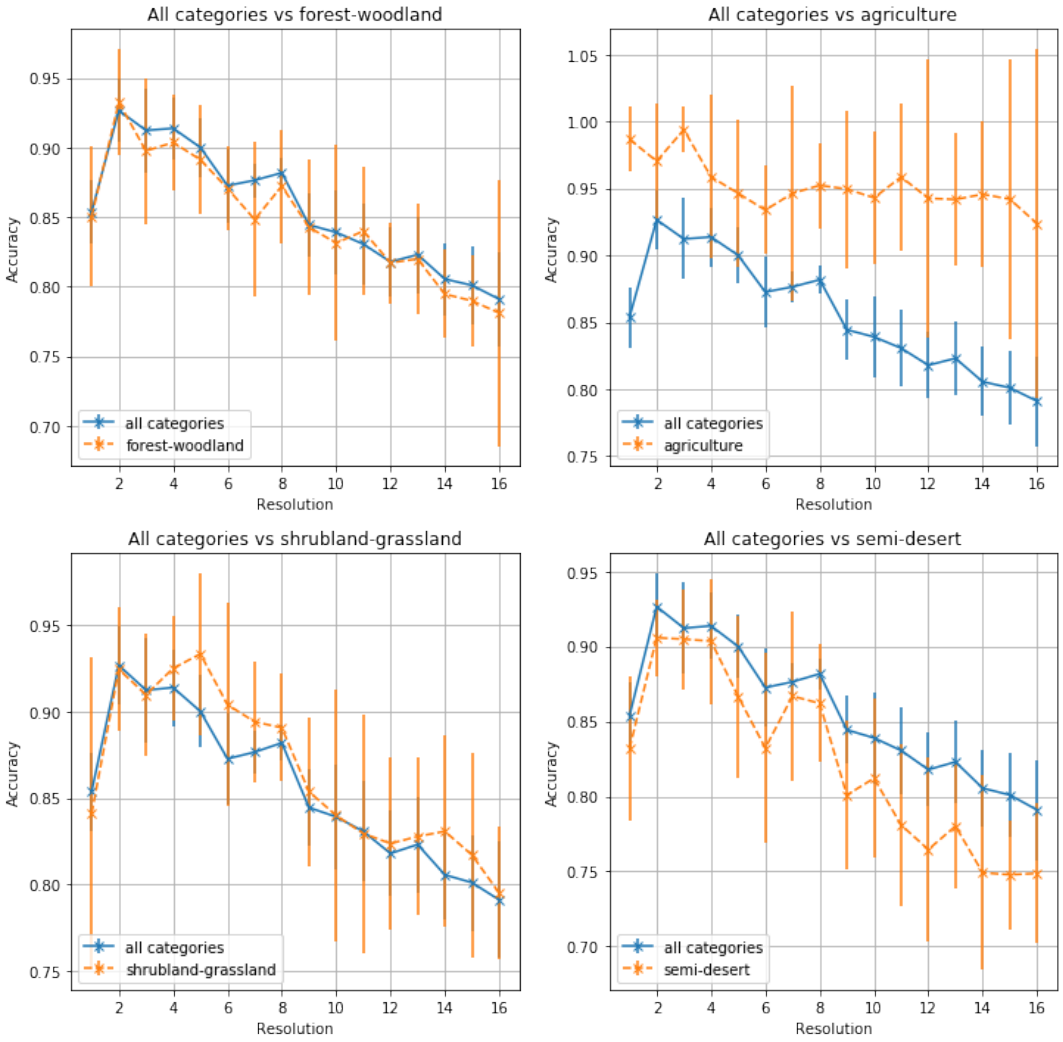

FIGURE 5.2: $1m$ **dataset**

5.3 Cost and Environmental Impact

It is a very costly business to capture earth imagery data with satellites. Here we discuss both financial cost as well as environmental cost due to carbon footprint required to build and launch a satellite, and to power computational ressources for data processing. We further study these costs as a function of resolution. However, our estimates are very rough approximations because many are factors 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 [55] 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 [56]. We hence can calculate the satellite cost per km^2. With $area/lifespan = 10^6 \times 365/4.2\ km^2$ 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	0.04198	4.198×10^{-8}
serving to final client	0.09	$/Gb	0.00047232	4.7232×10^{-10}
prediction (AWS)	0.05 & ∼6	$/h & s/km^2	0.00145	1.45×10^{-9}

TABLE 5.1: **Costs for image data processing.**

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 (provided by Satellogic). Note that we used the conversion factor 0.00524 for an uncompressed image to convert from Gigabytes to km^2 and we assume 12 months of data storage. The prediction step represents the prediction of human impact. It is estimated by loading 4 images that each have an area of about $500 \times 500 m^2$, calculating the ResNet activations of the final layer, and predicting the class using the models trained in chapter **??** 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 [57].

