UNIVERSITAT DE BARCELONA

FUNDAMENTALS OF DATA SCIENCE MASTER'S THESIS

# Man-made Structures Detection from Space

*Author:*
Eduard RIBAS FERNÁNDEZ

*Supervisor:*
Dr. Jordi VITRIÀ

*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 22, 2019

UNIVERSITAT DE BARCELONA

# *Abstract*

Facultat de Matemàtiques i Informàtica

**Man-made Structures Detection from Space**

by Eduard RIBAS FERNÁNDEZ

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

BigEarthNet Motivation: Existing Remote Sensing Datasets contain a small number of annotated images and therefore do not suffice to train a complex ConvNet with many different parameters. Remote sensing versus Computer (Using pretrained nets on ImageNet)

BagOfVisualWords: useless

Extreme value theory-based calibration: useless

Climate Change, population control, ilegal habitat/ilegal land use

The goal is not really to build the top performance, state-of-the-art model to detect all sort of human impact in satellite images, but rather to analyze the feasibility and cost of doing so at differen resolutions. Of course, better algorithms could be implemented to accurately identify certain types of human impact, but we consider a more general problem.

## 1.2 Satellogic

## 1.3 Thesis Outline

# Chapter 2

# Building Datasets

In this chapter, we will give an overview of existing (labelled) 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 existing labelled 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 labels, and variations of the images as large as possible with respect to different terrains. 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 $500 \times 500$ 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 for the images. Further it contains a description about the number of images, the resolution of the images, the size (in pixel) of the images where images are squared, 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 [28] | Sentinel-2 | 590,326 | 10, 20, 60 | 120, 60, 20 | $\sim 50$ |
| EuroSAT [14] | Sentinel-2 | 27,000 | 10 | 64 | 10 |
| UCMerced [34] | USGS | 2100 | 0.3 | 256 | 21 |
| DeepSat [1] | USGS | 405,000 | 1 | 28 | 6 |
| AID [33] | Google Earth | 10,000 | 0.5 - 8 | 600 | 30 |
| PatternNet [36] | 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 (*see link*) 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 has 21 categories of which only 2 belong to images without human impact, while the other 19 show human impact. The DeepSat dataset unfortunately consists of image patches which are only $28 \times 28$ large, so that we aren't able to study these images as a 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 images in each category. Here, different categories have different resolutions per pixel, 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 foecitifir querying images from their service Google Maps. 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) to return. Given this set of parameters one can calculate the resolution in meter per pixel [11], which is given by

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

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 location 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 resolution per pixel. The visible Google Maps image is an interpolation from different spectral bands, where the gray band has a higher resolution than the RGB bands. Therefore, the resolution estimated by Eq. 2.1 is not representing the actual pixel resolution for the three color bands. 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 first decided to focus on images of the United States, which allows for a large variety of different terrains. We then used as data source the Aerial Imagery datasets from USGS Earthexplorer (*see link*) which we combined with information about Land Cover and Land Use available from the USGS Land Cover Viewer (*see link*).

FIGURE 2.1: **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.

When looking for images 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, Forest-Woodland that can be found in the USGS Land Cover Viewer. Note that these categories served as a rough geographic reference to pin down geolocations of interest, in order to guarantee a dataset with a good variety of different

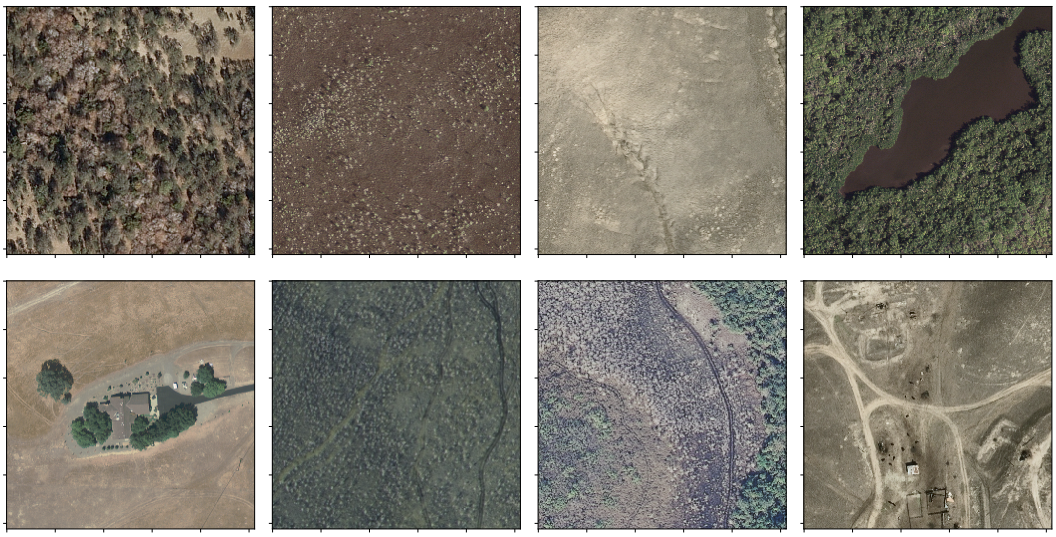

FIGURE 2.2: **Example images of category Shrubland-grassland.** The images in the first row do not contain any human influence, while the images in the second row show influence by humans. The images in this figure have a size of $512 \times 512$ pixels and a resolution of 0.3m per pixel.

FIGURE 2.3: **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 influence by humans. The images in this figure have a size of $512 \times 512$ pixels and a resolution of 0.3m per pixel.

terrains. We also found that it is harder to find images without human, which is why we selected many images from national parks. However, within a given area/terrain we always tried to have images with and without impact.

