University of Khartoum

Faculty of Engineering

Department of Electrical and Electronic Engineering

Prediction of Regional Biocapacity Using Satellite Imagery and Deep Learning

---

A thesis submitted in partial fulfillment of the requirements for the degree of B.Sc (HONS) Electrical and Electronic Engineering (Software Engineering)

Submitted by

Asim Abdalla Mohamed Mohamed Ali

154047

Omer Kamal Ali Obeid

154057

Supervised by

Dr. Anas Showk

May 2022

Declaration of Authorship

We, **ASIM** and **OMER**, hereby declare that this thesis entitled, **"Prediction of Regional Biocapacity Using Satellite Imagery and Deep Learning"**, is all our own work, except as indicated in the text.

The report has been not accepted for any degree and it is not being submitted currently in candidature for any degree or other reward.

Signed: Asim Abdalla Mohamed Mohamed Ali

Signed: Omer Kamal Ali Obeid

Date: 10/5/2022

Dedication

We dedicate this work to the people who feel and fight for a partition free world and who care for unbiased goodwill towards humankind.

To our parents who did not only raise and nurture us but also taxed their self dearly over the years for our education and intellectual development thank you for making us see this adventure to the end.

'We are all told we don't stand a chance, and yet we stand. We break but we keep going and that is not a flaw.
That's what makes us'

-Elliot Alderson

Acknowledgements

Praise be to Allah, the most beneficent and the most merciful.

This thesis would not have been possible without the support and help from our teachers, friends and families. We would like to express our special appreciation of the following people who supported us with our sincere gratitude:

Our supervisor Dr. Anas Showk, for guiding us patiently in the process of performing this thesis, and giving us technical helps and constructive suggestions for this study.

Our friends Ashraf Hatim, Akram Izzeldin, and Amro Almain for their enthusiastic help, kind support and useful advices in the process of this thesis.

Dr. Alsadig Saeed, for guiding us patiently during the revision process, and his useful suggestions and comments which improved the quality of our thesis.

Our parents and friends, for their endless love and continuous encouragement during the whole process of this thesis.

المستخلص

تعد صور الأقمار الصناعية لاستخدام الأراضي والتغيرات في الغطاء الأرضي ، خاصة تلك التي تسببها الأنشطة البشرية ، أحد أهم مكونات التغيير البيئي العالمي. الزيادة الصارخة في كمية صور الأقمار الصناعية المتاحة في السنوات الأخيرة ، جعلت تفسير هذه البيانات مشكلة صعبة على نطاق واسع ، فلذلك إستخلاص رؤى مفيدة من هذه الصور يتطلب فهماً ثرياً للمعلومات الموجودة فيها. في هذه الدراسة ، نقوم بتجميع الأدبيات المتزايدة التي تستخدم صور الأقمار الصناعية لفهم هذه النتائج ، مع التركيز على الأساليب التي تجمع بين الصور والتعلم العميق. و بما أن الشبكات العصبية التلافيفية حققت مكاسب كبيرة في معالجة الصور الطبيعية، فإن تطبيقها على صور الأقمار الصناعية متعددة الأطياف (حيث تحتوي الصور المدخلة على عدد كبير من القنوات) لا يزال غير مستكشف نسبياً. إننا نقترح تعلم تمثيلات ذات مغزى من صور الأقمار الصناعية من خلال الاستفادة من نطاقاتها الطيفية عالية الأبعاد لقياس ثلاث بنيات مختلفة للشبكات العصبية التلافيفية في سياق بيانات التدريب النادرة والصاخبة لتصنيف عشر مناطق جغرافية مختلفة ، مع دقة تصل إلى أكثر من ٩٥٪ قد تحققت في جميع الأبنية. نوضح أيضاً كيف يمكن استخلاص المعلومات القيمة من النتائج التي تم الحصول عليها من خلال نموذج التعلم العميق وتقديم منهجية لقياس القدرة الحيوية في جزيرة توتي وبورتسودان باستخدام النموذج الأفضل أداءً وربط النتائج التي تم الحصول عليها. و أخيراً فإننا نناقش تطبيقات هذا البحث، ونستكشف القيود التي تعترض التقدم المستقبلي ، ونسلط الضوء على اتجاهات البحث الرئيسية في هذا المجال.

Abstract

Satellite imagery for Land-use and land-cover changes (LULCC), especially those caused by human activities, is one of the most important components of global environmental change. With A stark increase in the amount of satellite imagery available in recent years has made the interpretation of this data a challenging problem at scale, deriving useful insights from such images requires a rich understanding of the information present in them. In this study, we synthesize the growing literature that uses satellite imagery to understand these outcomes, with an emphasis on approaches that combine imagery with deep learning. While Convolutional Neural Networks (CNNs) have made large gains in natural image processing, their application to multi-spectral satellite images (wherein input images have a large number of channels) remains relatively unexplored. We propose to learn meaningful representations from satellite imagery by leveraging its high-dimensionality spectral bands to benchmark three different state-of-the-art CNNs architectures in the context of scarce and noisy training data to classify ten different geographical regions, with an accuracy of more than 95% achieved in all the architectures. We also demonstrate how valuable information may be extracted from the results obtained by the deep learning model and introduce a methodology for gauging biocapacity in Tuti Island and Port Sudan by using the best-performing model and relate the results obtained. Finally, we discuss research and policy applications, explore constraints to future progress, and highlight key research directions for the field.

Table of Contents

List of Figures

List of Tables

List of Abbreviations

Adam	**Ada**ptive **M**oment
AI	**A**rtificial **I**ntelligence
ANN	**A**rtificial **N**eural **N**etwork
BN	**B**atch **N**ormalization
CCE	**C**ategorical **C**ross-**E**ntropy
CNN	**C**onvolutional **N**eural **N**etwork
CLR	**C**yclic **L**earning **R**ate
CV	**C**omputer **V**ision
DCNN	**D**eep **C**onvolutional **N**eural **N**etwork
EO	**E**arth **O**bservation
EM	**E**lectro**M**agnetic
FAIR	**F**acebook's **AI R**esearch lab
FC	**F**ully **C**onnected
FGE	**F**ast **G**eometric **E**nsembling
FoDaFo	**Fo**otprint **Da**ta **Fo**undation
GD	**G**radient **D**escent
GFN	**G**lobal **F**ootprint **N**etwork
GPU	**G**raphics **P**rocessing **U**nit
GSO	**G**eo**s**ynchronous **O**rbit
ILSVRC	**I**mageNet **L**arge **S**cale **V**isual **R**ecognition **C**hallenge
IRS	**I**ndian **R**emote **S**ensing
ISRO	**I**ndian **S**pace **R**esearch **O**rganization
LULC	**L**and **U**se **L**and **C**over
ML	**M**achine **L**earning
MSE	**M**ean **S**quared **E**rror
MSI	**M**ulti-**S**pectral **I**mager

NASA	**N**ational **A**eronautics and **S**pace **A**dministration
NFAs	**N**ational **F**ootprint and Biocapacity **A**ccounts
ReLU	**Re**ctifying **L**inear **U**nit
RS	**R**emote **S**ensing
ResNet	**Res**idual **Net**works
SGD	**S**tochastic **G**radient **D**escent
SSE	**S**nap**S**hot **E**nsembling
SWA	**S**tochastic **W**eight **A**veraging
TLE	**T**wo-**L**ine **E**lement
TPU	**T**ensor **P**rocessing **U**nits
TTA	**T**est **T**ime **A**ugmentation
UAV	**U**nmanned **A**erial **V**ehicles
VHR	**V**ery **H**igh **R**esolution

Chapter 1

Introduction

1.1 Background

Human beings, since the earliest stage of settlement, are dependent on the land for their food production and various sorts of economic development which have been constantly modifying the global landscape. The relentless pressure to meet the needs of the burgeoning population and demand-driven development activities have amplified the stress on earth's land [40]. In this context, anthropogenic activity, and its concomitant land use and land cover (LULC) changes have become an inevitable issue for the present time and accentuate the risks of environmental degradation around the globe [36].

Over half of the world's landscape is influenced by human activities or under some sort of anthropogenic development and since the historic past, many natural resources have been heavily used or even depleted in the worst cases [19]. The impacts of these widespread LULC changes in the natural environment are multi-faceted, including climate change, alteration of the hydrological cycle, increased water extraction, impairment of water quality, degradation of soil nutrients, amplified surface erosion, and loss of biodiversity [5]. Therefore, information on land use and land cover, changing trends, and optimal use of the land resources have become predestined criteria for land use planning and effective natural resources management of an area.

1.2 Problem Statement

In order to make informed decisions pertaining to the environment, we are equipped with senses that allow us to observe it. This enables us to make effective changes around us as appropriate or desirable. By taking a step back and extending this general process to a large scale, we notice a need for understanding complex phenomena around the world such as urban growth, climate change, biodiversity studies, and socioeconomic trends. This process very generally is referred to as *earth observation* and has applications in disaster response, resource management, and precision farming among others.

Historically, manual analyses of satellite and aerial imagery were feasible primarily because the volume of images available was quite low - but that is not the case now. Relevant information extraction from images thus becomes a problem with the high volume of data we deal with today. A major component of these problems is an annotation (or labeling), wherein one identifies the structures and patterns visible in a satellite image.
Over the years, research in the computer vision community has addressed this problem of automating the analysis of large-scale data in different ways. Machine learning techniques have proven to be strong candidates here, especially in the last few years. Machine learning research stems from the idea that a computer can be given the ability to learn, as a human would do, without being explicitly programmed. Deep learning is a subset of machine learning and refers to the application of a set of algorithms called neural networks, and their variants.

1.2.1 Objectives

Given the above background, the aims of this thesis can be stated as follow:

- To benchmark three state-of-the-art deep convolutional neural network models that takes in satellite data and categorizes this data into ten different predefined geographical regions, and outline how this classification model may be used to detect changes in land use or land cover.
- Utilize the best performing model to classify the same regions in Tuti Island and Port Sudan.
- We present an end-to-end approach for the estimation of a geographic region's biocapacity in Tuti Island and Port Sudan.

1.3 Methodology

Our dissertation work revolves around using computer vision techniques along with satellite imagery data and selected three preselected state-of-the-art CNNs architectures to classify 10 different geographical regions. These architectures were trained and built using PyTorch (A machine learning framework) with the help of Google Colab to write the required python notebooks. Then we proposed a deep learning model to classify the same regions in Tuti Island and Port Sudan and introduce a methodology to quantify the biocapacity of a geographic region in them.

1.4 Thesis Structure

The dissertation comprises **five** chapters, as follows: Chapter 1 gives a detailed overview and scope of the dissertation. It includes a general introduction, motivation for the research work, research objectives, relevance of the research, and thesis roadmap. Chapter 2 discusses theories that the research relies upon by defining key concepts, evaluating and combining relevant theories, this Chapter also presents, and explains the existing literature and the context of the study. It also provides smooth-flowing discussion of what research has already been done. Chapter 3 examines and presents different methods and techniques used to undertake the research, it includes **how the data is collected and analyzed** and **any tools or materials used in the research.** Chapter 4 explains, interprets, and elaborates the research findings and discusses the significance and implications of the results found. Chapter 5 summarizes the research outputs, explains limitations, and provides future research directions and recommends possible extensions on the present work.

Chapter 2

Theoretical Background & Literature Review

2.1 Satellites and Earth Observation

Satellites launched into space are mission specific, among which Earth Observation (EO) is one. The first spacecraft to have taken pictures of the Earth was Explorer 6, launched in 1959, and the number of Earth Observation, or remote sensing satellites has only increased since then. With the move toward automated drone delivery systems and autonomous vehicles, for instance, there is a greater demand for the use of satellite imagery that can be used as extraneous information for sensor fusion in the vehicles, to get a clearer context of surroundings. Understanding urban structures from images hence becomes important.

What is Remote Sensing?

Remote sensing is the process of detecting and monitoring the physical characteristics of an area by measuring it's reflected and emitted radiation at a distance (typically from satellite or aircraft). Special cameras collect remotely sensed images, which help researchers 'sense' things about the Earth.

2.1.1 Orbits

Satellites can be placed in several types of orbits around Earth. The three common classes of orbits are *low-Earth orbit* (approximately 160 to 2,000 km above Earth), *medium-Earth orbit* (approximately 2,000 to 35,500 km above Earth), and *high-Earth orbit* (above 35,500 km above Earth). Satellites orbiting at 35,786 km are at an altitude at which their orbital speed matches the planet's rotation, and are in what is called *geosynchronous orbit* (GSO). In addition, a satellite in GSO directly over the equator will have a *geostationary orbit*. A geostationary orbit enables a satellite to maintain its position directly over the same place on Earth's surface [30].
As mentioned both geosynchronous and geostationary satellites orbit at 35,786 km above Earth, geosynchronous satellites have orbits that can be tilted above or below the equator. Geostationary satellites, on the other hand, orbit Earth on the same

plane as the equator. These satellites capture identical views of Earth with each observation and provide almost continuous coverage of one area [30].

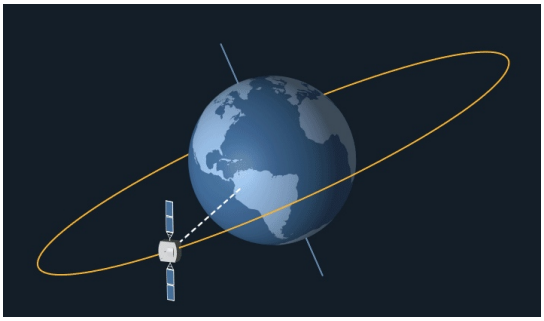


FIGURE 2.1: A Spacecraft in a Geostationary Orbit [30].

2.1.2 Satellite Sensors

Sensors, or instruments, aboard satellites and aircraft, use the Sun as a source of illumination or provide their own source of illumination, measuring the energy that is reflected back. Sensors that use natural energy from the Sun are called *passive sensors*; those that provide their own source of energy are called *active sensors*.

(i) **Passive sensors:**
Include different types of radiometers (instruments that quantitatively measure the intensity of electromagnetic radiation in select bands) and spectrometers (devices that are designed to detect, measure, and analyze the spectral content of reflected electromagnetic radiation). Most passive systems used by remote sensing applications operate in the visible, infrared, thermal infrared, and microwave portions of the electromagnetic spectrum. These sensors measure land and sea surface temperature, vegetation properties, cloud and aerosol properties, and other physical properties [30].
Note:
That most passive sensors cannot penetrate dense cloud cover and thus have limitations observing areas like the tropics where dense cloud cover is frequent.

(ii) **Active sensors:**
Include different types of radio detection and ranging (radar) sensors, altimeters, and scatterometers. The majority of active sensors operate in the microwave band of the electromagnetic spectrum, which gives them the ability to penetrate the atmosphere under most conditions. These types of sensors are useful for measuring the vertical profiles of aerosols, forest structure, precipitation and winds, sea surface topography, and ice, among others [30].

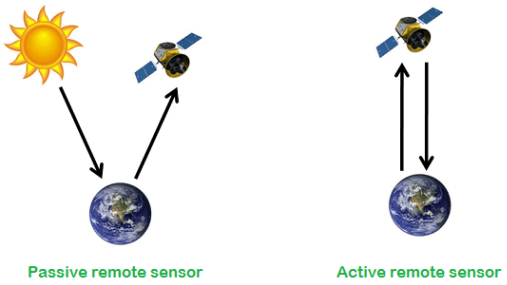


FIGURE 2.2: Passive and Active Remote Sensors [23].

Sensors are also capable of capturing other specific regions in the electromagnetic spectrum. Examples of prominent EO satellite programs include the LANDSAT program from the National Aeronautics and Space Administration (NASA), and the Indian Remote Sensing (IRS) program, from the Indian Space Research Organization (ISRO) [30].

The reason for using the electromagnetic (EM) spectrum lies in the fact that each and every object reflects, transmits, and absorbs light differently, depending on its chemical composition This property of an object is referred to as its spectral signature and is what makes remote sensing possible.

It is also important to take note of interference by the earth's atmosphere, which absorbs certain wavelengths in the EM spectrum. Hence, sensors are designed to measure specific ranges of wavelengths alone.

2.1.3 Observing with the Electromagnetic Spectrum

Electromagnetic energy, produced by the vibration of charged particles, travels in the form of waves through the atmosphere and the vacuum of space. These waves have different wavelengths (the distance from wave crest to wave crest) and frequencies; a shorter wavelength means a higher frequency. Some, like radio, microwave, and infrared waves, have a longer wavelength, while others, such as ultraviolet, x-rays, and gamma rays, have a much shorter wavelength. Visible light sits in the middle of that range of long to shortwave radiation. This small portion of the energy is all that the human eye is able to detect [30].