To finally obtain resolution dependence of 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 costs per pixel ($\times 10^{-6}$). We then multiply with the number of pixels we need to capture 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. We finally obtain the plot shown in Fig. 5.3. 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 already two orders of magnitued 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.

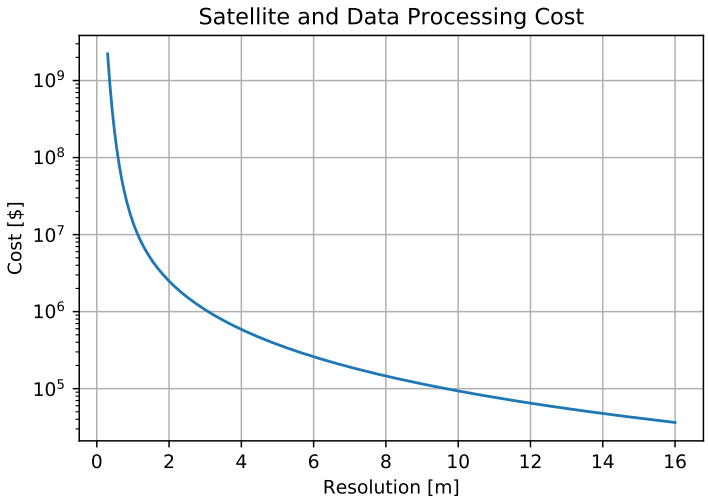

FIGURE 5.3: **Satellite and data processing cost.** The total financial cost to capture images with a satellite and process the data as function of resolution.

OPTIONAL In the remainder of this section we provide a brief discussion about the environmental cost associated to earth observation with satellite's. First, bringing the satellite into orbit is estimated as follows. The satellite needs to reach a certain velocity

$$v = \sqrt{\frac{Gm_e}{R}} \approx 8000 m/s \tag{5.1}$$

parallel to the surface of the earth in order to orbit earth, which can be calculated from the equilibrium between gravitational force $F_g = Gm_e m_s/R^2$ and centrifugal force $F_z = m_s v^2/R$. Here $G = 6.67 \times 10^{-11} m^3/(s^2 kg)$ is the gravitational constant, $m_e = 5.9 \times 10^{24} kg$ the mass of the earth, m_s the mass of the satellite, $R = 6371 km$ the radius of the earth. Note that we assume that the distance of the satellite to the surface of the earth is negligible compared to the radius of the earth. The approximate amount of fuel needed to accelerate a satellite to these levels can be calculated using the Tsiolkovsky Rocket Equation [58]. We obtain about 12kg of fuel per kg of final satellite weight. With a satellite weight of about 35kg and a factor of 3 to convert Kerosene consumption into CO2 emission [59], we obtain about 1200kg of CO2. Discuss Carbon footprint of datacenters...

Chapter 6

Conclusions

6.1 Conclusions

Lorem ipsum

6.1.1 Subsection 1

Nunc posuere

6.2 Further work

- Dataset: improve, enlarge, better classify, variety
- Model: CNN feature extraction, other architectures and number of activations, etc

Chapter 7

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

[17] 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.

[18] 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.

[19] 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.

[20] 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.

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

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

[23] *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.

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

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

[26] 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.

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

[28] 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.

[29] 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.

[30] 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.

[31] 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.

[32] 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.

[33] 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.

[34] 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.

[35] 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.

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

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

[38] Diederik P. Kingma and Jimmy Ba. "Adam: A Method for Stochastic Optimization". In: *CoRR* abs/1412.6980 (2015).

[39] 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.

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

[41] 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.

[42] 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.

[43] 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.

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

[45] 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.

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

[47] 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.

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

[49] 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.

[50] 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.

[51] 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.

[52] 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.

[53] 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.

[54] 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.

[55] *Planet*. URL: https://www.planet.com.

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

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

[58] *Tsiolkovsky Rocket Equation*. URL: https://en.wikipedia.org/wiki/Tsiolkovsky_rocket_equation.

[59] *Conversion from kerosence to CO2*. URL: https://www.engineeringtoolbox.com/co2-emission-fuels-d_1085.html.