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 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

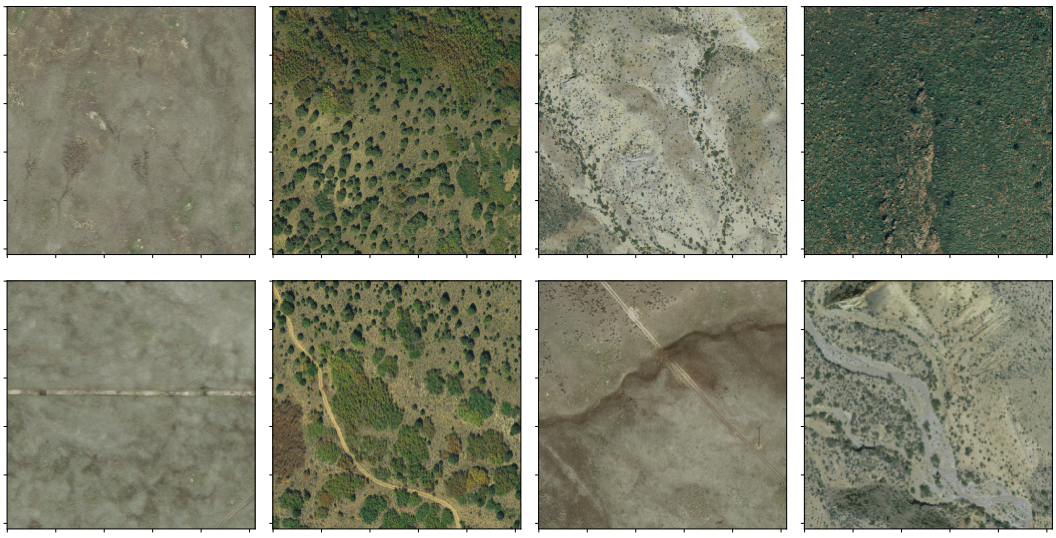

FIGURE 2.4: **Example images of category Semi-desert.** The images in the first row do not contain any human influence, while the images in the second row show influence by humans. The images in this figure have a size of 512 × 512 pixels and a resolution of 0.3m per pixel.

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 images for each dataset.

### 2.4.2 Data Processing and Labeling

Our data processing pipeline consists of the following steps:

- Download large raw images
- Crop images of size 512 × 512 pixels
- Label images with either zero (no human impact), one (minimal human impact), two (obvious human impact)
- Degrade images, i.e. reduce number of pixels and thereby resolution per pixel

Let us discuss each of these steps in more detail. Every raw image was processed, whereas the processed images of size 512 × 512 were saved in a folder named by its category. Note that every raw image resulted in approximately $100 - 150$ processed images, so that we ended up with more than 10,000 images for each dataset.

Within each category of the processed images we labelled a selected portion of the images, by moving them into the folder with the respective label name. The folder structure we used is the following, where pointy brackets '<parameter>' indicate a parameter and 'etc' stands for the three label folders.

```
{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
```

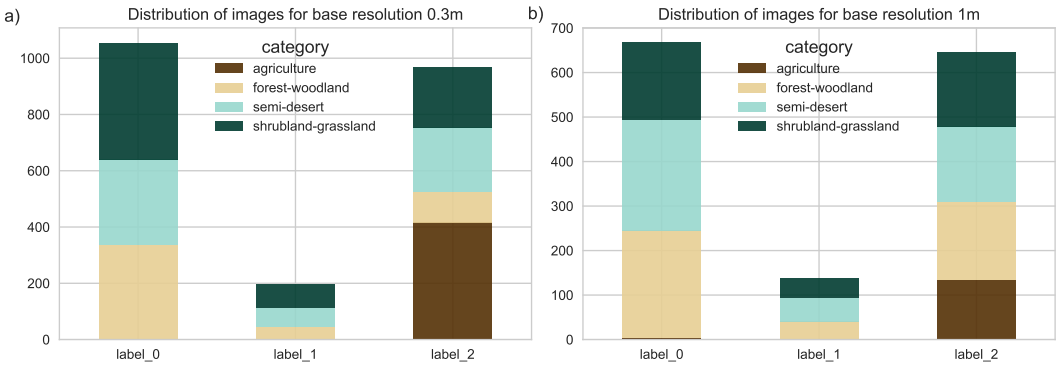


FIGURE 2.5: **Number of images per category and label.** (a) Distribution of images for dataset with resolution of 0.3m per pixel. (b) Distribution of images for dataset with resolution of 1m per pixel.

When labelling we sticked to the following 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 trace were classified into label 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 the class with label zero we classified another set of images with a similar forest type, but containing a building or a street, into the class with label two. The same applies for images in class with label one when they contain e.g. a small walking path. 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 human impact, and not to image artefacts such as color or texture.

In Figures 2.1 - 2.4 we display sample images for each of the four categories Agriculture, Shrubland-grassland, Forest-woodland and Semi-desert. These images belong to the dataset which has a resolution per pixel of 0.3m. Note that in Figs. 2.2 - 2.4 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. 2.1 all contain human influence.

The distribution of categories and labels is shown in Fig. 2.5. Overall, for the 0.3m dataset we classified about 2200 images, and for the 1m dataset we classified about 1450 images. During classification our main goal consisted in creating a balanced dataset between labels zero and two. The minority of images, roughly 10% of all classified images were assigned to label 1. These images were used at random to

investigate the behaviour of the Machine Learning classifier, which is discussed in chapter 4 VERIFY.

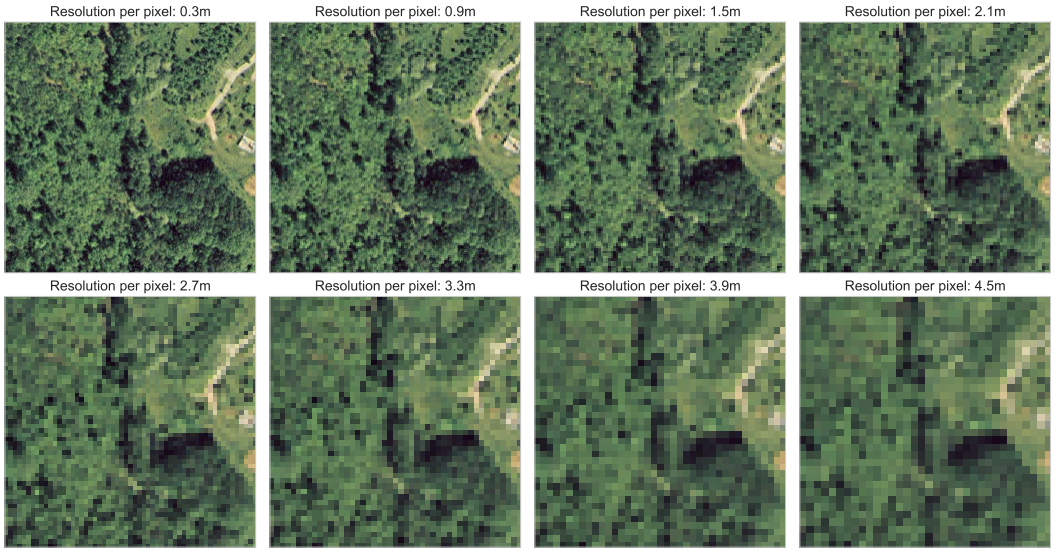


FIGURE 2.6: **Example of image downsampling**. The upper left image has the base resolution, 0.3m per pixel, and a size of $512 \times 512$ pixels whereas the lower right image has the worst resolution, 4.5m per pixel, and a size of $34 \times 34$ pixels. All intermediate images are downsampled by a factor corresponding to the resolution of the actual image divided by the base resolution. For instance, for the lower right image it is 15.