Some waves are absorbed or reflected by atmospheric components, like water vapor and carbon dioxide, while some wavelengths allow for unimpeded movement through the atmosphere; visible light has wavelengths that can be transmitted through the atmosphere. Microwave energy has wavelengths that can pass through clouds, an attribute utilized by many weather and communication satellites [30].

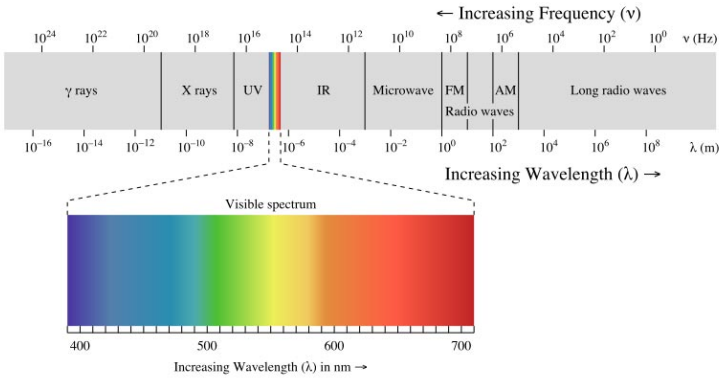


FIGURE 2.3: Illustration of the electromagnetic spectrum showing specific spectral wavelengths [9].

The primary source of the energy observed by satellites is the Sun. The amount of the Sun's energy reflected depends on the roughness of the surface and its albedo, which is how well a surface reflects light instead of absorbing it. Snow, for example, has a very high albedo and reflects up to 90% of incoming solar radiation. The ocean, on the other hand, reflects only about 6% of incoming solar radiation and absorbs the rest. Often, when energy is absorbed, it is re-emitted, usually at longer wavelengths. For example, the energy absorbed by the ocean gets re-emitted as infrared radiation [30].

All things on Earth reflect, absorb, or transmit energy, the amount of which varies by wavelength. Just as our fingerprints are unique to us, everything on Earth has a unique spectral fingerprint. Researchers use this information to identify differently Earth features as well as different rock and mineral types.

2.1.4 Image Resolution

Resolution plays a role in how data from a sensor can be used. Depending on the satellite's orbit and sensor design. There are four types of resolution to consider for any dataset-radiometric, spatial, spectral, and temporal.

1. **Radiometric resolution:** is the amount of information in each pixel, i.e. the number of bits representing the energy recorded.

2. **Spatial resolution:** is defined by the size of each pixel within a digital image and the area on Earth's surface represented by that pixel.

3. **Spectral resolution:** is the ability of a sensor to discern finer wavelengths, that is, having more and narrower bands. Many sensors are considered to be multi-spectral, meaning they have between 3-10 bands. Sensors that have hundreds to even thousands of bands are considered to be hyperspectral. The narrower the range of wavelengths for a given band, the finer the spectral resolution [30].

4. **Temporal resolution:** is the time it takes for a satellite to complete an orbit and revisit the same observation area. This resolution depends on the orbit, the sensor's characteristics, and the swath width. Because geostationary satellites match the rate at which Earth is rotating, the temporal resolution is much finer, at about 30s - 1min. Polar orbiting satellites have a temporal resolution that can vary from 1 day to 16 days [30].

2.2 Machine Learning

Machine learning is a field that is focused on the construction of algorithms that make predictions based on data. A machine learning task aims to identify (to learn) a function $f : X \to Y$ that maps the input domain X (of data) onto output the domain Y (of possible predictions)[15].
Functions f are chosen from different function classes, dependent on the type of learning algorithm that is being used. Mitchell (1997) defines "learning" as follows: "A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E" [15]. The performance measure P tells us quantitatively how well a certain machine learning algorithm is performing. For a classification task, the accuracy of the system is usually chosen as the performance measure, where accuracy is defined as the proportion for which the system correctly produces the output. Experience E that machine learning algorithms undergo are datasets. These datasets contain a set of examples that are used to train and test these algorithms.

2.2.1 Classification of Machine Learning Algorithms

Machine learning algorithms can be largely classified into three categories by the type of datasets that are used as experience. These categories are *supervised learning*, *unsupervised learning*, and *reinforcement learning*.

1) **Supervised learning:** Supervised learning systems make use of *labeled* datasets $(x, y) \in X \times Y$, where x represents a data point and y is the corresponding true prediction for x. This training set of input-output pairs is used to find a deterministic function that maps any input to output, predicting future input-output observations while minimizing errors as much as possible [11].

2) **Unsupervised learning:** Unsupervised learning systems use *unlabeled* datasets to train the system. The objective of unsupervised learning is to derive structure from unlabeled data by investigating the similarity between pairs of objects and is usually associated with density estimation or data clustering.

3) **Reinforcement learning:** Reinforcement learning systems do not experience a fixed dataset, but a feedback loop between the system and its experiences [11]. A dynamic environment is considered in which state-action-reward triples are observed as the data. The objective of reinforcement learning is mapping situations to actions with the goal of maximizing rewards.

Other learning systems exist that are a combination of two categories, such as **semi-supervised learning** that uses both labeled and unlabeled data.

2.2.2 Solving Machine Learning Tasks

A wide variety of tasks exist that could be solved with machine learning. Two popular machine learning tasks are **regression analysis** and **classification**.
In *regression analysis*, the relationship amongst variables is approximated, for the successful prediction of a value given some input. This task is solved by outputting a function $f : \mathbb{R}^n \to \mathbb{R}$ that fits the data [28]. Regression analysis can be used for example to forecast future stock prices in the trading world.
In *classification*, the machine is asked to determine the category n that a certain input belongs. The task can be solved by outputting a function $f : \mathbb{R}^n \to \{1, \cdots, n\}$ [28]. A popular classification problem is object recognition for intelligent systems. Classification can be used for example to classify objects in a warehouse to determine the correct destination of each object, with current state-of-the-art object recognition making use of deep learning algorithms [28].

2.3 Neural Networks

LeCun, Bengio, and Hinton, famous pioneers in deep learning, define that neural networks are "computational models that are composed of multiple processing layers [able] to learn representations of data with multiple levels of abstraction" [22], which implies that they receive input, process it, and produce an output. The model's internal structure determines which tasks it can solve.

2.3.1 Biological Neurons

The exact workings of the human brain are still a mystery. Yet, some aspects of this amazing processor are known. In particular, the most basic element of the human brain is a specific type of cell that, unlike the rest of the body, doesn't appear to regenerate. Because this type of cell is the only part of the body that isn't slowly replaced, it is assumed that these cells are what provide us with our abilities to remember, think, and apply previous experiences to our every action. These cells, all 100 billion of them, are known as neurons. Each of these neurons can connect with

up to 200,000 other neurons, although 1,000 to 10,000 is typical.
The fundamental processing element of a neural network is a neuron. This building block of human awareness encompasses a few general capabilities. Basically, a biological neuron receives inputs from other sources, combines them in some way, performs a generally nonlinear operation on the result, and then outputs the final result.
Within humans, there are many variations on this basic type of neuron, further complicating man's attempts at electrically replicating the process of thinking. Yet, all natural neurons have the same four basic components. These components are known by their biological names - dendrites, soma, axon, and synapses. Dendrites are hair-like extensions of the soma which act like input channels. These input channels receive their input through the synapses of other neurons. The soma then processes these incoming signals over time. The soma then turns that processed value into an output which is sent out to other neurons through the axon and the synapses.

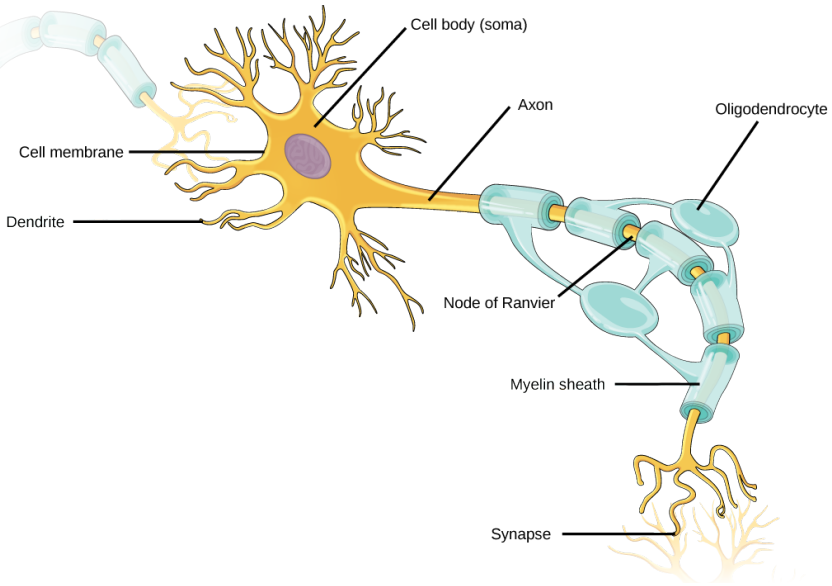


FIGURE 2.4: Biological Neuron Components [25].

2.3.2 Perceptron

The perceptron is a mathematical model of a biological neuron. While in actual neurons the dendrite receives electrical signals from the axons of other neurons, in the perceptron these electrical signals are represented as numerical values. At the synapses between the dendrite and axons, electrical signals are modulated in various amounts. This is also modeled in the perceptron by multiplying each input value by a value called the weight. An actual neuron fires an output signal only when the total strength of the input signals exceeds a certain threshold. This phenomenon is modeled in a perceptron by calculating the weighted sum of the inputs to represent the total strength of the input signals and applying a step function on the sum to determine its output. As in biological neural networks, this output is fed to other

perceptrons.
Perceptrons were developed in the 1950s and 1960s by the scientist Frank Rosenblatt, inspired by earlier work by Warren McCulloch and Walter Pitts. A perceptron takes several binary inputs, $x_1, x_2, \ldots$, and produces a single binary output.
The perceptron isn't a complete model of human decision-making, it only solves a small range of problems like linear separable classification, However, it can not be used for non-separable classification data.

2.3.3 Artificial Neural Networks

Artificial Neural Networks (ANNs) are relatively crude electronic models based on the neural structure of the brain, The brain basically learns from experience. These biologically inspired methods of computing are thought to be the next major advancement in the computing industry. Even simple animal brains are capable of functions that are currently impossible for computers. Computers do rote things well, like keeping ledgers or performing complex math. But computers have trouble recognizing even simple patterns much less generalizing those patterns of the past into actions of the future.
ANNs are built by a series of "neurons" (or "nodes") which are organized in layers [29]. These neurons exhibit global behavior determined by the established connections between the various processing elements and the related parameters within the neural network architecture. Each connection that connects the neurons in consecutive layers is weighted.

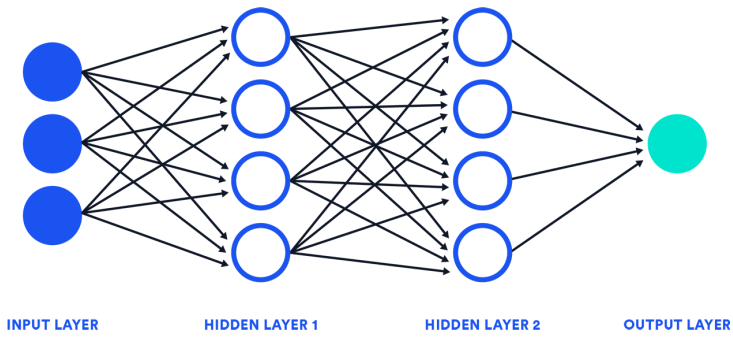


FIGURE 2.5: Artificial Neural Networks Model [7].

2.3.4 Deep Learning Neural Networks

Deep learning models are composed of *layers*, which are themselves built up by artificial *neurons*. These neurons are inspired by their biological counterpart and hence give the term "neural network" its name. Each processing layer applies a certain mathematical operation to its input. These operations depend on the connections

between the neurons and their connection strengths; the *weights* of the connections. For each model input x, there is the desired network output (*ground truth*) y_t and the actual (predicted) output y_p. As the actual output depends on all the weights inside the network, there is a set of weights, for which y_p is closest to y_t . In order to find this optimum, a technique called *backpropagation* is used, Discussed in subsection 2.4.7.1. The input to such a network could be for example an image and the task to classify the displayed content. The output of later layers becomes harder and harder to interpret by humans, which is why neural networks are often criticized as nontransparent [13]. This is the key difference between deep learning and traditional machine learning; in the traditional case, the features a model uses to make a prediction are handcrafted by humans and therefore understandable. In the case of deep learning, they are a product of the backpropagation process and a result of numerical optimization. The term "deep learning" refers to the fact, that the neural networks employed can have many layers (100 or even more)[24].
There are many different types of neural networks; in this thesis, we will concentrate on so-called convolutional neural networks (CNNs), as they are best suited for the classification and segmentation of images.
The input to the network is an image, which usually consists of three channels (red, green, blue) in the depth dimension, but in remote sensing satellite images can have many more channels or bands.

2.4 Basic Architecture Of Convolutional Neural Networks

CNNs are feedforward networks in that information flow takes place in one direction only, from their inputs to their outputs. Just as ANN is biologically inspired, so are CNNs. The visual cortex in the brain, which consists of alternating layers of simple and complex cells, motivates their architecture.
CNN architectures come in several variations; however, in general, they consist of convolutional and pooling (or subsampling) layers, which are grouped into modules. Either one or more fully connected layers, as in a standard feedforward neural network, follow these modules. Modules are often stacked on top of each other to form a deep model.
First, a convolution acts on the input image, which is then followed by an activation function and a pooling operation. This pattern is repeated until a final convolution produces a vector containing class probabilities. The convolution learns the features necessary for classification, the activation function introduces a non-linearity into the network, which is essential for the learning process and the pooling reduces the lateral dimension and filters out the most relevant information. As this is a widespread

sequence in the structure of neural networks, these operations will be explained next in this order.

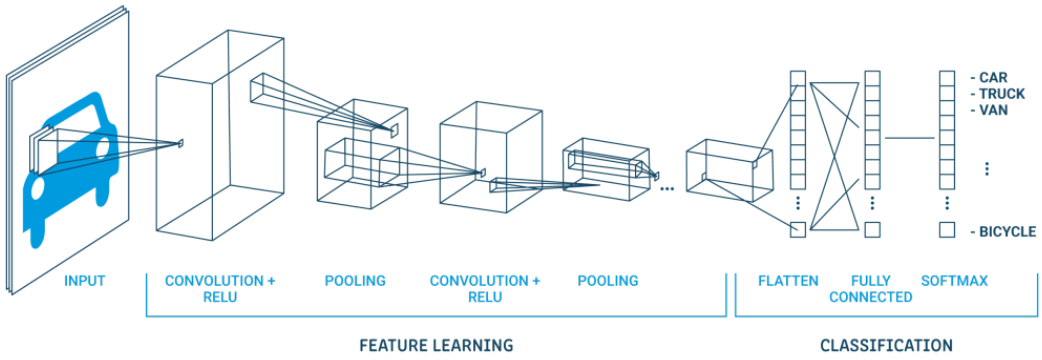


FIGURE 2.6: Convolutional Neural Network image classification pipeline [1].

2.4.1 Convolutional Layers

The convolutional layers serve as feature extractors, and thus they learn the feature representations of their input images. The neurons in the convolutional layers are arranged into feature maps. Each neuron in a feature map has a receptive field, which is connected to a neighborhood of neurons in the previous layer via a set of trainable weights are sometimes referred to as a filter banks. Inputs are convolved with the learned weights in order to compute a new feature map, and the convolved results are sent through a nonlinear activation function. All neurons within a feature map have weights that are constrained to be equal; however, different feature maps within the same convolutional layer have different weights so that several features can be extracted at each location. More formally, the kth output feature map Y_k can be computed as:

$$Y_k = f(W_k * x) \tag{2.1}$$

where the input image is denoted by x; the convolutional filter related to the kth feature map is denoted by W_k; the multiplication sign in this context refers to the $2D$ convolutional operator, which is used to calculate the inner product of the filter model at each location of the input image; and $f(\cdot)$ represents the nonlinear activation function, The details of convolutional operation are explained in Appendix Convolution Operation. Nonlinear activation functions allow for the extraction of nonlinear features. Traditionally, the sigmoid and hyperbolic tangent functions were used; recently, rectified linear units (ReLUs) have become popular. Their popularity and

success have opened up an area of research that focuses on the development and application of novel Deep Convolutional Neural Network (DCNN) activation functions to improve several characteristics of DCNN performance.