The last step of the data processing pipeline consisted in downsampling the processed and labelled images, to obtain images with a lower resolution per pixel. We used a Lanczos filter [8] for the sampling, which is based on a sinusoidal kernel. In Fig. 2.6) 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).

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 [17], and image [12], video [31] and audio [15] processing while RNNs brought about advances in research and applications on sequential data, i.e. in speech and text [5]. 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 [24] and the analysis of data from elementary particle physics [4].

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 [6], 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 [9].

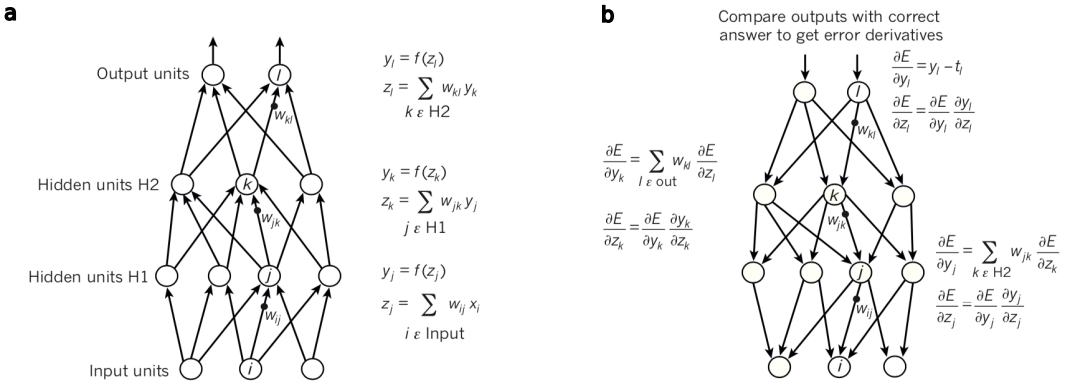


FIGURE 3.1: **Example of 3-Layer Neural Network. (a)** Feed-forward representation of a neural network with two hidden units H1 and H2 and a binary output unit. The inputs to every layer are weighted averages, specified by the weights $w$, of the outputs of the previous layer. In every layer, the outputs 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 [20].

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 [3] or variants of it [16]. The computation of the gradient of the objective function with respect to all weights of the network is accomplished using backpropagation [25] (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 [21]. 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 [10], 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 [32, 19] 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 [26] 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 [18]. 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)).

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

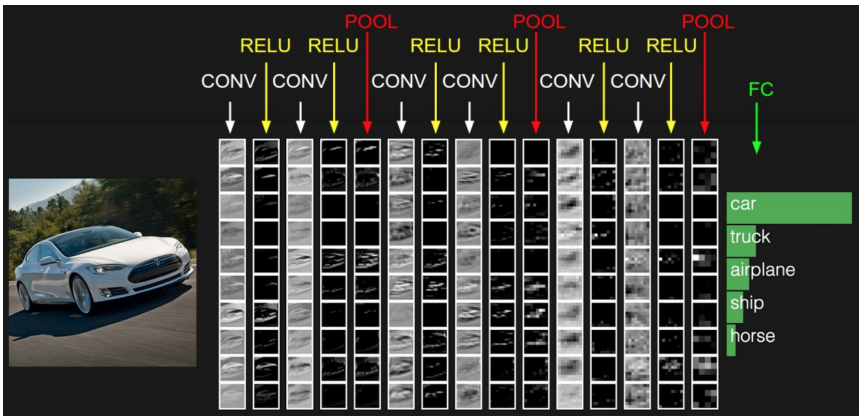


FIGURE 3.2: **Activations of a convolutional neural network used for classification.** The forward pass in this network is computed from left to right. Every column represents a layer of the network and the small images are the filter activations when passing the passing the image through the network.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

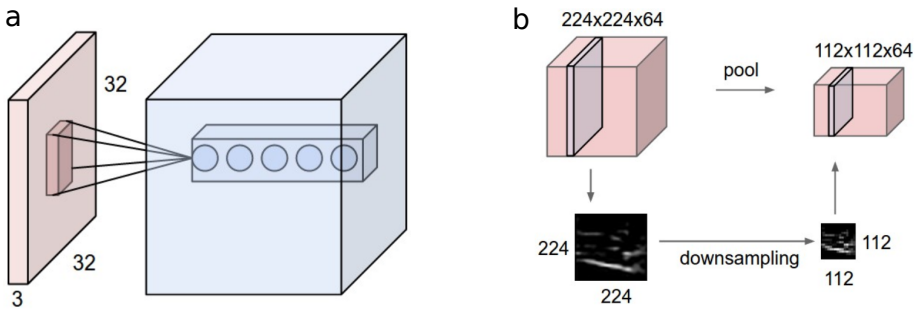


FIGURE 3.3: **Example of a convolutional layer (a) and a pooling layer (b). (a)** In this example a five small filter (represented by the 5 dots in the output volume) are applied to the input image of dimensions $32 \times 32 \times 3$. The filters extend over the entire input depth, but look only at 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 [27].

would, for instance, transform a $4 \times 4$ input image to a $2 \times 2$ 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 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.