2.4.2 Neural Network's Activation Functions

There are many different activation functions available for neural networks. Their most important feature is that they are non-linear functions. In contrast, the convolution is a linear operation. This has the effect that, if one applies many convolutions after each other to form an output, there exists a single convolution that forms the very same output. This means that a neural network without non-linearities can essentially perform only linear regression tasks. Therefore, non-linearities play a key role in neural networks. The following is not a comprehensive list, but rather gives an overview of the most important activation functions [11].

(i) **Sigmoid activation:** The sigmoid activation or logistic function is a widely used activation function and has a characteristic "S" shape. It has continuous derivatives and is bounded between 0 and 1. This is why it is used to map arbitrary values to this interval. The sigmoid function can for example be used as the last activation in binary classification tasks, but not for categorical classification, and has the advantage that its first derivative is simple to compute.

$$S(x) = \frac{1}{1 + e^{-x}} \tag{2.2}$$

$$S'(x) = S(x)(1 - S(x)) \tag{2.3}$$

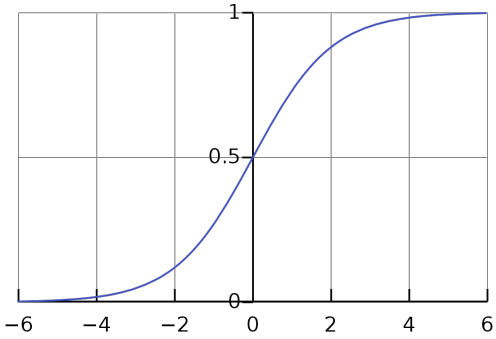


FIGURE 2.7: Sigmoid activation function [34].

(ii) **ReLU activation:** The rectifying linear unit (ReLU) activation was used to model the behavior of biological neurons as early as 1975 [10]. Biological neurons start to fire, when their cumulated input exceeds a certain threshold and mostly sit idle the remaining time [4]. The ReLU function mimics this behavior by mapping all negative inputs to zeros and by letting all positive input

pass through, so the threshold value is zero. It has proved to allow for efficient training of neural networks [20]. The derivative of the ReLU function is the Heaviside- or step-function.

$$\text{ReLU}(x) = \max(0, x) \tag{2.4}$$

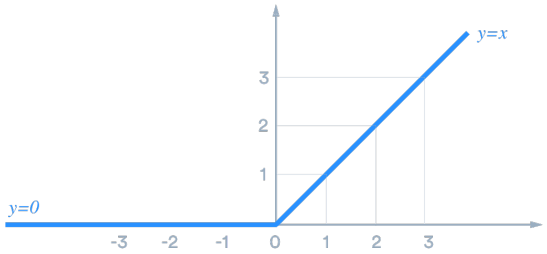


FIGURE 2.8: The Rectified Linear Unit (ReLU) activation function [2].

(iii) **Softmax activation:** The softmax activation is a generalization of the sigmoid function that can handle multi-dimensional input. Let $\boldsymbol{z}$ be a C-dimensional input vector to the function. It then returns C values between zero and one, that add up to one. The function attenuates the highest value of the input vector, which turns out closest to one. This makes the softmax function ideal for categorical classification tasks, where the network has to "decide" which output value is most relevant.

$$\sigma(x) = \frac{e^{z_j}}{\sum_{k=1}^{C} e^{z_k}} \quad \text{for } j = 1, \ldots, C \tag{2.5}$$

2.4.3 Pooling Layer

The purpose of the pooling layers is to reduce the spatial resolution of the feature maps and thus achieve spatial invariance to input distortions and translations. Initially, it was common practice to use average pooling aggregation layers to propagate the average of all the input values, of a small neighborhood of an image to the next layer. However, in more recent models, max pooling aggregation layers propagate the maximum value within a receptive field to the next layer. Formally, max pooling selects the largest element within each receptive field such that

$$Y_{kij} = \max_{(p,q)\in\mathfrak{R}_{ij}} x_{kpq} \tag{2.6}$$

where the output of the pooling operation, associated with the kth feature map, is denoted by Y_{kij}, x_{kpq} denotes the element at location (p, q) contained by the pooling region $\mathfrak{R}_{ij}$, which embodies a receptive field around the position (i, j) (Yu et al., 2014).

After the convolutional layers, the pooling layers are perhaps the most important. They recapitulate the responses of neighboring neurons from the same kernel map and thus reduce the dimensions of their input representations. Significantly, they provide DCNNs with their spatial invariance, This is to **decrease the computational power required to process the data** through dimensionality reduction. Furthermore, it is useful for **extracting dominant features** which are rotational and positional invariant, thus maintaining the process of effectively training of the model.

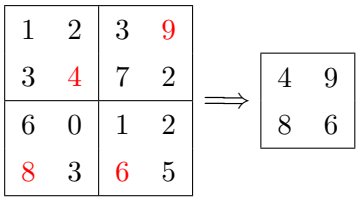


FIGURE 2.9: Example of Max Pooling Operation.

2.4.4 Regularization Mechanisms

DCNNs are very expressive models, capable of learning exceptionally complicated relationships between their inputs and outputs. However, with limited training data, even for larger data sets, many of these complicated mappings are due to sampling noise. Thus they exist in the training set rather than in the test set, irrespective of whether they are drawn from the same data distribution. This leads to overfitting, which can be mitigated by regularization. Although the easiest and most common method to reduce overfitting is data augmentation, it requires a larger memory footprint and comes at a higher computational cost. Furthermore, despite the regularization effects of several other diverse methods, including L_1 and L_2 regularization, stopping training early, stochastic pooling, unique activation functions, model averaging, novel loss functions, and the successful application of Dropout which has led to its extensive use and inspired numerous improvements.

In Dropout, each unit of a layer's output is retained with probability p; else, it is set to zero with probability $1-p$, with 0.5 being a common value of p When Dropout is applied to a fully connected layer of a DCNN (or any Depp Neural Network (DNN)), the output of the layer $r = [r_1, r_2, \ldots, r_d]^T$, can be expressed as:

$$r = m \star a(W_v) \tag{2.7}$$

where $\star$ denotes the element-wise product between a binary mask vector m and the matrix product between the input vector $v = [v_1, v_2, \ldots, v_n]^T$ and the weight matrix W (with dimensions $d \times n$), followed by a nonlinear activation function. The primary

benefit of Dropout is its proven ability to significantly reduce overfitting by effectively preventing feature coadaptation.

2.4.5 Loss Functions

The loss function is a measure for the error of a prediction with respect to the ground truth. The choice of the loss function is crucial for the success of the network training and depends on the task to solve. However, there is not only one possible loss per task and the decision on which to use has to be made on empirical grounds and based on the task's properties. If the neural network is for example used to approximate a function, which is essentially a regression, the mean squared error (MSE) is an appropriate choice. When it comes to the task of classification, other loss functions are suited better because the dependent variables in classification tasks are discrete (0 or 1) and not continuous as in regression. Classifications are normally represented as vectors; when there are C classes, the vector is C-dimensional and each entry corresponds to a class. If a ground truth vector should represent class i, the i-th entry is set to one and all others are zero. In contrast, the network output is the output of e.g. the softmax function (eq. (2.5)), with real values that are normalized to one, as vector entries - this is best understood as a probability distribution. The difference between the true and predicted probability distribution is commonly measured using categorical cross-entropy (CCE) [11].

$$H(y_t, y_p) = -\sum_{i=1}^{C} \mu_i y_{t,i} \log(y_{p,i}) \tag{2.8}$$

y: Vector containing the ground truth or the prediction.

μ_i: Weight parameter for class i.

In a classification task with mutually exclusive classes, only one entry of the true probability distribution can be one. In this case, the CCE ignores the predictions for all other classes and the prediction for the true class is logarithmically weighted. Therefore, the resulting loss can be between infinity for a completely wrong, and zero for a perfect prediction. In segmentation tasks, the CCE is calculated per pixel of the output probability map and then averaged over all pixels.

2.4.6 Optimization Techniques

- *Gradient-Based Learning*: In fully supervised DCNNs, the loss function, which is in most cases the softmax loss is usually minimized using some form of stochastic gradient descent (SGD). For this technique, the gradient is evaluated using the popular backpropagation algorithm (Explained in sub-section

2.4.7.1). While gradient descent, made popular by Rumelhart et al. (1986), was used in many of the early CNNs.

- *Enhanced Initialization Schemes*: Poor initialization of DCNN parameters, which are typically in the millions and in particular, their weights can hamper the training process because of the vanishing/exploding gradient problem (Bengio et al., 1994), and hinder convergence. Thus, their initialization is extremely critical.

- *Batch Normalization*: In addition to having a large number of parameters, the training of DCNNs is convoluted by a phenomenon known as internal covariate shift, which is caused by changes to the distribution of each layer's inputs because of parameter changes in the previous layer. This phenomenon has severe consequences, which include slower training due to lower learning rates, the need for careful parameter initializations, and complexities when training DCNNs with saturating nonlinear activations. To reduce the consequences of internal covariate shift, Ioffe and Szegedy proposed a technique known as batch normalization (BN). This technique introduces a normalization step, which is simply a nonlinear transformation applied to each activation, that fixes the means and variances of layer inputs. To allow integration with SGD (Bottou, 1998, 2010), which also uses mini-batches during training, BN computes the mean and variance estimates after mini-batches rather than over the entire training set.
  Specifically, for a mini-batch $B = x1, \ldots, n$, with activation x and dimension n, the mini-batch mean and variances are first computed by $\mu_B \leftarrow \frac{1}{n}\sum_{j=1}^{n} x_j$ and ${\sigma_B}^2 \leftarrow \frac{1}{n}\sum_{j=1}^{n}(x_j - \mu_B)^2$, respectively. The jth dimension is then normalized by the following expression,

$$\hat{x_j} \leftarrow \frac{x_j - \mu_B}{\sqrt{{\sigma_B}^2 + \epsilon}} \tag{2.9}$$

  where ϵ is a constant, introduced for arithmetical stability. The normalized values $\hat{x}$ are then scaled and shifted for enhanced representation by the following expression:

$$Y_j \leftarrow \zeta\hat{x_j} + \beta \equiv BN_{\zeta,\beta}(x_j), \tag{2.10}$$

  where ζ and β are learnable parameters. The result of the second transformation Y is propagated to other layers of the network.

- *Skip Connections*: Even though increasing network depth generally leads to increased performance, improving the classification accuracy of DCNNs is not as straightforward as simply adding layers. some of the complications include

overfitting, an increased computational burden, memory footprint, and degradation. In particular, degradation remains a key challenge. Thus, overcoming it is imperative in order to investigate the benefits of very deep networks for several applications; this has led to work that focuses on skip (or shortcut) connections within DCNNs.

- *Residual networks*: Residual networks, also address the degradation problem, using skip connections. The main idea of residual networks is to learn an additive residual function with respect to an identity mapping that is based on the preceding layer's inputs, accomplished by attaching an identity shortcut connection. Residual modules perform the following computation:

$$y_1 = h(x_1) + \mathcal{F}(x_1, W_{l,k|\leqslant k\leqslant K}) \tag{2.11}$$

$$x_{l+1} = f(y_1) \tag{2.12}$$

where the input to the lth residual module is denoted by x_l, W_l represents its weights and biases, K is the number of layers in a module, $\mathcal{F}$ represents the residual function such as a stack of convolutional filters, f is the operation that follows element-wise addition, and h is an identity mapping of the form $h(x_l) = x_l$. Residual networks have brought about some empirical success; in particular, they have performed exceptionally well on the demanding ImageNet challenge[1].

2.4.7 Training

CNNs, and ANNs in general use learning algorithms to adjust their free parameters (i.e. the biases and weights) in order to attain the desired network output. The most common algorithm used for this purpose is backpropagation. Backpropagation computes the gradient of an objective (also referred to as a cost/loss/performance) function to determine how to adjust a network's parameters in order to minimize errors that affect performance. A commonly experienced problem with training CNNs, and in particular DCNNs, is overfitting, which is poor performance on a held-out test set after the network is trained on a small or even large training set. This affects the model's ability to generalize on unseen data and is a major challenge for DCNNs that can be assuaged by regularization, which is surveyed in sub-section 2.4.4.

[1]The ImageNet Large Scale Visual Recognition Challenge is a benchmark in object category classification and detection on hundreds of object categories and millions of images. The challenge has been run annually from 2010 to the present, attracting participation from more than fifty institutions.

2.4.7.1 Backpropagation

Backpropagation is the algorithm, that enables neural networks to "learn". It is an optimization procedure for the internal network state; the weights and entries of the convolutional kernels.
Consider the following sequence: we have a network that has produced an output from a given input. We can then calculate the error the network made and loss, according to sub-section 2.4.5. The loss depends on the desired network output, the actual output, and by that also all the weights inside the network. We now want to update the set of weights ω by some δ in a way that the loss is minimized.

$$\omega_{new} = \omega_{old} + \delta \tag{2.13}$$

$$L(y_t, y_p : \omega_{new}) < L(y_t, y_p : \omega_{old}) \tag{2.14}$$

How do we calculate the δ in a way that minimizes the loss? As the loss and all internal functions of the network are differentiable, we can calculate the derivative of the loss function with respect to every weight. This gradient represents the amount and direction of influence each weight has on the loss. The gradient points uphill in the direction of the steepest slope in the "loss-landscape". Knowing this, we can write down an equation that optimizes the weights:

$$\omega_{new} = \omega_{old} - \alpha \nabla_{\omega_{old}} L \tag{2.15}$$

Where ∇_ω is the nabla-operator, which takes the derivative of L with respect to every weight and α is the so-called learning rate, which is essentially a step-width that defines how far each update step goes against the direction of the gradient. The minus sign is chosen because we want to descend into the loss landscape. Equation (2.15) is the most simple form of an update rule: The gradient descent.

How the gradient is calculated? The answer is trivial to give analytically: By using the chain rule of differential calculus. But computationally, it can be challenging to find the gradients in an efficient manner and much of the processing time is spent in this step. The chain rule defines how composite functions are derived: Let z be a variable, that depends on y, where y itself depends on x. The derivative of z after x can then be written as the product of two simpler derivatives.

$$\frac{\partial z}{\partial x} = \frac{\partial z}{\partial y} \frac{\partial y}{\partial x} \tag{2.16}$$

Neural networks can be seen as computational graphs, as each node (operation) is connected to others. Figure 2.10 depicts a very simple graph: The repeated application of the same function f to some input w. The intermediate results are then given by $x = f(w), y = f(x)$ and finally $z = f(y)$. The function f here symbolically stands for some function that appears in the network, such as the convolution or ReLU. The output z could be the loss function and w, which is now the input, could as well be some weight parameter deep inside the network. Using the chain rule, we can then calculate the derivative of z after w:

$$\frac{\partial z}{\partial w} = \frac{\partial z}{\partial y}\frac{\partial y}{\partial x}\frac{\partial x}{\partial w} = f'(y)f'(x)f'(w) \tag{2.17}$$

We immediately see that we have to calculate the intermediate results x and y before we can calculate the derivatives. This is called the *forward pass*, which is followed by the *backward pass*, Where we actually calculate the derivatives. The backward pass is only needed during the training of the network, which is why the pure application of the network is faster.

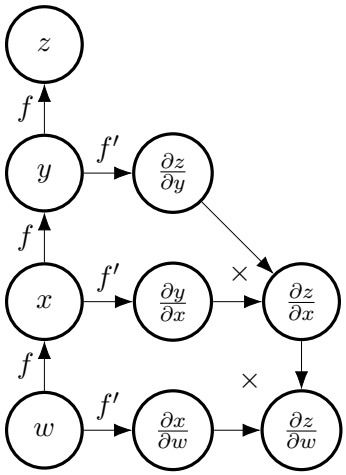


FIGURE 2.10: Scheme of the backpropagation process. Once the forward pass is complete, one can start to calculate the derivatives, going from top to bottom. The "×" indicates multiplication. The desired final derivative is shown in the lower right.

2.5 Computer Vision

Computer vision is a branch of computer science that aims to replicate portions of the human visual system's complexity, allowing computers to recognize and interpret things in photos and videos in the same manner that people do. Until recently, computer vision could only do restricted tasks.