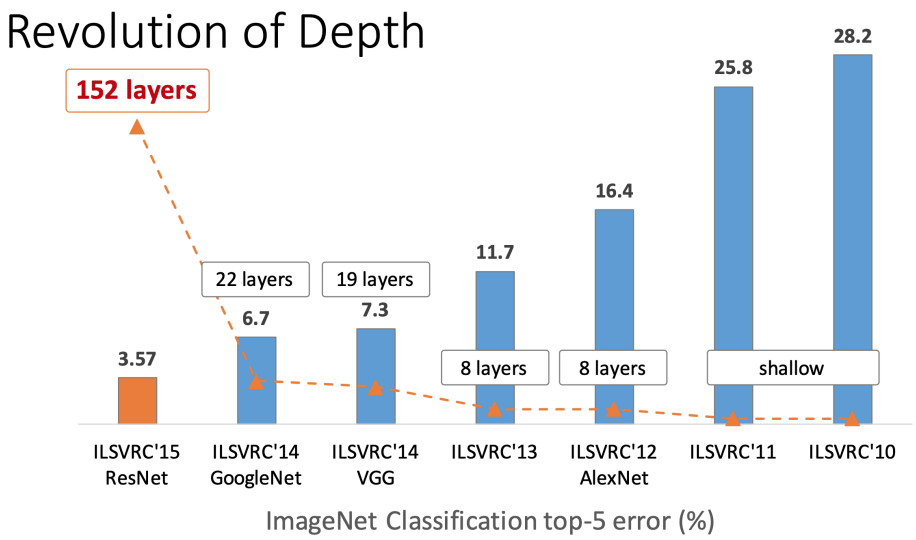


FIGURE 3.4: Top-5 test error of the winning solutions of the Large Scale Visual Recognition Challenge through years 2010 to 2015. Figure is adapted from this link.

The winner of the ILSVRC2013 was the ZFNet [35], 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 [26] had 19 layers and the GoogleNet had 22 layers [30]. To be able to increase the number of layers, researchers from Oxford used very small filters ($3 \times 3$) 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 [13], 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 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.

So instead of simply stacking more and more layers together He et al. developed residual blocks containing an identity mapping additionally to the convolutional

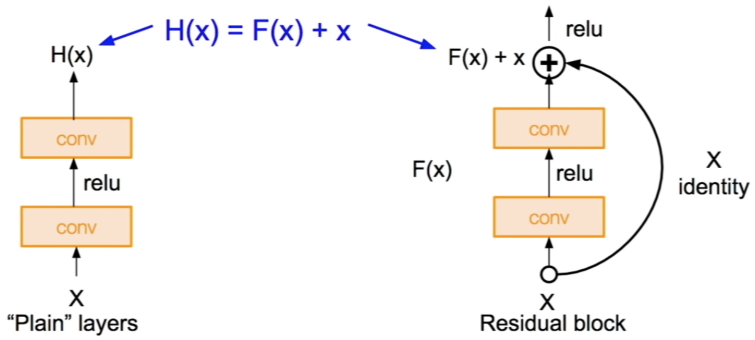


FIGURE 3.5: **Residual block of ResNet architecture.** Comparison between plain convolutional architecture (left) and convolutional architecture using a residual block. For the plain structure the network learns the mapping $H(x)$ while for the residual structure the network learns a residual $F(x) = H(x) - x$. x here is the identity mapping. Figure is adapted from [27].

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.

The architecture of a ResNet with 34 layers is shown in Fig. 3.6. The first convolutional layer has a filter size of $7 \times 7$ while all consecutive filters are chosen to be as small as possible ($3\times$). 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 [29]. We therefore decided to use a ResNet architecture that was pretrained on ImageNet for the experiment in this thesis.

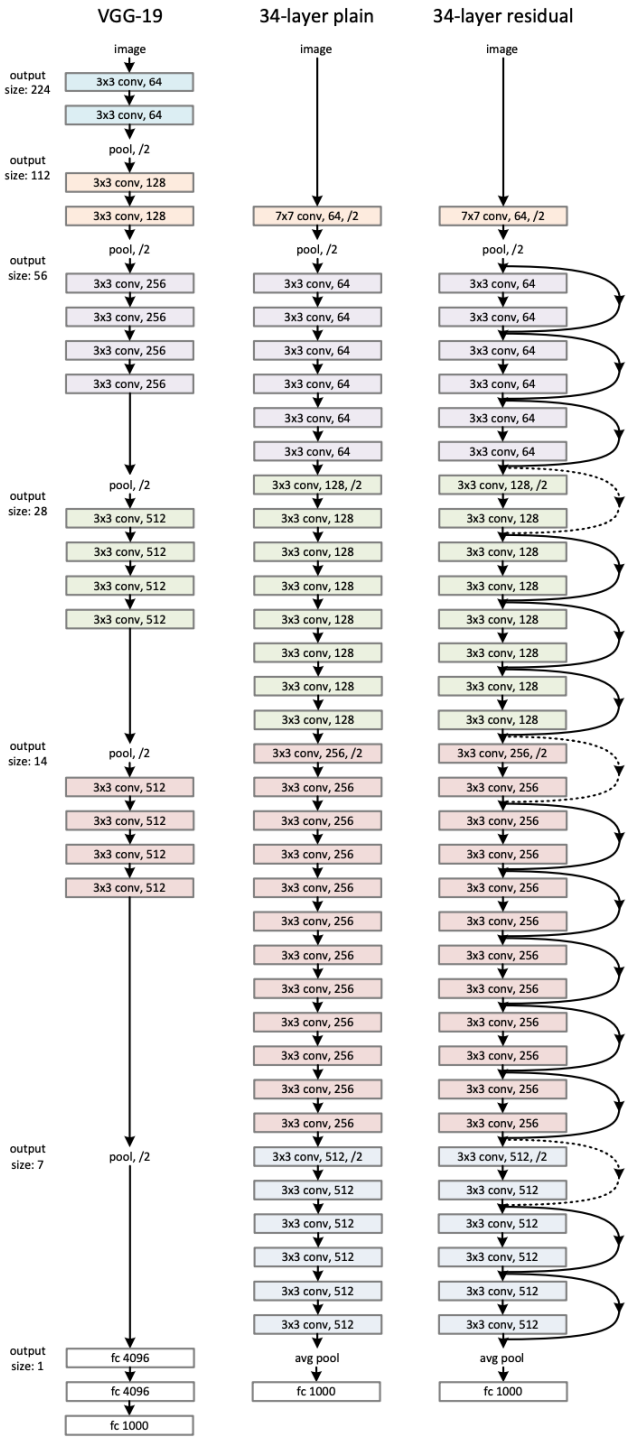


FIGURE 3.6: **Comparison between deep convolutional neural network architectures.** On the left we have a VGG network with 19 layers, in the center a plain ResNet with 34 layers, and on the right a ResNet with 34 layers and residual connections. Figure is adapted from[13].

# 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 [23, 22], SURF [2], Histograms of Oriented Gradients (HOG) [7] and Gabor filters.

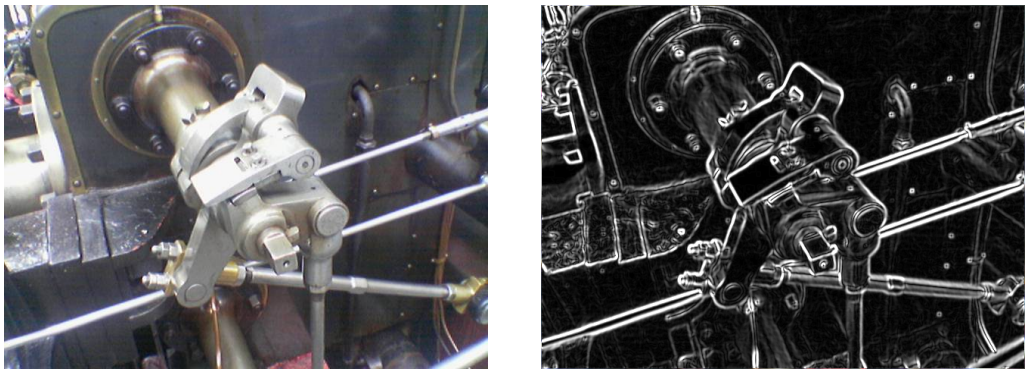

FIGURE 4.1: **Example of the Sobel filter**

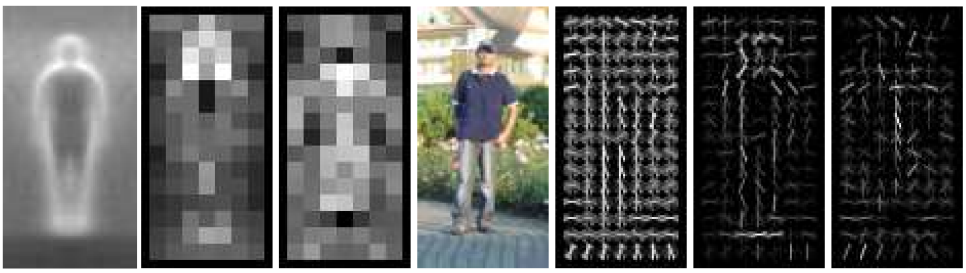

FIGURE 4.2: **Examples of HOG detector [7]**

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 big 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**.

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

### 4.2.1 ResNet activations

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. 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 4.3, 4.4, 4.5 and 4.6 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 (Fully Connected) 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 optimizer and a binary cross-entropy loss function. This same architecture is then used and trained separately for each of the resolutions considered.

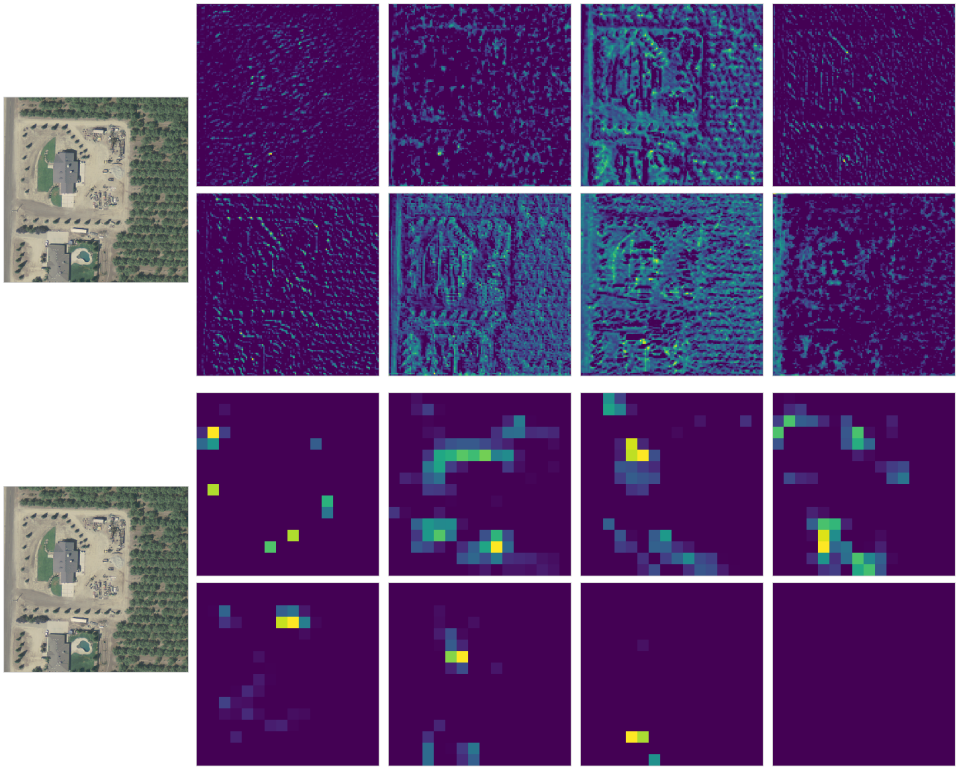

FIGURE 4.3: **ResNet activations of an Agriculture image: $10^{th}$ layer (top) and final layer, $49^{th}$ (bottom).**