Artificial intelligence has made enormous strides in recent years, surpassing humans in several tasks related to object detection and categorization thanks to developments in artificial intelligence and innovations in deep learning and neural networks. The

quantity of data we collect today, which is subsequently utilized to train and improve computer vision is one of the driving drivers behind its rise. Along with a massive volume of visual data (about 3 billion photographs are posted on the internet every day), the computational power needed to evaluate it is now available. The accuracy rates for object detection have increased as the area of computer vision has progressed with new hardware and algorithms. Today's systems have improved from 50% accuracy to 99 percent in less than a decade, making them more accurate than humans in swiftly reacting to visual inputs.

Early computer vision investigations began in the 1950s, and by the 1970s, it was being used commercially to discern between typed and handwritten text. Today, computer vision applications have evolved tremendously.

Computer vision is, in some ways, all about pattern recognition. So one approach to teach a computer to interpret visual data is to give it photos, thousands, if not millions, of labeled images, and then apply them to various software procedures, or algorithms, that allow the computer to search for patterns in all the parts that correspond to those labels.

Prior to the introduction of deep learning, the capabilities of computer vision were severely limited, requiring a great deal of manual coding and work on the part of developers and human operators. If you wanted to conduct facial recognition, for example, you would need to do the following:

1/ Create a database: You had to take individual photographs in a certain format of all the subjects you wished to follow.

2/ Annotate images: You would then have to enter numerous crucial data points for each individual photograph, such as the distance between the eyes, the width of the nasal bridge, the distance between the top lip and the nose, and hundreds of other measures that characterize each person's unique traits.

3/ Capture new photos: Next, whether from photographs or video sources, you'll need to capture fresh images. Then you had to repeat the measuring procedure, this time highlighting the important places on the image. You also have to consider the angle from which the photo was shot.

After all of this painstaking labor, the program would eventually be able to compare the measures in the new image to those in its database and inform you if it matched any of the profiles it was tracking. In actuality, very little automation was used, and the majority of the job was done manually. The error margin was still significant.

Machine learning offered a fresh perspective on computer vision challenges. Developers no longer have to manually code every rule into their vision apps thanks to machine learning. Instead, they created "features," which were tiny apps that could

recognize certain patterns in photographs. They then utilized a statistical learning technique to find patterns, categorize photos, and detect objects in them, such as linear regression, logistic regression, decision trees, or support vector machines (SVM). Many challenges that were previously difficult for traditional software development tools and methodologies were solved thanks to machine learning.
Deep learning offered a fundamentally new way of approaching machine learning. Deep learning is a powerful technique for computer vision. Creating a solid deep learning algorithm usually boils down to accumulating a huge quantity of labeled training data and fine-tuning parameters like neural network type and a number of layers, as well as training epochs. Deep learning is easier and faster to build and deploy than prior forms of machine learning.
For example, using deep learning to create a facial recognition program just requires you to create or select a pre-built algorithm and train it using examples of the faces of the individuals, it must recognize. The neural network will be able to recognize faces without any more instructions on characteristics or measurements if there are enough instances (many examples).

2.5.1 Computer Vision Challenges

- Assisting computers in seeing is proving to be difficult.
- Creating a machine that sees as we do is a surprisingly tough endeavor, not just because it's difficult to program computers to do it, but also because we don't fully understand how human vision works.
- Understanding the perception organs, such as the eyes, as well as the interpretation of perception within the brain, is necessary for studying biological vision. Much progress has been achieved, both in terms of tracking the process and uncovering the system's tricks and shortcuts, though there is still a long way to go, as with any brain research.

Despite recent advances that have proved outstanding, we are still far from solving computer vision. However, a number of healthcare institutions and businesses have already discovered methods to adapt CV systems driven by CNNs to real-world situations. This pattern is unlikely to change very soon.

2.6 Computer Vision and Satellite Imagery

Computer Vision (CV), deep learning, and artificial intelligence (AI) methods for applications in remote sensing can support and resolve challenges for large satellite image data sets by utilizing high-performance based models to collect and identify features in an environment with precise accuracy and speed.

Computer Vision algorithms can eliminate noise and enhance satellite and aerial picture data, allowing for better analysis of broad regions, such as classifying objects, features, detecting change, data fusion, cloud removal, and spectrum analysis from imagery, overcoming remote sensing data problems.

These models can provide near real-time reports for areas with complex feature distribution, such as mineral mapping, agriculture, climate change, environmental monitoring, and wildlife conservation, using images collected by satellites or unmanned aerial vehicles (UAV). They use CV, AI, ML (Machine Learning), and deep learning methods.

The earth observation sector is clearly booming in terms of raw data. Investing in open data from satellite constellations like MODIS, Landsat, and Sentinel has democratized access to real-time satellite images of the whole world (although at a lesser resolution than you're used to seeing on Google Maps). Meanwhile, cloud companies such as Amazon Web Services and Google Cloud have gone so far as to offer free storage of satellite data, further boosting worldwide use of these photos.

The problem is that deciphering the substance of satellite pictures is not a simple process. For more than 70 years, experts in the field of remote sensing have used computational approaches to solve the problem of interpreting earth images. Even simple tasks like detecting building footprints or differentiating tree canopy in metropolitan areas were time-consuming sub-specialties of the discipline until recently. Then, in 2012, the "deep learning revolution" ushered in a whole new era of helpful algorithms that could be used to satellite images to provide cutting-edge findings.

Most common deep learning architectures aren't built for imagery that's a gigabit or bigger, has over a dozen channels (the majority of which aren't visible), and is saved in spatially referenced file formats like GeoTIFF and JPEG2000. While breakthroughs in machine learning for computer vision jobs have prompted some, such as Google's former CEO Eric Schmidt called picture identification a "solved issue", there are still significant challenges for even simple processes for those interested in applying these approaches to satellite images.

Satellite imagery, unlike web-based images, has generally been difficult to obtain. In recent years, however, satellite imagery-focused analogs to ImageNet have filled the vacuum. For example, SpaceNet is a semi-annual competition and dataset collection aimed at extracting information from satellite photos. Similar contests, such as xView and Functional Map of the World has greatly increased the availability of high-quality geographic datasets for testing new algorithms.

Despite the fact that satellite imagery training datasets are publicly available, the difficulty of actually handling the data or modifying the architecture of typical machine learning models to operate with it is still primarily in the research phase. Azavea has put a lot of effort into making this last piece of the jigsaw easier, particularly

with our Raster Vision open source python library for applying machine learning to satellite photos. Users can perform three dirty things in an elegant way with Raster Vision:

1) Convert satellite images into a format that most machine learning systems can understand. You can "split" a huge image into hundreds or thousands of tiny photos that can be used to train a model and then retroactively patched back together while retaining the important geographical information necessary for most mapping operations.

2) Explain how to use standard machine learning frameworks, such as PyTorch, to quickly train models, analyze their findings, and run several experiments in parallel.

3) Package trained models so they may be quickly deployed in various situations and used to predict fresh data.

Satellite imagery is still considerably more available than the commercial and scientific communities' capacity to analyze it. Raster Vision is just a first step toward a future in which extracting answers from satellite pictures is as simple as asking questions. Satellite imagery is an invaluable resource for addressing issues such as diverse and essential as assessing the effects of climate change, projecting agricultural yields, and estimating global progress toward the Sustainable Development Goals. Machine learning, and especially fast-evolving sub-disciplines like deep learning, hold the potential of making satellite imagery processing simpler, more scalable, and more universally applicable.

2.7 Ensemble Learning Techniques

Ensembling is a technique that combines individual detectors to improve detection performance. In recent data science competitions, such as Kaggle[2] , assembling methods have been widely used to get high rankings. The few remote sensing publications that address ensembling mostly deal with mid-resolution photos and earth observation applications like land use classification, but never with Very High Resolution (VHR) images for military applications or object detection.
Recent improvements in the quality and quantity of VHR Earth observation photos from commercially available satellites have enabled breakthroughs in tools for mapping and processing objects and areas of interest to the defense and security communities.
Image analysts can only look at a small portion of this data, thus finding a vehicle in

[2]Kaggle offers a no-setup, customizable, Jupyter Notebooks environment, access to free GPUs and a huge repository of community-published data & code.

the desert could take hours. Automatic techniques to monitor broad areas in important regions can be developed using computer vision algorithms and deep learning technologies applied to satellite photos. Large amounts of high-quality commercial satellite images may be gathered to label and train algorithms for detecting, classifying, and identifying infrastructures such as aircraft, vehicles, vessels, roads, and buildings.
Earth observation poses new challenges due to the diversity of environment types and conditions, as well as end-user expectations, at a time when it is becoming easier to classify cats and dogs in various situations.

2.7.1 Ensemble Of Neural Networks

The most promising machine learning approach for object detection and classification in images is CNN. It's a type of supervised method that uses a huge dataset of labeled objects to learn a generalized model of the information. At prediction time, the combination of convolutional kernels and a powerful graphics processing unit (GPU) makes CNN a powerful tool capable of human-level accuracy.
Ensemble approaches are used to solve a wide range of machine learning challenges. It is known to lower the bias of a single detector or its variance by combining different methods.
Bias and variation in prediction are well-known concerns that can be produced by using a model that is either small or large in comparison to the dataset, or by the fact that earth observation data is too heterogeneous for the model to capture the entire picture.
The majority of assembling methods increase computing time (training and/or prediction time), yet they are an effective means to advance the state-of-the-art.

2.7.2 State Of The Art of Ensembling Techniques

All ensemble learning techniques rely on a mix of variables in the following areas:

i) Transformations of the data used to train the models in the training data space.

ii) Transformations of the models trained on the dataset in model space.

iii) Prediction space: prediction transformations and combinations.

1- Training data variation-based methods: Bagging, also known as *bootstrap aggregation* is a technique for creating new datasets by extracting random sub-samples from an existing one. The variance of the training data is reduced and the accuracy of the final assembling model is improved by training the model on such random sub-samples.

Bagging can be used to train many models at the same time. The use of data augmentation, sometimes known as "*synthetic bagging*" is another technique to accomplish bagging without splitting the dataset; models are then trained on the same dataset with different forms of augmentations.

2- Model variation-based methods: Distinct CNN architectures have different strong points in remote sensing, some models may be better at distinguishing nearby objects, identifying tiny elements, or offering more exact segmentation. Combining models with different hyperparameters, such as loss and optimizer, is also fun.

A) **Snapshot Ensembling methods:** The eponymous sub-method Snapshot Ensembling (SSE) is a method that provides a large number of complimentary CNNs 'cheaply' can be coupled to increase performance. Despite the fact that the CNNs have the same architecture, the use of sophisticated "cyclical scheduling" of the learning rate allows the networks to attain various wide or narrow local optima as they train. To allow the network to converge to some local minima, the learning rate is gradually reduced. After that, the algorithm takes a snapshot of the weights before boosting the learning rate to 'jump' to another optimum. This cycle can be performed as many times as necessary to get a set of snapshots of the required size.

B) **Stochastic Weight Averaging (SWA):** is an optimization method that approximates Fast Geometric Ensembling[3] (FGE) while lowering computing costs at prediction time. A mixture of two models is used during training, The first retains a running average of the weights at each iteration, while the second explores the weight space using a cycling learning rate scheduler. Stochastic Weight Averaging gives a single model with averaged weights at inference.

C) **Hydra:** Hydra is a CNN assembling approach that involves training an initial "coarse" model, referred to as the "*body*". A snapshot of the training with average results. A set of "heads" is fine-tuned from this body using various data augmentations. This method, like snapshot ensembling, generates a series of local minima from a single neural network model.

3- Prediction strategies based on variation:

A- **Methods based on Voting:** This straightforward fusion method combines predictions by counting the detectors that agree. This vote might be based on the detector's raw pixel outputs or object wised from a vectorized shapefile.

[3]FGE is a variation of snapshot ensembling using a different scheduling scheme; rapidly cycling linear piecewise variations of the learning rate instead of the original smooth cosine one.

B- **Test Time Augmentation (TTA):** Translation, rotation, scaling, and other transformations do not make CNN inherently invariant. By applying such adjustments to the input image before feeding it to the network, this variability can be reduced and the model's performance improved. While TTA does not require any additional training, it does increase the prediction time linearly.

C- **Stacked Generalization:** Stacking uses a "meta-learner," which takes the individual detector predictions as input and trains to provide a refined output. While stacking has the ability to learn the best detector fusion, it is very particular to the input detectors and is a somewhat rigid method, as the "meta-learner" must be retrained if an input detector is changed.

D- **Dropout as a Bayesian Approximation:** During training, dropout is the process of turning off random neurons in the network. This technique, which can be thought of as an ensemble of network sub-components in the model space, aids in limiting overfitting during training. explains the concept of employing dropout not only during training but also during prediction: numerous predictions are made by deleting random neurons from the network. The average of these forecasts can then be used to improve generalization.

At each stage of the detection pipeline, assembling techniques provide a toolkit for improving the performance of a given model with a particular training dataset (training data, model, prediction).

2.8 AiTLAS

The AiTLAS toolbox (Artificial Intelligence Tool-box for Earth Observation) contains cutting-edge machine learning methods for exploratory and predictive satellite imagery analysis, as well as a repository of AI-ready Earth Observation (EO) datasets. It may be used to perform a wide range of EO activities, including land use and cover categorization, crop type prediction, semantic segmentation, and so on. The major purpose of AiTLAS is to improve the usability and acceptance of novel AI methods (and models) by EO professionals by providing easy access to standardized EO datasets and allowing benchmarking of various existing and novel AI methods customized for EO data [8].

AiTLAS was written in Python and is available under the MIT license. It was created as a library, but it may also be used as a standalone program. All of the models, datasets, and tasks can be specified using a proprietary JSON format or by programming the relevant classes to be initialized. The toolbox is powered by a number of dependencies, the most important of which being PyTorch (surveyed in section 3.3.

This serves as the foundation for the development of deep learning models and behavior.
All of AiTLAS's features are divided into five distinct modules:

1. **aitlas.base** - The AiTLAS core module, which contains the abstract definitions for everything.
2. **aitlas.models** - Contains particular deep learning model implementations.
3. **aitlas.datasets** - Lists the datasets and dataset types that AiTLAS supports.
4. **aitlas.tasks** - Contains workflows that can be executed right away.
5. **aitlas.utils** - This file contains useful utility functions that can be used both inside and outside of the toolbox.

2.8.1 AiTLAS Methods and models

A variety of models that have been found to improve prediction performance are supported by the toolkit. Its software architecture's flexibility allows users to quickly incorporate additional models, even their own custom-made models. DeepLabv3, Fast R-CNN, VGG16, and Unsupervised DeepCluster are among the models currently available in AiTLAS. The implemented models cover a wide range of EO use cases, including classification of land use and land cover, semantic segmentation, and object recognition in the context of EO, among others. Multi-class and multi-label classification is supported by the models. Multi-class models assign a single label to each image, but multi-label models assign many labels (or, in fact, land covers) to each image, which is more realistic.

2.8.2 AiTLAS Datasets

AiTLAS has a number of datasets available, divided into:

1- land-use/cover classification datasets.

2- object segmentation datasets.

AiTLAS is an open-source, cutting-edge toolbox for exploratory and predictive analysis of satellite images for a variety of EO purposes. AiTLAS features a number of distinguishing characteristics. For starters, it's modular and versatile, making it simple to configure, implement, and extend new data and models. Then there's the fact that it's generic and may be used for a wide range of activities and operations. Finally, it is simple to use. This, in addition to assisting the AI community by giving access to structured EO data, it enables and accelerates EO experts' adoption of (advanced) machine learning methods, bringing the two communities closer together.

2.9 Land Use and Land Cover Change

Although the terms '*land use*' and '*land cover*' are sometimes used interchangeably, each has a distinct meaning. Land cover is the bio-physical layer covering the earth surface, while land use represents the human utilization of the land cover. Land cover is defined by the attributes of the earth's land surface captured in the distribution of vegetation, water, desert, and ice and the immediate subsurface, including biota, soil, topography, surface, and groundwater, and it also includes those structures created solely by human activities such as mine exposures and settlement. On the other hand, land use is the intended employment of management strategy placed on the land cover by human agents, or land managers to exploit the land cover and reflects human activities such as industrial zones, residential zones, agricultural fields, grazing, logging, and mining among many others. Land use change is defined to be any physical, biological, or chemical change attributable to management, which may include conversion of grazing to cropping, change in fertilizer use, drainage improvements, installation and use of irrigation, plantations, building farm dams, pollution and land degradation, vegetation removal, changed fire regime, the spread of weeds and exotic species, and conversion to non-agricultural uses.

Land use and land cover changes result from various natural and human factors within social, economic, and political contexts. Hence, the local human activities expressing the drivers can be determined by measuring the rates and types of changes and analyzing other relevant sources of data like demographic profiles, household characteristics, and policies related to land resources administration.

To achieve this, it is crucially important to consider multiple sources of information and to acquire temporal, spatial, and other non-spatial forms of data. This is due to the fact that land use attributes are complex and the boundaries between different types of data are quite diffuse [3]. LULC change studies have been designed to improve understand of the human and biophysical forces that shape land use and land cover change. Thus, linking human behavior and social structures to biophysical attributes of the land is a fundamental aspect of LULCC research [3].

Land use and land cover play an important role in global environmental change and sustainability, including response to climate change, effects on ecosystem structure and function, species and genetic diversity, water and energy balance, and agro-ecological potential [3].

Land use and land cover mapping are one of the most important and typical applications of remote sensing data [6]. Remotely sensed data are a useful tool and have scientific value for the study of human environment interactions, especially land use and land cover changes [27].

Chapter 3

Methodology

This chapter details and explains the processes and methods used to carry out this thesis work, including how the data was collected, the models applied, and the various techniques utilized to improve the models' performance, among other things.

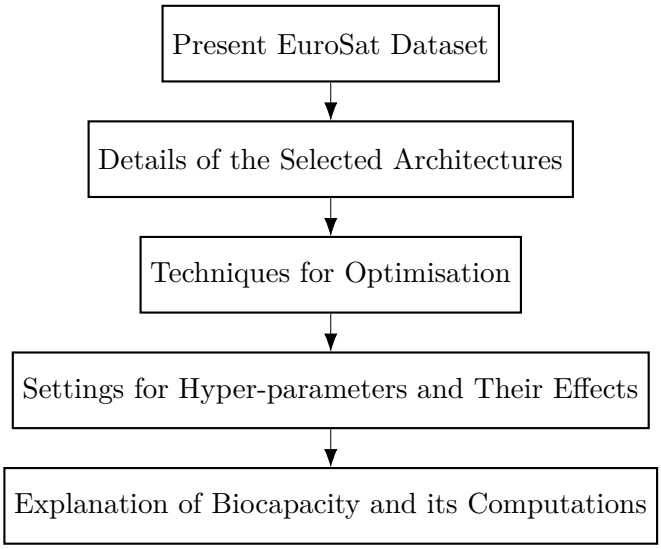


FIGURE 3.1: Flow chart of the procedures and methods used to carry out this thesis work.

3.1 EuroSat Dataset

Patrick Helber, Benjamin Bischke, Andreas Dengel, and Damian Borth in [14] proposed a novel satellite image dataset for the task of land use and land cover classification, it was made available in 2019. The proposed EuroSAT dataset consists of 27,000 labeled images of 64×64 pixels at the resolution of 10 meters per pixel, with 10 different land use and land cover classes, The ten classes can be split into two broad categories: *Natural* (forest, river, sea/lake, herbaceous vegetation) and *man made* (annual crop, residential, highway, industrial, pasture, permanent crop)., Each class has between 2,000 and 3,000 samples. The satellite images are associated with the cities covered in the European Urban Atlas. A significant difference to previous

datasets is that the presented satellite image dataset is multi-spectral covering 13 spectral bands in the visible, near infrared, and short wave infrared part of the spectrum [1]. A satellite scans the Earth to acquire images of it. Patches extracted out of these images are used for classification. In addition, the proposed dataset is georeferenced and based on openly and freely accessible Earth observation data allowing a unique range of applications. More details about dataset creation and acquisition are explained in Appendix A.2.

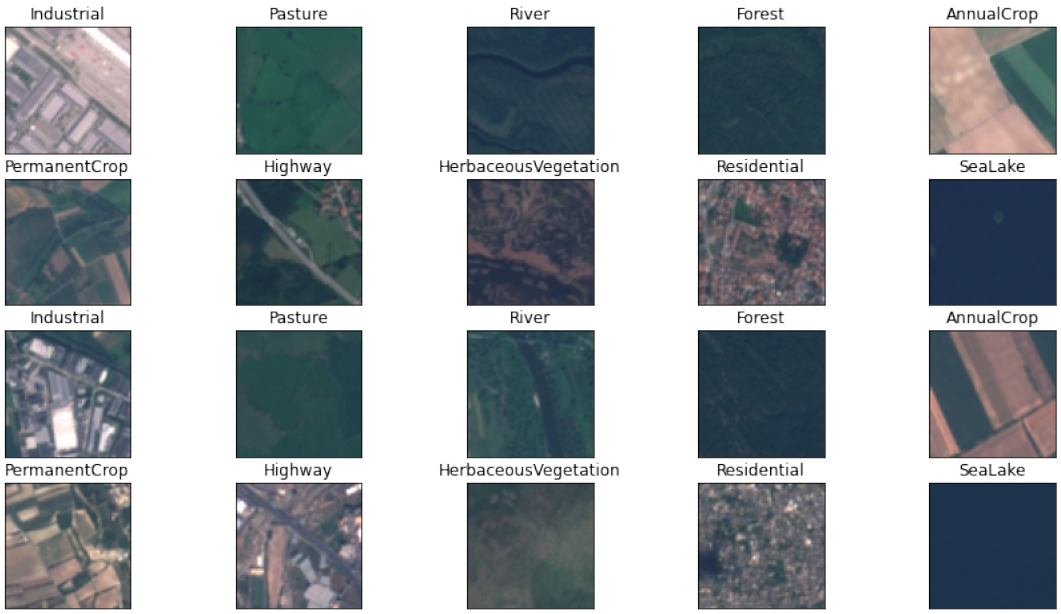


FIGURE 3.2: Overview of EuroSAT dataset classes [21].

3.2 Convolution Neural Network Architectures

3.2.1 Residual Networks

The Residual Networks (ResNet) architecture was first presented in 2015 by He et al. from Microsoft Research. By that time, this architecture won several competitions and set a new record for the ImageNet dataset classification task. The main idea behind ResNet is that its building blocks are designed to "learn residual functions with reference to the layer inputs, instead of learning unreferenced functions" [13]. Imagine an arbitrary input x to a layer of a neural network and the optimal output y. In the case of an unreferenced function, the network directly approximates a function f that generates $y : y = f(x)$. What is meant by "residual function" in this context, is that the new network now approximates $y = x + f(x)$. So instead of learning how the input has to be transformed to generate the desired output, the network learns the difference between input and optimal output. Figure 3.3 shows how this concept

[1] The labeled dataset EuroSAT is made publicly available at `https://github.com/phelber/eurosat`.

is applied to the layers of a convolutional network.
An observation from earlier models was that adding more layers to an existing network not necessarily increased the performance. The key contribution of He et al. was to develop a layer or block, that was able to easily learn the identity transformation (e.g. by setting $f(x) = 0$). Stacking these blocks on top of an existing network should ideally not deteriorate its performance, since the new blocks can learn the identity transformation, thereby retaining the base model's performance. But the newly added blocks increase the number of degrees of freedom and the network depth, therefore enhancing the capacity to learn complex features. So in case, there is a better representation of the data, the new network should be able to learn it.

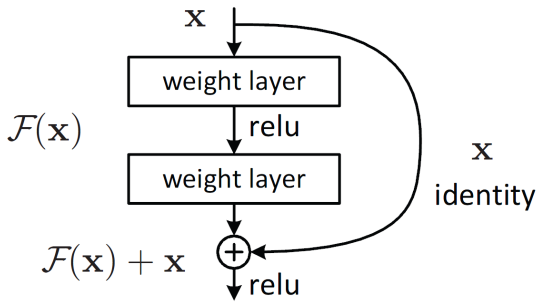


FIGURE 3.3: ResNet building block [31].

The core building block of the ResNet architecture is depicted in Figure 3.3. The block receives an input to which it applies a convolution, followed by batch normalization and the ReLU activation. This is repeated a second time and the input is added to the resulting feature map. With this architecture, it is easier to learn the identity transformation, because the optimizer just has to drive the weights in the convolutional layers to zero.
ResNets were proposed as a family of multiple deep neural networks with similar structures but different depths. ResNet introduces a structure called *residual learning unit* to alleviate the degradation of deep neural networks. This unit's structure is a feedforward network with a shortcut connection that adds new inputs into the network and generates new outputs. The main merit of this unit is that it produces better classification accuracy without increasing the complexity of the model [26].

3.2.1.1 The intuition behind Residual blocks

If the identity mapping is optimal, We can easily push the residuals to zero ($f(x) = 0$) then to fit an identity mapping (x, input = output) by a stack of non-linear layers. In simple language, it is very easy to come up with a solution like $f(x) = 0$ rather than $f(x) = x$ using a stack of non-linear CNN layers as a function. So, this function $f(x)$ is what the authors called Residual function [16].
There are two kinds of residual connections:

1. The identity shortcuts (x) can be directly used when the input and output are of the same dimensions.

$$\mathbf{y} = \mathcal{F}(x, W_i) + x. \tag{3.1}$$

2. When the dimensions change,

   A) The shortcut still performs identity mapping, with extra zero entries padded with the increased dimension.

   B) The projection shortcut is used to match the dimension (done by 1*1 conv) using the following formula:

$$\mathbf{y} = \mathcal{F}(x, W_i) + W_s x. \tag{3.2}$$

The first case adds no extra parameters, the second one adds in the form of W_s.

We have selected **ResNet152** as it achieves the **best accuracy** among ResNet family members. Table 3.1 illustrates the basic architecture of Resnet152.

TABLE 3.1: Basic architecture of Resnet152.

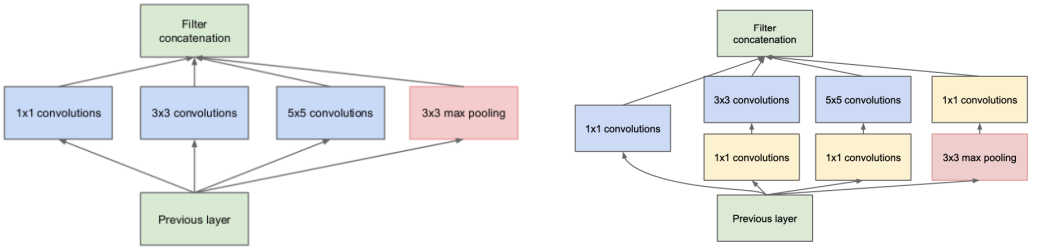
Layer name	**Output size**	**152-layer**
conv1	112×112	7×7, 64, stride 2
conv2_x	56×56	3×3 max pool, stride 2
		$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$
conv3_x	28×28	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 8$
conv4_x	14×14	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 36$
conv5_x	7×7	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$
	1×1	average pool, 1000-d fc, softmax
FlOPS		11.3×10^9

3.2.2 Google (or Inception) Network

An Inception Module is an image model block that aims to approximate an optimal local sparse structure in a CNN. Put simply, it allows us to use multiple types of filter size, instead of being restricted to a single filter size, in a single image block, which we then concatenate and pass onto the next layer.
In general, one can view the Inception model as a logical culmination while taking inspiration and guidance from the theoretical work by Arora et al in [37]. The benefits of the architecture are experimentally verified on the ILSVRC[2] 2014 classification and detection challenges, on which it significantly outperforms the current state of the art.

3.2.2.1 Architectural Details

The main idea of the Inception architecture is based on finding out how an optimal local sparse structure in a convolutional vision network can be approximated and covered by readily available dense components. Arora et al in [37] suggests a layer-by-layer construction in which one should analyze the correlation statistics of the last layer and cluster them into groups of units with high correlation. These clusters form the units of the next layer and are connected to the units in the previous layer.



(A) Inception module, naïve version (B) Inception module with dimension reductions

FIGURE 3.4: Inception module [37].

3.2.2.2 Inception module with dimension reductions

Judiciously applying dimension reductions and projections wherever the computational requirements would increase too much otherwise. This is based on the success of embeddings: even low dimensional embeddings might contain a lot of information about a relatively large image patch. However, embeddings represent information in a dense, compressed form, and compressed information is harder to model. 1×1 convolutions are used to compute reductions before the expensive 3×3 and 5×5 convolutions. Besides being used as reductions, they also include the use of rectified linear activation which makes them dual-purpose [37].

[2]The ImageNet project runs an annual software contest called The ImageNet Large Scale Visual Recognition Challenge (ILSVRC) which evaluates algorithms for object detection and image classification at a large scale.

In general, an Inception network is a network consisting of modules of the above type stacked upon each other, with occasional max-pooling layers with stride 2 to halve the resolution of the grid. For technical reasons (memory efficiency during training), it seemed beneficial to start using Inception modules only at higher layers while keeping the lower layers in traditional convolutional fashion. This is not strictly necessary, simply reflects some infrastructural inefficiencies in our current implementation. **One of the main beneficial aspects** of this architecture is that it allows for increasing the number of units at each stage significantly without an uncontrolled blow-up in computational complexity. The ubiquitous use of dimension reduction allows for shielding a large number of input filters of the last stage to the next layer, first reducing their dimension before convolving over them with a large patch size. Another practically useful aspect of this design is that it aligns with the intuition that visual information should be processed at various scales and then aggregated so that the next stage can abstract features from different scales simultaneously [37].

3.2.3 Efficient Network

When convolutional neural networks are developed, they are done so at a fixed resource cost. These networks are scaled up later to achieve better accuracies when more resources are available. A ResNet18 model can be scaled up to a ResNet200 model by adding more layers to the original model. In most situations, this scaling technique has helped provide better accuracies on most benchmarking datasets. But the conventional techniques of model scaling are very random. Some models are scaled depth-wise, and some are scaled widthwise. Some models simply take in images of a larger resolution to get better results. This technique of randomly scaling models requires manual tuning and many person-hours, often resulting in little or no improvement in performance. The authors of the Efficient Network (EfficientNet) in [38] proposed scaling up CNN models to obtain better accuracy and efficiency in a much more moral way.

EfficientNet uses a technique called *compound coefficient* to scale up models in a simple but effective manner. Instead of randomly scaling up width, depth, or resolution, compound scaling uniformly scales each dimension with a certain fixed set of scaling coefficients. Using the scaling method and AutoML, the authors of efficient developed seven models of various dimensions, which surpassed the state-of-the-art accuracy of most convolutional neural networks, and with much better efficiency.

3.2.3.1 Compound Model Scaling

For developing the method of compound scaling, the authors systematically studied the impacts that each scaling technique has on the model's performance and efficiency. They figured that while scaling single dimensions helps improve model performance,

balancing the scale in all the three dimensions *width*, *depth*, and *image resolution*, considering the variable available resources best improve the overall model performance.
The compound scaling method is based on the idea of balancing dimensions of width, depth, and resolution by scaling with a constant ratio [38]. The equations below show how it is achieved mathematically,

$$\text{Depth}\, d = \alpha^{\phi},\ \text{Width}\, w = \beta^{\phi},\ \text{Resolution}\, r = \gamma^{\phi} \tag{3.3}$$

$$\text{such that}\, \alpha \cdot \beta^2 \cdot \gamma^2 \approx 2$$

$$\alpha \geqslant 1, \beta \geqslant 1, \gamma \geqslant 1$$

The intuition for the networks is, that if the input image is bigger, then the network needs more layers to increase the receptive field and more channels to capture more fine-grained patterns on the bigger image. The compound scaling technique also helped improve the model efficiency and accuracy of previous CNN models such as MobileNet and ResNet by around 1.4% and 0.7% ImageNet accuracy, respectively, compared to other random scaling methods.

3.3 Pytorch

PyTorch is an open source machine learning framework based on the Torch library, used for applications such as computer vision and natural language processing, primarily developed by Facebook's AI Research lab (FAIR). It is free and open-source software released under the Modified BSD[3] license. Although the Python interface is more polished and the primary focus of development, PyTorch also has a C++ interface [41].
PyTorch is a Python package that provides **two high-level features**:

- Tensor computation (like NumPy[4]) with strong GPU acceleration.
- Deep neural networks built on a tape-based auto-grad system.

Usually, PyTorch is used either as a replacement for NumPy to use the power of GPUs, or as a deep learning research platform that provides maximum flexibility and speed.

[3]BSD licenses are a family of permissive free software licenses, imposing minimal restrictions on the use and distribution of covered software.
[4]NumPy is a Python library that is used for working with arrays.