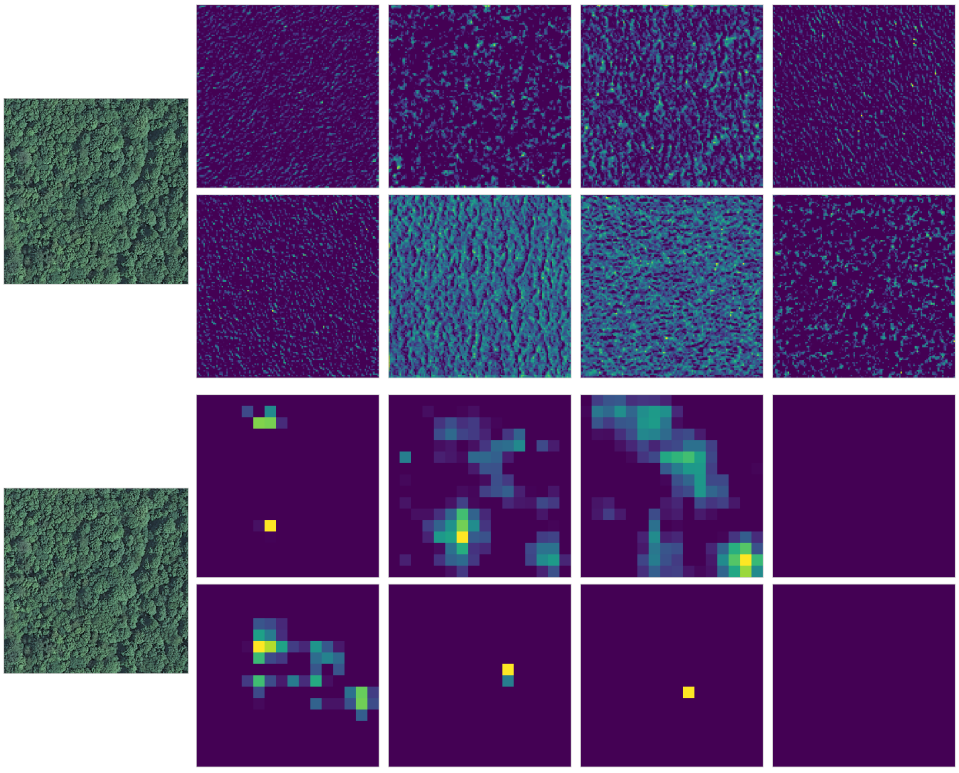

FIGURE 4.4: **ResNet activations of a Forest-woodland image: $10^{th}$ layer (top) and final layer, $49^{th}$ (bottom).**

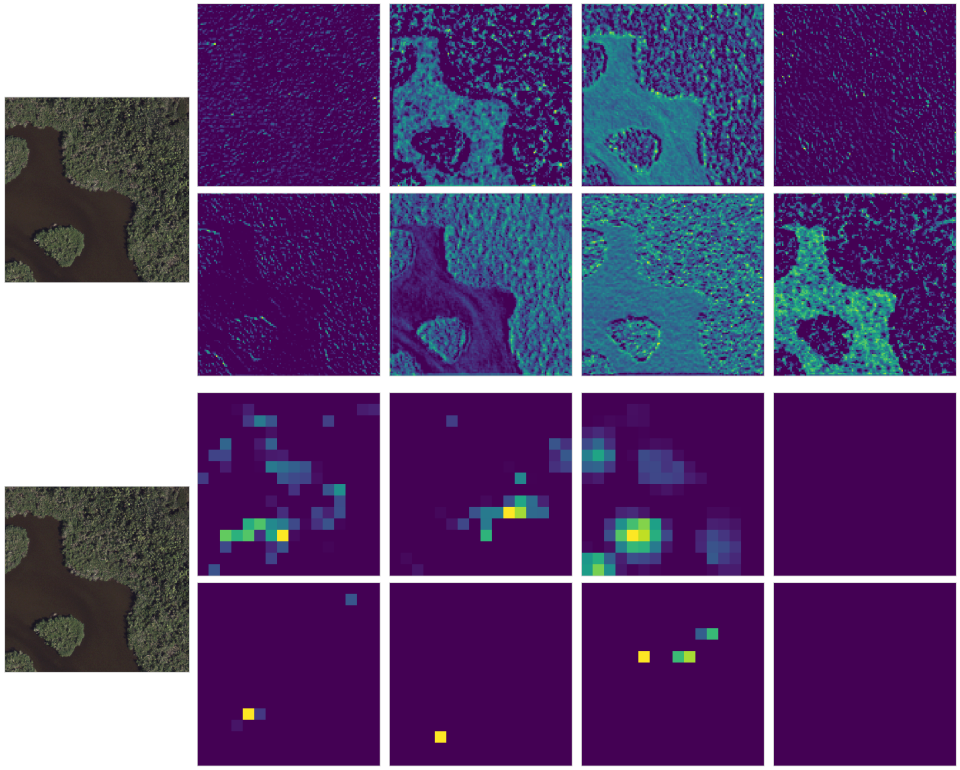

FIGURE 4.5: **ResNet activations of a Shrubland-grassland image:** $10^{th}$ **layer (top) and final layer,** $49^{th}$ **(bottom).**

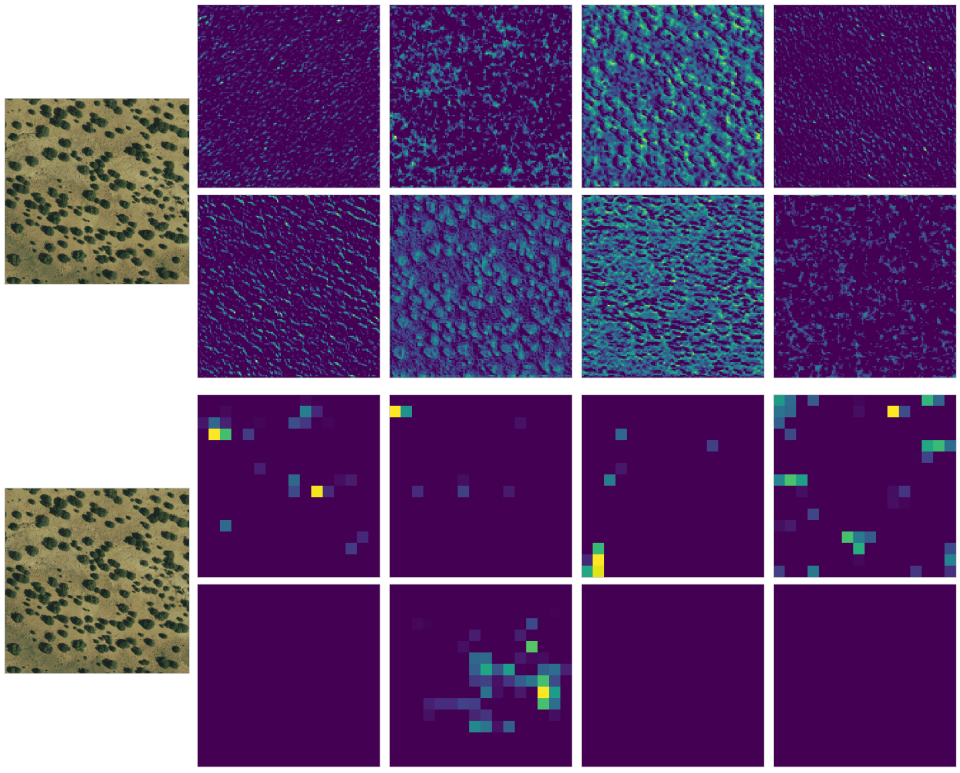

FIGURE 4.6: **ResNet activations of a Semi-desert image:** $10^{th}$ **layer (top) and final layer,** $49^{th}$ **(bottom).**