3.3.1 PyTorch Tensors

PyTorch defines a class called Tensor `torch.Tensor` to store and operate on homogeneous multidimensional rectangular arrays of numbers. PyTorch Tensors are similar to NumPy Arrays, but can also be operated on a CUDA-capable Nvidia GPU. PyTorch supports various sub-types of Tensors [41].

3.4 Google Collaboratory

Also called Google Colab It is a product of Google Research and is used to write and run Python and other languages from our browser. Colab is a hosted Jupyter notebook[5] , installed and configured so that we don't have to do anything on our computer, but just work from the browser. Unlike Jupyter notebook Colab runs entirely in the cloud. Colab at the moment only supports Python code and its ecosystem of third-party tools. In Addition, It provides us with free GPU and Tensor Processing Units (TPU) services.

3.5 Adam Optimizer

The optimization algorithm is generally understood to be gradient descent and its variants. In the present work, we use the Adam optimization algorithm. A simple implementation of gradient descent might not work very well in a deep network, since it faces issues with navigating around local optima. This is rectified by introducing another parameter, the momentum, which helps to aid the gradient descent (GD) update process as necessary to reach an optimal point. Adam optimizer is one such implementation, short for Adaptive Moment Estimation,[18] where adaptive learning rates are calculated for each parameter. In the case of a noisy gradient parameter, as occurs near a local optima, the ω value is updated using an estimation of the first and second moments of the gradients [18]. These values correspond to the mean and variance of the gradients respectively. These are shown below:
We calculate the gradients of the loss L w.r.t. the weights ω at time step t,

$$g_t = \nabla_\omega L_t(\omega_{t-1}) \tag{3.4}$$

Then we update biased first and second moment estimate,

$$m = \beta_1 m + (1 - \beta_1) g_t \tag{3.5}$$

$$v = \beta_2 v + (1 - \beta_2) {g_t}^2 \tag{3.6}$$

[5] An interactive computing environment where users can experiment with and share code.

$$\omega_t = \omega_{t-1} - \epsilon \frac{m}{\sqrt{v}} \tag{3.7}$$

where β_1 and β_2 are constants used to specify the decay rate, m is the mean, and v, the variance of the gradients. ϵ is suggested to be set at 10^{-8} [18].

3.6 Hyper-Parameter Settings

The aforementioned algorithms and models require special hyper-parameters to be set by the user. Setting these hyper-parameters will have a non-negligible effect on an algorithm's performance in training the network. These effects must be accounted for when performing experiments with these algorithms. In this section, we will provide an overview of the most common hyper-parameters that we utilized while designing our model. Note that these are the most well-known and most widely used parameters.

3.6.1 Epochs

Firstly and maybe most importantly, the number of iterations performed by the descent algorithm has to be defined. Usually, we iterate through the dataset as long as the "stopping criterion is not met". In practice, this criterion is defined by specifying the number of epochs for which to perform the loop. We do this because it usually can't be known when the minimum is reached. In other words, we usually do not know what the optimum value $f_{optimum}$ for the loss function f will be. Thus, we cannot define a condition that depends on such a value (e.g. "while $f > f_{optimum}$ "). The preferred solution is therefore to perform a predefined number of epochs and take the optimal state of the neural network, found within this period. This optimal state is the released network.
A single epoch is defined as a complete run through the training data. An iteration is a single update step of the iterative optimizer. The number of iterations performed by the optimizer n_{iter} equals the number of epochs n epochs times the number of batches n batches in the data set.

3.6.2 Batch Size

The most important aspect of this choice is the advantage we gain in the computational power needed. Calculating the loss and gradient for smaller batches uses less GPU memory. This might be an essential modification for us to even be able to perform the necessary calculations since data sets tend to be very large containing at least many tens of thousands of samples, possibly even millions.

However, we also have to acknowledge that our choice of the batch size will directly affect our results. Keskar et al. in [17] argue that large batch sizes converge to sharp

minima. Rather than giving a theoretical explanation, they show their results experimentally. Figure 3.5 illustrates why sharp minima are problematic. The likelihood of a sharp minimum, evaluated on the training data, coinciding with a near-minimum point of the loss function, evaluated on the test data, is less than when we are at a flat minimum. Training with large batch sizes might yield trained networks that perform just as well on the training data as those that were trained using smaller batches. Nonetheless, the solutions to the optimization problem in Fig. 2.6 do not generalize well on new data. Normally, an optimal batch size lies in the range of 32 and 512 samples.

Note. Due to a large number of spectral bands in satellite images, We had to keep the batch size small as we would frequently run out of memory. We finally had to settle on a batch size of **64** images.

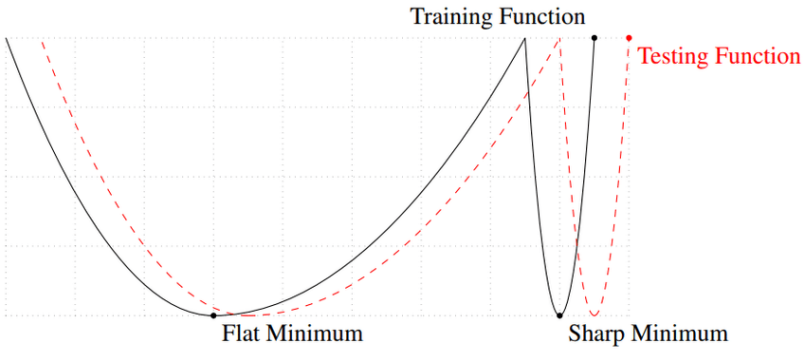


FIGURE 3.5: Sketch of both a sharp and a flat minimum found within the loss function. The y-axis shows the values of f , the loss function, and the x-axis indicates the value of the parameter of the model [42].

3.6.3 Learning Rate Scheduling

The learning rate α is considered one of the most important hyperparameters for training deep neural networks, but choosing it can be quite hard. Rather than simply using a fixed learning rate, it is common to use a learning rate scheduler. The simplest and perhaps most used adaptation of learning rate during training are techniques that reduce the learning rate over time, by adjusting the learning rate α between epochs or iterations as the training progresses. The value of α depends on the iteration $j \in \{1, ..., n_{iter}\}$.

One way a scheduler can be constructed is by introducing Cyclic Learning Rate (CLR) scheme [35], This scheme might produce incredible results for the convergence of the algorithms.

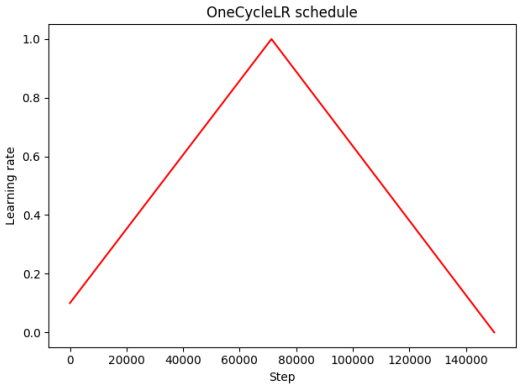


FIGURE 3.6: Learning Rate Scheduling over 1 cycle.

A CLR scheme usually starts with an initially small value α min (minimum learning rate) and continuously increases the learning rate to a larger value α max (maximum learning rate), after which it the decreases the learning rate back down to α min. The update of the learning rate usually happens after each iteration. The ascent and descent of the learning rate parameter itself are usually linear. Nonetheless, alternatives can be thought of and have already been implemented and tested successfully. This cycle can be performed arbitrarily often during the training. If exactly one complete cycle is performed over the predefined number of epochs, we call that a 1-*cycle*. The learning rate in a 1-*cycle* can be defined as:

$$\alpha^{(i)} = \begin{cases} \alpha_{min} + \frac{2(j-1)(\alpha_{max}-\alpha_{min})}{n_{iter}}, & j \in \{1, ..., \lceil \frac{n_{iter}}{2} \rceil\} \\ \\ \alpha_{max} + \frac{2(j-\frac{n_{iter}}{2})(\alpha_{min}-\alpha_{max})}{n_{iter}}, & j \in \{\lceil \frac{n_{iter}}{2} \rceil + 1, ..., n_{iter}\} \end{cases} \tag{3.8}$$

yielding the special points

$$\alpha^{(1)} = \alpha_{min} \tag{3.9}$$

$$\alpha^{(\frac{n_{iter}}{2})} = \alpha_{max}, \text{ if } n_{iter} \text{ uneven} \tag{3.10}$$

$$\alpha^{(n_{iter})} = \alpha_{min} \tag{3.11}$$

3.7 Biocapacity

The biocapacity or biological capacity of an ecosystem is an estimate of its production of certain biological materials such as natural resources, and its absorption and filtering of other materials such as carbon dioxide from the atmosphere. Biocapacity is expressed in terms of global hectares per person, this is dependent on the human population. A global hectare is an adjusted unit that represents the average biological productivity of all productive hectares on Earth in a given year (because not all hectares produce the same amount of ecosystem services). Biocapacity is calculated from United Nations population and land use data, and may be reported at various regional levels, such as a city, a country, or the world as a whole. For example, there were 12.2 billion hectares of biologically productive land and water areas on this planet in 2016. Dividing by the number of people alive in that year, 7.4 billion, gives a biocapacity of 1.6 global hectares per person. These 1.6 global hectares include the

areas for wild species that compete with people for space.
Biocapacity is used together with Ecological Footprint as a method of measuring human impact on the environment. Biocapacity and Ecological Footprint are tools created by the Global Footprint Network[6] (GFN) and used in sustainability studies around the world [12].

3.7.1 Applications of Biocapacity

A rise in the world population can lead to a reduction in biocapacity. This is typically due to the fact that the Earth's resources must be shared; as a result, there is little available to meet the growing demand of the growing population. This problem can currently be solved through outsourcing. However, resources will run out as a result of rising demand, and a collapse of an ecosystem could ensue as a result of such behaviors. A biocapacity deficit is detected when the population's ecological footprint exceeds its biocapacity.
The term "global biocapacity" is occasionally used to characterize an ecosystem's complete capacity to support numerous continuous activities and changes. A 'biocapacity deficit' occurs when a population's ecological footprint exceeds the biocapacity of the environment in which it lives. Overuse of one's own ecosystems ("overshoot"), net imports, or use of the global commons are three sources of such a deficit.
According to the latest data from the GFN, humans used an equivalence of 1.7 Earths in 2016. Carbon dioxide emissions from fossil fuel combustion are the primary cause of global ecological overshoot. Additional pressures like as greenhouse gas emissions, climate change and ocean acidification can exacerbate the issue. In terms of biocapacity, 1.7 Earths suggests that renewable resources are being depleted since they are being utilized faster than they can be replenished. As a result, the resources humanity utilizes in a year will take one year and eight months to renew, including absorbing all of the waste we make. So, instead of consuming a year's worth of resources every year, we consume resources that should last us one year and eight months every year. In addition, if the situation worsens, places will be designated as ecological reserves in order to protect their ecosystems. Agricultural land, forest resources, and rangeland are all examples of decreasing resources. When biocapacity is compared to ecological footprint, it can be determined whether a population, region, country, or portion of the planet is living within its means. As a result, the Ecological Footprint Analysis is the study of biocapacity and ecological footprint.
Furthermore, environmentalists have devised ecological footprint calculators for a single person or a group of people to see if they are consuming more than their population allows. As a result, biocapacity data will be compared to their ecological footprint to

[6]Global Footprint Network is a research organization that is changing how the world manages its natural resources and respond to climate change.

see how much they can contribute to or detract from long-term development.
In general, biocapacity refers to the number of resources accessible to humans at a certain point in time to a specific population (supply), as opposed to ecological footprint, which refers to a regional ecosystem's environmental demand. Biocapacity can predict human consequences on the planet. Biocapacity will be able to estimate and maybe assess the effects on ecosystems based on collected human consumption findings by evaluating land productivity (i.e. the resources available for human consumption). Multiplying the actual physical area by the yield factor with the appropriate equivalency factor yields the biocapacity of an area. The most common unit of measurement for biocapacity is global hectares. Biocapacity can be used to determine the Earth's carrying capacity since global hectares can translate human consumption like food and water into a measurement.

3.7.2 Biocapacity Calculation

National Footprint and Biocapacity Accounts (NFAs) provide the core data required for all Ecological Footprint analyses worldwide. They are now being maintained by York University's Ecological Footprint Initiative for the Footprint Data Foundation (FoDaFo). FoDaFo was established in 2019 by York University and GFN to be the stewards of those NFAs.
The Accounts measure the ecological resource use and resource capacity of nations over time. Based on approximately 15,000 data points per country per year, the Accounts calculate the Footprints of more than 200 countries, territories, and regions from 1961 to the present.
The calculations in the NFAs are based on United Nations or UN affiliated data sets, including those published by the Food and Agriculture Organization, United Nations Commodity Trade Statistics Database, and the UN Statistics Division, as well as the International Energy Agency. Supplementary data sources include studies in peer-reviewed science journals and thematic collections. Of the countries, territories, and regions analyzed in the Accounts, 150 had populations over one million and typically have more complete and reliable data sets. For most of those, GFN is able to provide time series of both Ecological Footprint and biocapacity.
The Ecological Footprint is derived by tracking how much biologically productive area it takes to provide for all the competing demands of people. These demands include space for food growing, fiber production, timber regeneration, absorption of carbon dioxide emissions from fossil fuel burning, and accommodating built infrastructure. A country's consumption is calculated by adding imports to and subtracting exports from its national production.
The Ecological Footprint uses yields of primary products (from cropland, forest, grazing land and fisheries) to calculate the area necessary to support a given activity.

Biocapacity is measured by calculating the amount of biologically productive land and sea area available to provide the resources a population consumes and to absorb its wastes, given current technology and management practices. To make biocapacity comparable across space and time, areas are adjusted proportionally to their biological productivity. These adjusted areas are expressed in "global hectares". Countries differ in the productivity of their ecosystems, and this is reflected in the Accounts.
Results from this analysis shed light on a country's ecological impact. A country has an ecological reserve if its Footprint is smaller than its biocapacity; otherwise, it is operating with an ecological deficit. The former is often referred to as ecological creditors and the latter as ecological debtors.
Today, most countries, and the world as a whole, are running ecological deficits. In fact, today over 85% of the world population lives in countries with an ecological deficit. The world's ecological deficit is referred to as a global ecological overshoot.

Chapter 4

Results and Discussion

4.1 Training Of 13-band version Of EuroSat

As mentioned in section 3.2 EuroSat dataset has a 13-channel version which we decided to train and benchmark the best performing models of three preselected state-of-the-art CNN architectures the selected GoogleNet and EfficientNet. The model code of theses architectures is written from scratch using PyTorch framework and the code is executed, trained, and tested on Google Colab. The parameters used to train both models are the same so we no-bias can occur and we know the best performing models on 13-band satellite images, The parameters that are fed to the model are given below.

Table 4.1: Hyper-parameters settings for the models

Optimizer	Adam
Learning Rate	Scheduled using CLR
Epochs	20
Batch Size	64

4.1.1 Learning Rate Scheduler Creation

For the scheduling of the learning rate, we used `OneCycleLR` class from `torch.optim.lr_scheduler` module. which sets the learning rate of each parameter group according to the 1cycle learning rate policy. The 1cycle policy as discussed in Section 3.6.3 anneals the learning rate from an initial learning rate to some maximum learning rate and then from that maximum learning rate to some minimum learning rate much lower than the initial learning rate.

Note. The scheduler uses the maximum learning rate from the graph. The number of epochs to train for and the steps per epoch must be entered. It is common practice to use the batch size as the steps per epoch which is what we did.

The values outputted by `OneCycleLR` are plotted in the Figure below. As it is obvious from the graph the maximum value is **0.01**.

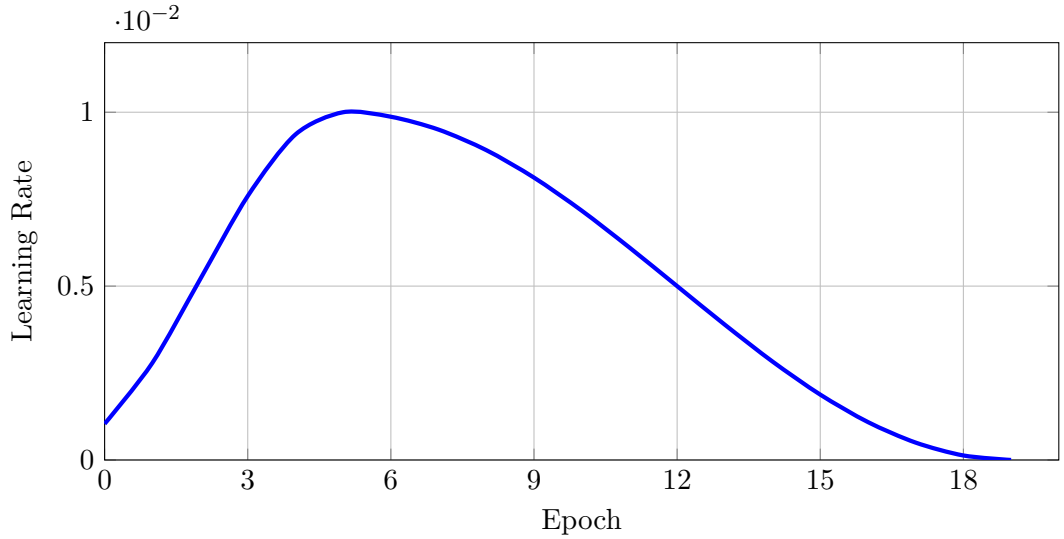


FIGURE 4.1: Learning rate values outputted by `OneCycleLR` class during training.

4.1.2 Architectures Performance & Evaluation

After the training process of each model, we evaluated these chosen architectures with different metrics explained in Section 4.1.2.1. To evaluate the performance of the models we have split our dataset into three parts:

(i) **Training Set:** Used for learning.

(ii) **Validation Set:** Used to tune the parameters of a classifier.

(iii) **Test Set:** Used only to assess the performance of a fully-specified classifier against unseen data.

Since our dataset contains 27,000 images we divided the dataset as shown below. We tried different splits but this one gave us the best result so far:

Set	*Training*	*Validation*	*Test*
Size	15,000	6,000	6,000

4.1.2.1 Metrics

In order to measure the performance of a network, we will define different metrics. Depending on the task, a suitable metric has to be chosen. When we use a classifier to differentiate between C classes, the network returns a normalized vector $y_p \in \mathbb{R}_C$. Each entry of the vector corresponds to the probability of the respective class and will be denoted with $y_{p,c}$. When a network for image segmentation is used, it returns an entire probability map or a matrix $Y_p \in \mathbb{R}^{H \times W \times C}$, respectively. This matrix is H rows high, W columns wide, and has C channels in depth. For the calculation of the

metrics, this matrix is reshaped by concatenating its rows ($\mathbb{R}^{H\times W\times C} \rightarrow \mathbb{R}^{H\cdot W\times C}$). Essentially, it now holds C vectors with $H \cdot W$ entries, which will be denoted with $\mathbf{y}_{p,c}$ (bold means vector). These vectors will then be compared to their ground truth counterparts $\mathbf{y}_t$.

Accuracy, Precision, and Recall

The accuracy, or more specifically the Rand Index, measures the agreement of prediction and ground truth. One can apply it to predicted masks or to sets of objects. It is zero if the prediction misses every ground truth element and one if both match perfectly.

$$\text{Accuracy} = \frac{t_p + t_n}{t_p + t_n + f_p + f_n} \tag{4.1}$$

t_p : True positives, t_n : True negatives, f_p : False positives, f_n : False negatives

The precision is the ratio between the true positives and all the positives. For example in our problem it tells how many non-industrial regions were falsely marked as industrial regions. High precision means, that almost every identified region is an industrial region. In the context of remote sensing, the precision metric is also called *user's accuracy*.

$$\text{Precision} = \frac{t_p}{t_p + f_p} \tag{4.2}$$

In contrast, the recall measures how many instances were missed during prediction. A high recall means, that every instance was found, regardless of how many regions were falsely marked positive. The recall is also known as *producer's accuracy*.

$$\text{Recall} = \frac{t_p}{t_p + f_n} \tag{4.3}$$

The F_1-Score is a "trade-off" between the two and is defined as their harmonic mean:

$$F_1 = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{4.4}$$

Macro Average (Unweighted Mean)

The macro-average precision, recall, and F1 score is calculated as the arithmetic means of individual classes' precision and recall scores. This method treats all classes

equally regardless of their support values, Macro-average method can be used when you want to know how the system performs overall across the sets of data.

4.1.2.2 Models Evaluation & Results

In all the models we got stunning results with +95% test accuracy in each model.

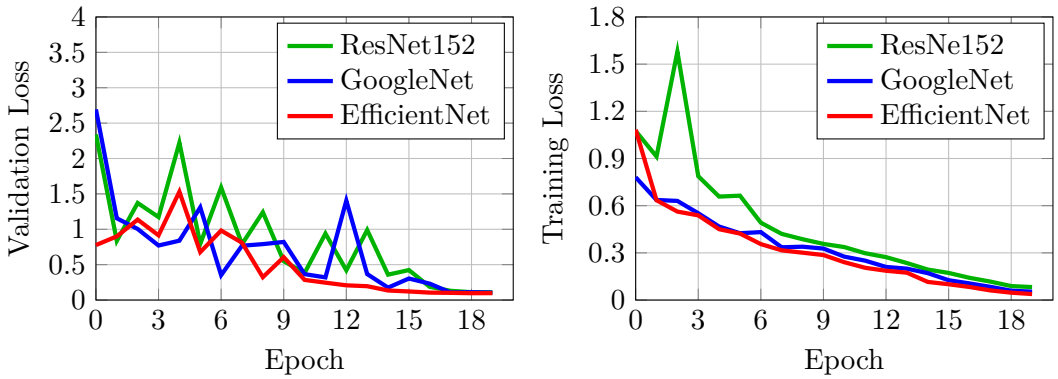
TABLE 4.2: Classification Accuracy (%) of the best performing classifiers.

Architecture	**Train Acc. %**	**Validation Acc. %**	**Test Acc. %**
ResNet152	98.67	96.35	96.55
GoogleNet	99.08	96.97	97.45
EfficientNet	99.84	97.04	97.81

Even if it's intuitive, accuracy can be misleading and it should be used only for a balanced dataset. So we calculated Recall, Precision, and F1 Scores[1] to see if the findings we got were reliable. The results are displayed in the table below.

TABLE 4.3: Precision, Recall, and F1 scores obtained by The selected models.

Architecture	**Marco Recall**	**Marco Precision**	**Marco F1 Score**
ResNet152	0.9633	0.9624	0.9604
GoogleNet	0.9752	0.9734	0.9702
EfficientNet	0.9763	0.9749	0.9738



(A) Validation Loss.

(B) Training Loss.

FIGURE 4.2: Training and Validation losses over 20 epochs for the three selected networks.

[1]The implementation and evaluation of the models is published at `https://github.com/AsimZz/Univeristy-of-Khartoum-Graduation-Project/blob/master/cnn_models_and_training.ipynb`

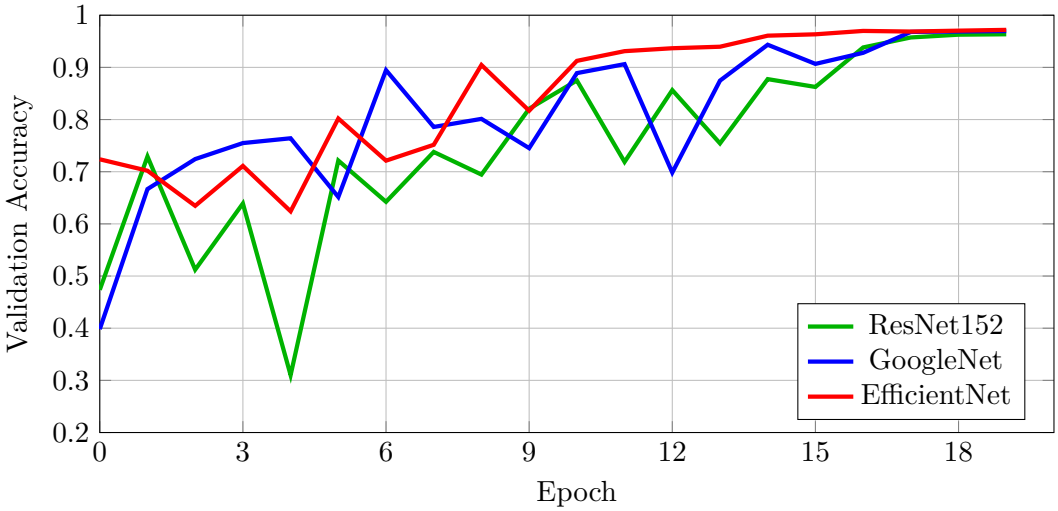


FIGURE 4.3: Validation accuracy for the selected networks throughout the training process.

All of the models achieved pretty close scores and produced **outstanding** results, although the EfficientNet model learns faster than the other two and barely overshoots throughout the training phase.

4.2 Biocapacity Estimation

4.2.1 Evaluating EfficientNet Against Tuti Island and Port Sudan

Because EuroSat dataset contains only images from European regions, we were curious to see how the model would generalize and perform in other parts of the world. For this task specifically, we want to estimate the biocapacity of Both Tuti Island and Port Sudan and compare their results.

Because feature importance in multi-spectral images is equal for each band, and biocapacity computation requires significant importance in only RGB channels, so we extracted the Red, Green, and Blue bands from each image in the dataset and trained the new images in our winner model (EfficientNet) by just using RGB channels. After training, we obtained a loss of **0.33** and a test accuracy of **97.7**%, which is pretty much identical to what we gained in the multi-spectral version of the model, but the model is now more biased toward RGB colors.

(A) Tuti Island Map. (B) Port Sudan Map.

FIGURE 4.4: A snapshot of Tuti Island and Port Sudan Maps which are obtained from Google Earth.

4.2.2 Calculation of biocapacity for Single Land Use Type

$$BC = A \times YF \times EQF \tag{4.5}$$

Where:

$BC \equiv$ biocapacity of a given land use type, measured in global hectares 'gha'

$A \equiv$ Area of a given land use type within a country, measured in National-average hectares (for a given land use type) 'nha'

$YF \equiv$ Yield factor of a given land use type within a country, 'wha/nha'

$EQF \equiv$ Equivalence factor for given land use type, 'gha/wha'

4.2.3 Biocapacity Computations

Using the predicted label distributions from our EfficientNet model, we can calculate the total biocapacity of both regions, but since the GFN classifies land uses into 6 categories, we had to allocate our 10 classes to the 6 categories, the details of this process are shown in Table 4.4.

Since our model requires the land image to be in a shape of 64×64, we have sliced the island map to 2500 tiles and then we resized each tile to 64×64 images, and then we fed the sliced images to our model. The predicted classes from Tuti island and Port Sudan is given in Tables 4.5a and 4.5b respectively.

TABLE 4.4: Mapping of EuroSat classes to Global Footprint Network classes.

GFN Land Category	**Equivalent Land Types**
Crop Land	AnnualCrop + PermanentCrop
Forest Land	Forest
Grazing Land	Pasture + HerbaceousVegetation
Marine Fishing Grounds	River
Infrastructure	Industrial + Highway + Residential
Fishing Grounds	SeaLake

Table 4.5: Tuti Island and Port Sudan Classes Distribution.

(a) Tuti Island Classes Distribution.

GFN Land Category	Distribution
Crop Land	999
Forest Land	0
Grazing Land	17
Marine Fishing Grounds	7
Infrastructure	1476
Fishing Grounds	0

(b) Port Sudan Classes Distribution.

GFN Land Category	Distribution
Crop Land	578
Forest Land	4
Grazing Land	51
Marine Fishing Grounds	16
Infrastructure	1581
Fishing Grounds	269

We then use Eq. 4.5 to calculate Biocapacity[2]. The Equivalence factors are used to convert a specific land type into a universal unit of biologically productive area, a global hectare. Yield Factors account for differences between countries and years in productivity of a given land type. Equivalence Factors & Yield Factors of Sudan is written in Table 4.6.

Table 4.6: Values for Equivalence Factors and Sudan's Yield Factors.

Land Use Type	Equivalence Factor	Sudan Yield Factor
Crop Land	2.5	0.437
Forest Land	1.262	0.439
Grazing Land	0.453	1
Marine Fishing Grounds	0.364	1.47
Infrastructure	2.5	0.437
Fishing Grounds	0.364	1

The results of our analysis show that the Tuti Island has a biocapacity of **1.086** *gha* while Port Sudan has a biocapacity of **0.9965** *gha*. Thus, we can conclude that the land represented by Tuti Island is more biologically productive than the land in Port Sudan.

[2]The code developed to compute the biocapacity can be found at `https://github.com/AsimZz/Univeristy-of-Khartoum-Graduation-Project/blob/master/RGB_Biocapacity_.ipynb`

Chapter 5

Conclusion and Future Work

5.1 Conclusion

A major motivation for undertaking this work was to gain an understanding to design, implement and evaluate a deep learning pipeline with the focus on satellite image data, which was defined as a LULC problem. In this work, we have compared methods for optimal utilisation of multi-spectral information in satellite imagery. We provided benchmarks for EuroSat dataset with its spectral bands using three state-of-the-art deep CNNs. We are able to optimize reading the GeoTiff image format (multispectral images format) which consume a lot of memory during the preprocessing phase, which is a challenging problem when combining computer vision with satellite imagery. The models chosen were built from scratch using PyTorch library. Our findings demonstrate a fascinating step forward in remote sensing because we were able to achieve high test set accuracy, precision, and recall with the proposed architectures, allowing us to classify and predict LULC changes. We have explored and test the best model against two local regions in Sudan (Tuti Island and Port Sudan). Our experiments revealed that the model correctly predicted the majority of most of the classes but failed to recognise others, due to the variations between the regions where the model was trained (Europe) and the regions where it was tested (Africa). Finally, we used the predictions results to estimate biocapacity values for those two regions that sound reasonable in terms of to Sudan's overall biocapacity.
Overall, our work shows a high-performing classification model with a real world application that can aid researchers and policy makers in better understanding environmental impacts and changes in natural landscapes over time.

5.2 Future Work

Helping researchers and policymakers going forward, we have identified several areas where we believe additional research would be particularly beneficial:

- Create a novel dataset in Africa based on remotely sensed satellite images . In addition, using this new dataset, train and validate a deep learning model by comparing it to more Sudanese states and cities.
- Examine model performance under various atmospheric conditions, cloud cover, and so on.
- Include a study of a region's changing footprint over time, as well as the calculating ecological footprints to estimate biocapacity defects/reserved for specific region.
- Build an open-source satellite imagery data framework that accepts any satellite image dimensions, preprocesses it, and applies deep learning techniques based on user's intent.
- Apply different learning techniques to land-use classification tasks such as transfer learning methods that have been shown to improve predictions results.
- Test more datasets with even higher resolutions, which are computationally more demanding, particularly in terms of GPU memory.

References

[1] *A Comprehensive Guide to Convolutional Neural Networks*. https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53. Accessed: 2022-03-25.

[2] *A Practical Guide to ReLU*. https://medium.com/@danqing/a-practical-guide-to-relu-b83ca804f1f7. Accessed: 2022-03-28.

[3] X. Baulies and G. Szejwach. "LUCC data requirements workshop : survey of needs, gaps and priorities on data for land-use/land-cover change research". In: (Nov. 1997), pp. 11–14.

[4] Mark Bear, Barry Connors, and Michael Paradiso. *Neuroscience: Exploring the brain: Fourth edition*. Jan. 2015, pp. 1–975.

[5] Patchareeya Chaikaew. "Land Use Change Monitoring and Modelling using GIS and Remote Sensing Data for Watershed Scale in Thailand". In: (2019).

[6] Nektarios Chrysoulakis et al. "Combining satellite and socioeconomic data for Land Use Models estimation". In: 2004.

[7] *Deep Artificial Neural Networks Model*. https://www.pngwing.com/en/free-png-pbana. Accessed: 2022-03-22.

[8] Ivica Dimitrovski et al. "AiTLAS: Artificial Intelligence Toolbox for Earth Observation". In: (2022).

[9] *Electromagnetic Spectrum*. https://www.japanistry.com/electromagnetic-spectrum/. Accessed: 2022-03-22.

[10] Kunihiko Fukushima. "Cognitron: A self-organizing multilayered neural network". In: 3-4 (1975), pp. 121–136.

[11] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016.

[12] Laurel Hanscom. *Working Guidebook to the National Footprint and Biocapacity*. Tech. rep. Pages: 73. May 2019.

[13] Kaiming He et al. "Deep Residual Learning for Image Recognition". In: (2015).