All this architecture has been implemented with Python and Keras. Figure 4.7 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 4.7: **Model build with Keras**

ARCHITECTURE SCHEMA

### 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.

RE-DO PLOTS

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

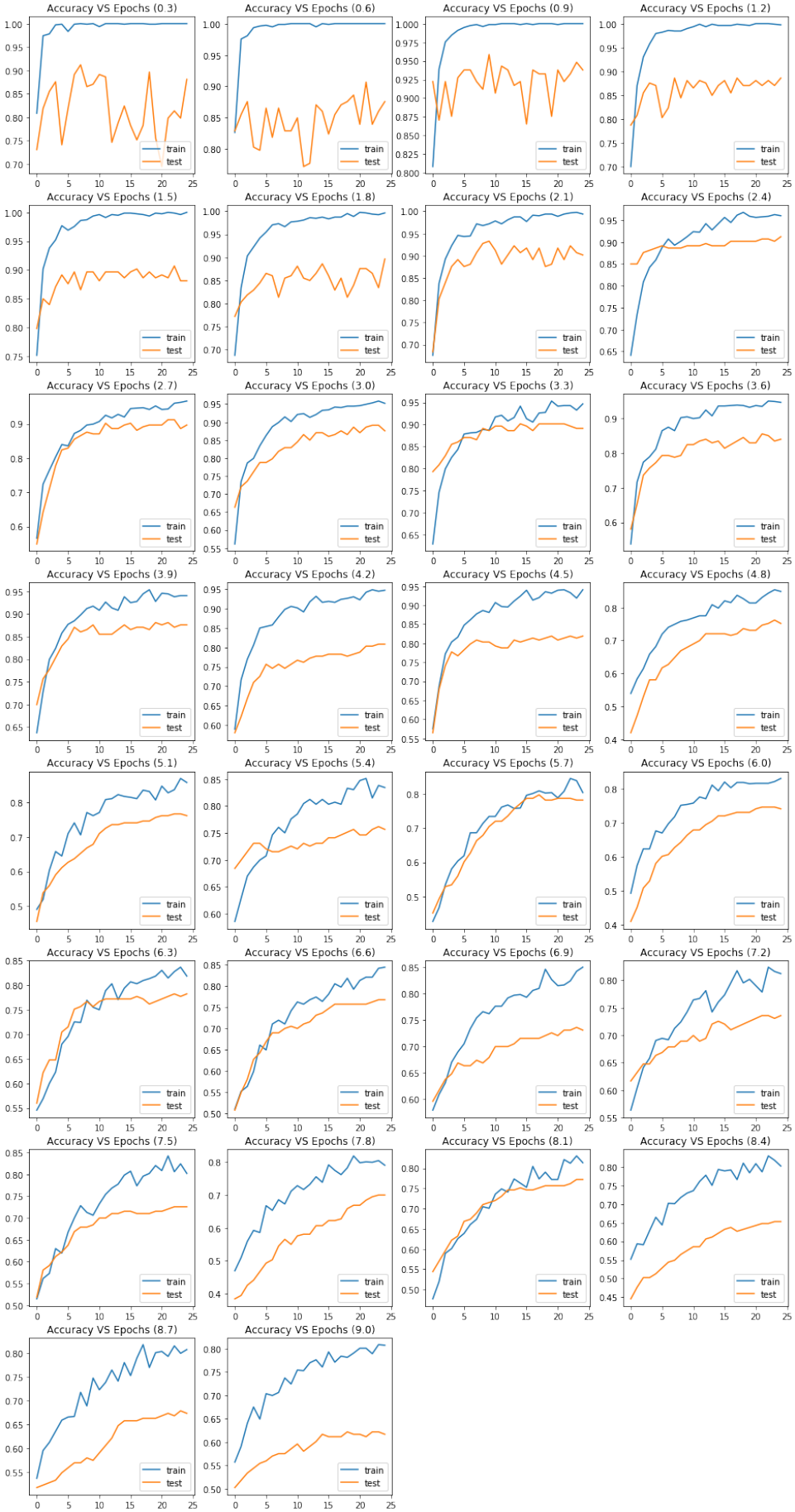


FIGURE 4.8: **Convergence plots for all downgraded resolutions of the $0.3m$ dataset.**