[14] Patrick Helber et al. "EuroSAT: A Novel Dataset and Deep Learning Benchmark for Land Use and Land Cover Classification". In: (2017).

[15] Rob Hierons. “Machine learning”. In: 3 (1997), pp. 191–193.

[16] Prakash Jay. *Understanding and Implementing Architectures of ResNet*. 2018. (Visited on 02/07/2018).

[17] Nitish Shirish Keskar et al. “On Large-Batch Training for Deep Learning: Generalization Gap and Sharp Minima”. In: (2016).

[18] Diederik P. Kingma and Jimmy Ba. “Adam: A Method for Stochastic Optimization”. In: *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*. Ed. by Yoshua Bengio and Yann LeCun. 2015.

[19] Kees Klein Goldewijk et al. “The HYDE 3.1 spatially explicit database of human-induced global land-use change over the past 12,000 years”. In: *Global Ecology and Biogeography* 1 (2011), pp. 73–86.

[20] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. “ImageNet Classification with Deep Convolutional Neural Networks”. In: *Advances in Neural Information Processing Systems*. Ed. by F. Pereira et al. Vol. 25. 2012, pp. 1097–1105.

[21] *Land Cover Classification Using TensorFlow in Python*. `https://towardsdatascience.com/land-cover-classification-using-tensorflow-in-python-791036eaa373`. Accessed: 2022-04-08.

[22] Yann LeCun, Y. Bengio, and Geoffrey Hinton. “Deep Learning”. In: (May 2015), p. 436.

[23] *LIDAR (Light Detection And Ranging)*. `https://www.physics-and-radio-electronics.com/blog/lidar-light-detection-ranging/`. Accessed: 2022-03-22.

[24] Gary Marcus. “Deep Learning: A Critical Appraisal”. In: (2018).

[25] *Neuron Nervous system Cell*. `https://www.pngwing.com/en/free-png-pkjmd`. Accessed: 2022-03-22.

[26] Long Nguyen et al. “Deep CNNs for microscopic image classification by exploiting transfer learning and feature concatenation”. In: May 2018, pp. 1–5.

[27] Samuel Nii and Samuel Codjoe. “Integrating Remote Sensing, GIS, Census, and Socioeconomic Data in Studying the Population–Land Use/Cover Nexus in Ghana: A Literature Update”. In: *Africa Development* (Jan. 2007), pp. 197–212.

[28] *NIPS’12: Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*. 2012.

[29] Nancy Noella. “Diagnosis Of Alzheimer’s And Parkinson’s Disease Using Artificial Neural Network”. In: *International Journal of Scientific Technology Research* 9 (Mar. 2020), p. 3659.

[30] *Remote Sensing: An Overview*. `https://earthdata.nasa.gov/learn/backgrounders/remote-sensing`. Accessed: 2022-04-08.

[31] *Residual CNNs for Image Classification Tasks*. `https://neurohive.io/en/popular-networks/resnet/`. Accessed: 2022-04-09.

[32] *Sentinel 2A*. `https://www.n2yo.com/satellite/?s=40697`. Accessed: 2022-05-20.

[33] *Sentinel-2*. `https://en.wikipedia.org/wiki/Sentinel-2`. Accessed: 2022-05-20.

[34] *Sigmoid function*. `https://en.wikipedia.org/wiki/Sigmoid_function`. Accessed: 2022-03-28.

[35] Leslie N. Smith. “Cyclical Learning Rates for Training Neural Networks”. In: *2017 IEEE Winter Conference on Applications of Computer Vision (WACV)*. 2017, pp. 464–472.

[36] Marcelo C. C. Stabile. “Deconstructing the complexity of land use and cover classification and land change modelling”. In: 2011.

[37] Christian Szegedy et al. “Going Deeper with Convolutions”. In: (2014).

[38] Mingxing Tan and Quoc V. Le. “EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks”. In: (2019).

[39] *Two-line element set*. `https://en.wikipedia.org/wiki/Two-line_element_set`. Accessed: 2022-05-20.

[40] Jan Weinzettel et al. “Affluence drives the global displacement of land use”. In: *Global Environmental Change* 2 (2013), pp. 433–438.

[41] Wikipedia contributors. *PyTorch — Wikipedia, The Free Encyclopedia*. 2022. URL: `https://en.wikipedia.org/w/index.php?title=PyTorch&oldid=1075942489` (visited on 03/08/2022).

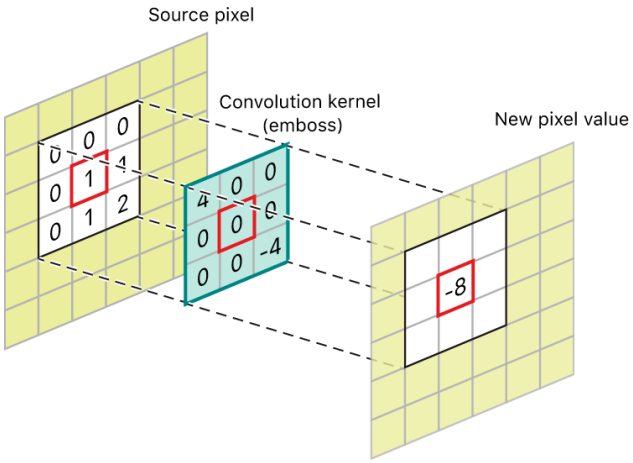
[42] Xubo Yue, Maher Nouiehed, and Raed Kontar. “SALR: Sharpness-aware Learning Rates for Improved Generalization”. In: (Nov. 2020).

Appendix A

Convolution Operation

A.1 Convolutional Layers

The mathematical operation of convolution is the central building block of CNNs, hence giving them their name. In the case of discrete raster data, such as images, the convolution receives two inputs: the image I and a kernel K, with which the image is convolved. Both image and kernel are in this case three dimensional arrays with I being H pixels high, W pixels wide, and having C channels in depth. The kernel is a matrix with dimensions $H_k \times W_k \times C$, so image and kernel share the number of channels. In the lateral direction (height and width), the kernel is much smaller than the image, being usually 3×3 or 5×5 pixels in size. The output of the convolution is a new array with dimensions $(H - H_k + 1)(W - W_k + 1)$,so the depth dimension is "consumed" by the convolution and the lateral dimensions are reduced as well. The convolution operation can be seen as sliding the kernel over the input image, computing the element-wise product at each position i, j. Following Eq. A.1 which gives the mathematical definition of the convolution, as used in deep learning.



$$(0 \cdot 4) + (0 \cdot 0) + (0 \cdot 0) + (0 \cdot 0) + (1 \cdot 0) + (1 \cdot 0) + (0 \cdot 0) + (1 \cdot 0) + (2 \cdot -4) = \textbf{-8}$$

FIGURE A.1: Convolution process computations.

$$(I * K)_{i,j} = \sum_{l=0}^{H_k-1} \sum_{m=0}^{W_k-1} \sum_{n=1}^{C} I_{i+l,j+m,n} K_{l,m,n} \tag{A.1}$$

I: Input image, K: Convolution kernel, i, j: Indices of the convolution output array
l, m, n: Spatial and channel indices

Figure A.1 demonstrates how the convolution is computed for a grayscale image. In order to achieve an output with the same lateral extent as the input, the input is padded, which means that the border values are mirrored with an appropriate width (here one).

A.2 Padding and Stride

Padding and *stride* are means of controlling the output size of a convolution operation and how it extracts information from its input. When padding an image or feature map, additional values are inserted around its border. Padding can be done with a constant value (usually zero) or a reflection of the image border. The larger the padding width p, the larger the output after the convolution. Padding can therefore be used to retain a feature map's lateral size during the convolution operation.
The kernel can be slid over the input with a certain step width, the *stride*. A larger stride s results in a smaller output image; a stride of two e.g. halves the output size. Above we assumed a stride of one, so no aggressive reduction of information took place. Figure A.2 illustrates the combination of padding and stride:

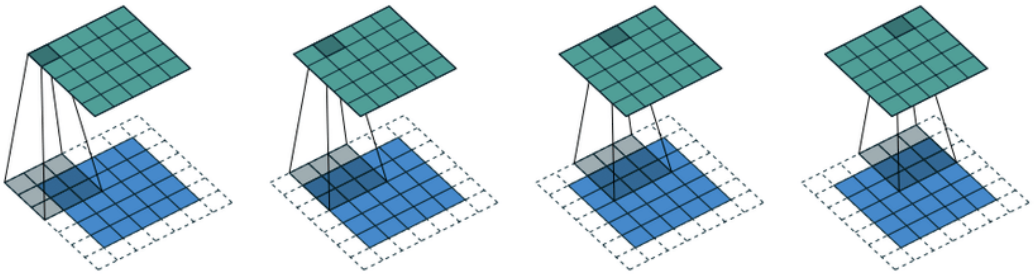


FIGURE A.2: the input (blue) is padded with $p = 1$. The kernel size is 3 and the stride is $s = 2$. As a consequence, the output (cyan) has half the size of the padded input.

Appendix B

EuroSat Dataset Preview

B.1 Dataset Acquisition

As they mentioned in their paper they used multispectral image data provided by the Sentinel-2A satellite in order to address the challenge of land use and land cover classification. Sentinel-2A is one satellite in the two-satellite constellation of the identical land monitoring satellites Sentinel-2A and Sentinel-2B. The satellites were successfully launched in June 2015 (Sentinel-2A) and March 2017 (Sentinel-2B). Both sun-synchronous satellites capture the global Earth's land surface with a Multi-spectral Imager (MSI) covering the 13 different spectral bands listed in Table B.1. The three bands B01, B09, and B10 are intended to be used for the correction of atmospheric effects. The remaining bands are primarily intended to identify and monitor land use and land cover classes. In addition to the mainland, large islands, as well as inland and coastal waters, are covered by these two satellites. Each satellite will deliver imagery for at least 7 years with a spatial resolution of up to 10 meters per pixel. Both satellites carry fuel for up to 12 years of operation which allows for an extension of the operation. The two-satellite constellation generates a coverage of almost the entire Earth's land surface about every five days, i.e. the satellites capture each point in the covered area about every five days. This short repeat cycle as well as the future availability of the Sentinel satellites allows continuous monitoring of the Earth's land surface for about the next 20 - 30 years. Most importantly, the data is openly and freely accessible and can be used for any application (commercial or non-commercial use) [14].
To construct an image classification dataset, they performed the following two steps:

1. *Satellite Image Acquisition*: We gathered satellite images of European cities distributed in over 34 countries as shown in Figure B.1.

2. *Dataset Creation*: Based on the obtained satellite images, we created a dataset of 27,000 georeferenced and labeled image patches. The image patches measure 64×64 pixels and have been manually checked

1. Satellite Image Acquisition: The chosen satellite images are associated with the cities covered in the European Urban Atlas. The covered cities are distributed over the 34 European countries: Austria, Belarus, Belgium, Bulgaria, Cyprus, Czech Republic (Czechia), Denmark, Estonia, Finland, France, Germany, Greece, Hungary, Iceland, Ireland, Italy / Holy See, Latvia, Lithuania, Luxembourg, Macedonia, Malta, Republic of Moldova, Netherlands, Norway, Poland, Portugal, Romania, Slovakia, Slovenia, Spain, Sweden, Switzerland, Ukraine, and United Kingdom.

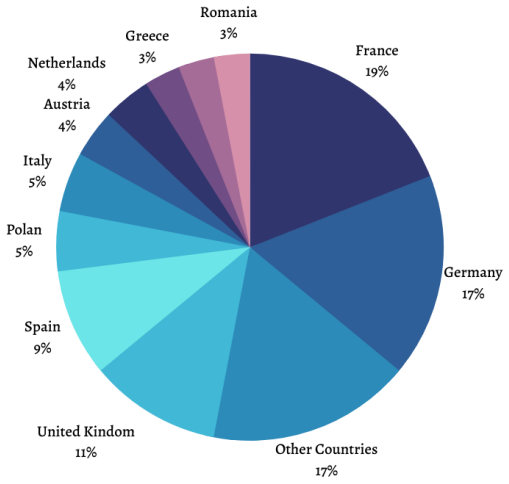


FIGURE B.1: EuroSAT dataset distribution.

The objective is to cover as many countries as possible in the EuroSAT dataset in order to cover the high intra-class variance inherent to remotely sensed images. Within one class of the EuroSAT dataset, different land types of this class are represented such as different types of forests in the forest class or different types of industrial buildings in the industrial building class. Between the classes, there is a low positive correlation. The classes most common to each other are the two presented agricultural classes and the two classes representing residential and industrial buildings [14].

2. Dataset Creation: The generation of the benchmarking EuroSAT dataset was motivated by the objective of making this open and free satellite data accessible to various Earth observation applications and the observation that existing benchmark datasets are not suitable for the intended applications with Sentinel-2 satellite images. The dataset consists of 10 different classes with 2,000 to 3,000 images per class. In total, the dataset has 27,000 images. The patches measure 64x64 pixels. They have chosen 10 different land use and land cover classes based on the principle that they showed to be visible at the resolution of 10 meters per pixel and are frequently enough covered by the European Urban Atlas to generate thousands of image patches.

To differentiate between different agricultural land uses, the proposed dataset covers the classes of the annual crop, permanent crop sand pastures. The dataset also discriminates built-up areas. It, therefore, covers the classes of the highway, residential buildings, and industrial buildings. The residential class is created using the urban fabric classes described in the European Urban Atlas. Different water bodies appear in the classes river and sea & lake.

TABLE B.1: All 13 bands covered by Sentinel-2's Multispectral Imager (MSI). The identification, the spatial resolution, and the central wavelength is listed for each spectral band.

Band	**Spatial Resolution** m	**Central Wavelength** nm
B01 - Aerosols	60	443
B02 - Blue	10	490
B03 - Green	10	560
B04 - Red	10	665
B05 - Red edge 1	20	705
B06 - Red edge 2	20	740
B07 - Red edge 3	20	783
B08 - NIR	10	842
B08A - Red edge 4	20	865
B09 - Water vapor	60	945
B10 - Cirrus	60	1375
B11 - SWIR 1	20	1610
B12 - SWIR 2	20	2190

Appendix C

Sentinel-2 Satellite

Sentinel-2 is a Copernicus Earth observation spacecraft that systematically collects optical pictures at high spatial resolution (10 m to 60 m) across land and coastal waterways. Sentinel-2A and Sentinel-2B are the existing satellites in the constellation; a third satellite, Sentinel-2C, is now undergoing testing in preparation for launch in 2024. Agricultural monitoring, emergency management, land cover categorization, and water quality are among the services and applications supported by the mission [33].
The European Space Agency designed and operates Sentinel-2, and the satellites were built by a partnership led by Airbus Defence and Space [33].

C.1 Characteristics of Sentinel-2

- Multi-spectral data with 13 bands in the visible, near infrared, and short wave infrared part of the spectrum.
- Coverage of land surfaces from 56° S to 84° N, as well as coastal waterways and the whole Mediterranean Sea.
- Revisiting every 10 days under the same viewing angles.. Sentinel-2 swaths overlap at high latitudes, therefore certain places will be examined twice or more every 10 days, although from different viewing angles.
- Spatial resolution of $10m$, $20m$ and $60m$.
- The field of vision is 290 kilometres.
- Data policy that is free and open.

C.2 Sentinel-2 Launches

The first satellite, Sentinel-2A, was launched aboard a Vega launch vehicle on June 23, 2015 at 01:52 UTC. Sentinel-2B was launched on 7 March 2017 at 01:49 UTC, also aboard a Vega rocket. Sentinel-2C is scheduled to launch in 2024 on a Vega-C launch vehicle [33].

FIGURE C.1: View of Sentinel-2 satellite [33].

C.3 Two-line Element Set

A two-line element set (TLE) is a data format encoding a list of orbital elements of an Earth-orbiting object for a given point in time, the epoch. Using a suitable prediction formula, the state (position and velocity) at any point in the past or future can be estimated to some accuracy [39].
TLEs can describe the trajectories only of Earth-orbiting objects. TLEs are widely used as input for projecting the future orbital tracks of space debris for purposes of characterizing "future debris events to support risk analysis, close approach analysis, collision avoidance maneuvering" and forensic analysis [39].

TABLE C.1: Two Line Element Set (TLE) of Sentinel-2 [32].

1	40697U	15028A	22145.46633874	0.00000189	00000-0	88716-4 0	9999
2	40697	98.5687	220.5700	0001242	91.346	268.7863	14.30815399361569