# Chapter 5

# Results

## 5.1 Transfer Learning on Aerial Imagery

## 5.2 Man-made Structures Detection at Different Scale

RE-DO PLOTS

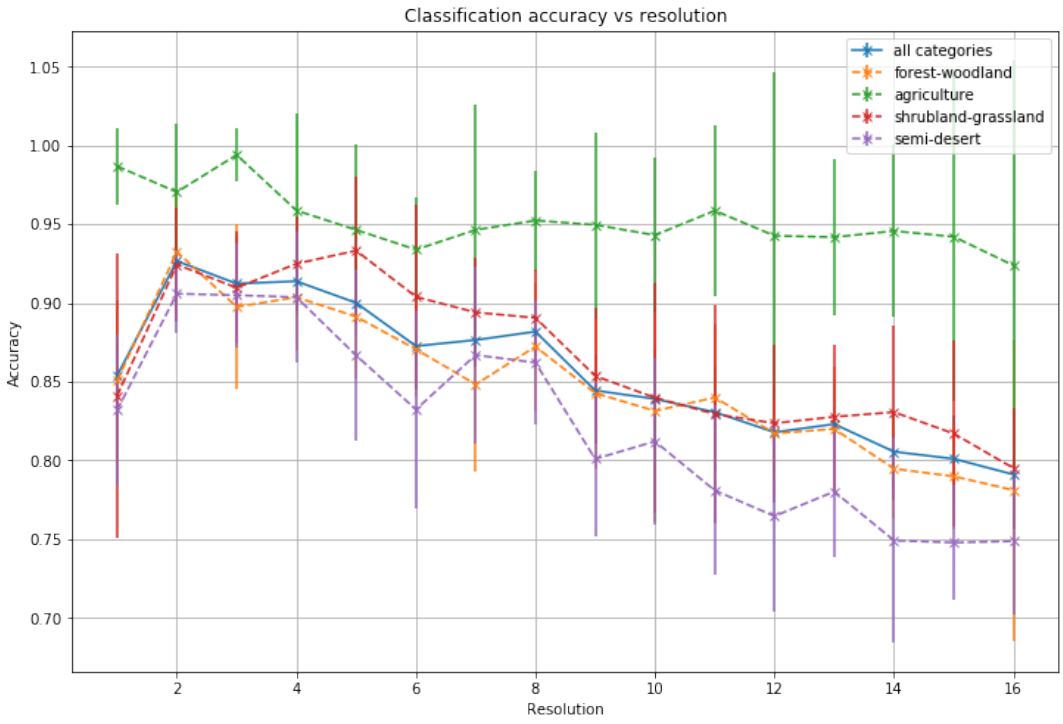


FIGURE 5.1: 1$m$ **dataset**

## 5.3 Cost and Environment impact

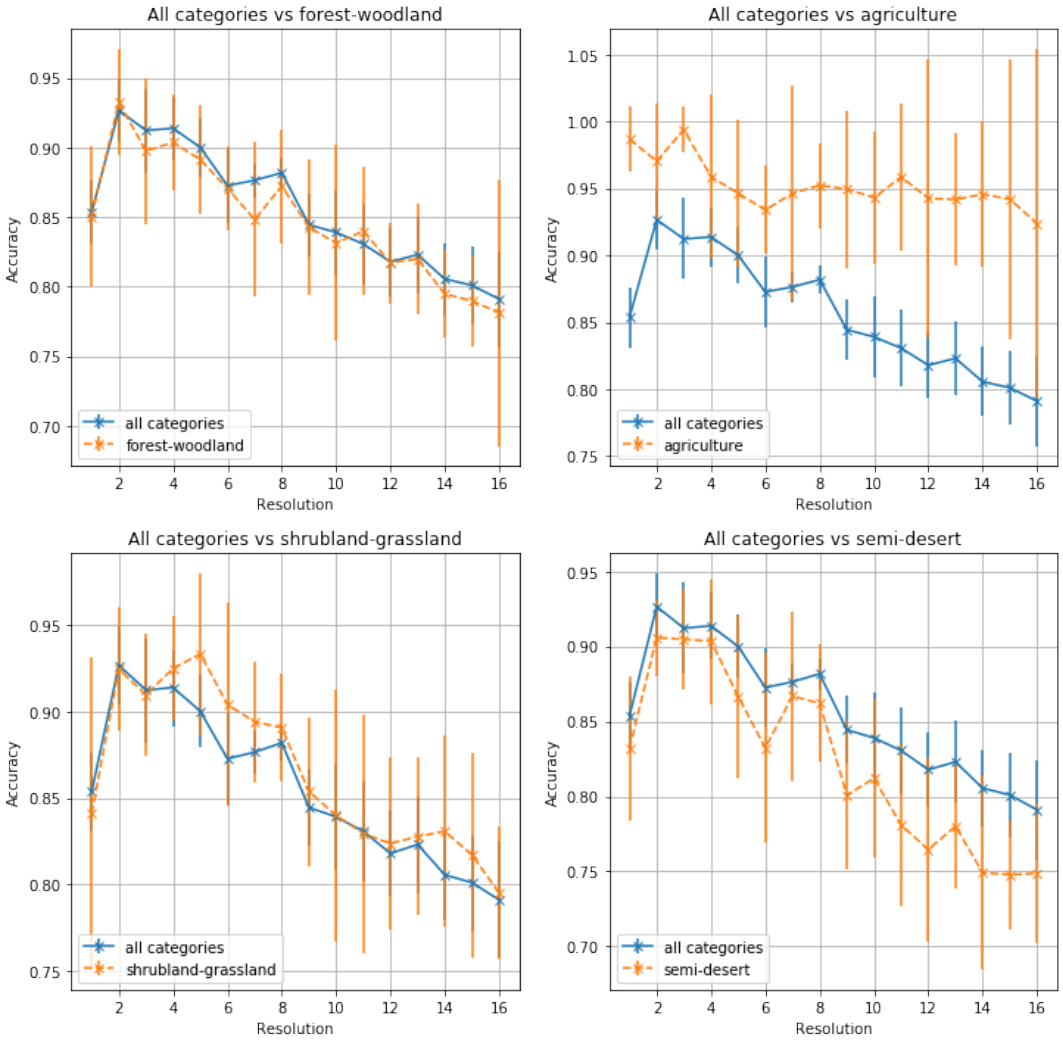


FIGURE 5.2: 1*m* **dataset**

# 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

# Bibliography


[1] 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.

[2] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. *SURF: Speeded Up Robust Features*. Tech. rep. 2006. URL: http://www.vision.ee.ethz.ch/{~}surf/eccv06.pdf.

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

[4] 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.

[5] 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.

[6] 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.

[7] Navneet Dalal and Bill Triggs. *Histograms of Oriented Gradients for Human Detection*. Tech. rep. 2005. URL: http://lear.inrialpes.fr.

[8] 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.

[9] 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.

[10] 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.

[11] *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.

[12] 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.

[13] 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.

[14] 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.

[15] 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.

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

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

[18] 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.

[19] 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.

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

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

[22] David G Lowe. *Accepted for publication in the*. Tech. rep. 2004. URL: https://www.cs.ubc.ca/{~}lowe/papers/ijcv04.pdf.

[23] David G Lowe. *Object Recognition from Local Scale-Invariant Features*. Tech. rep. 1999. URL: https://www.cs.ubc.ca/{~}lowe/papers/iccv99.pdf.

[24] 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.

[25] 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.

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

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

[28] 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.

[29] 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.

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

[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] 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.

[33] 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.

[34] 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.

[35] 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.

[36] 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.