Carl von Ossietzky University Oldenburg

Master's Program Physics

MASTER THESIS

Deutsches Zentrum
für Luft- und Raumfahrt
German Aerospace Center

Institute of
Networked Energy Systems

Pipeline Detection with Satellite Images Using Machine Learning

Submitted by: Jan Dasenbrock

Supervising Examiner: Prof. Dr. Carsten Agert

Second Examiner: Dr. Adam Pluta

Oldenburg, 04.11.2020

Contents

1 Introduction **1**

2 Gas Transport Networks **3**

2.1 Fuel Gases . . . 3

2.2 The European Gas Transport Network . . . 4

2.3 The Construction of Pipelines . . . 8

3 Satellite Imagery of Pipelines **11**

3.1 Sensors . . . 12

3.2 Vegetation Changes . . . 16

4 Machine Learning **19**

4.1 Neural Networks . . . 19

4.1.1 Neuron . . . 20

4.1.2 Structure of Neural Networks . . . 21

4.1.3 Training of Neural Networks . . . 23

4.1.4 Cost Functions . . . 24

4.1.5 Back-Propagation . . . 25

4.1.6 Optimizer . . . 29

4.2 Convolutional Neural Networks . . . 31

4.2.1 Max-Pooling . . . 33

4.2.2 Batch Normalization . . . 34

4.2.3 Concatenation . . . 34

4.3 Semantic Image Segmentation . . . 35

4.4 Performance Assessment . . . 36

4.4.1 Confusion Matrix . . . 36

4.4.2 Intersection over Union . . . 37

4.4.3 Cross-Validation . . . 38

5 Methods **41**

5.1 Data Acquisition and Pre-Processing . . . 41

5.1.1 Pipeline Course and Construction Date 41
5.2 Data Preparation 44
5.3 Model Design and Training 45
5.3.1 Encoder 46
5.3.2 Decoder 46
5.3.3 Skip Connections 47
5.3.4 Training 47

6 Results and Discussion **51**

7 Proposed Process **61**
7.1 Definition of Region and Generation of Image List 62
7.2 Pipeline Detection 62
7.3 Post-Processing 63

8 Conclusion and Outlook **65**

Bibliography **67**

List of Figures

2.1 Natural gas use in Europe by sector (2010). 4
2.2 Map of 44 TSO members, 3 associated partners and, 9 observers of ENTSOG (2020). 6
2.3 Detailed map of the ENTSOG members in Austria, Germany, and Switzerland with TSO members and observers of ENTSOG (2020). . 6
2.4 European gas transport network as in 2009. Background colors indicate members of the European Union. 7
2.5 Composition of European natural gas imports of 2018 with year-over-year growth compared to 2017. 8
2.6 Schematic illustration of offshore pipeline construction 9
2.7 Pipeline joint with concrete and anti-corrosion coating of Nord Stream for public display in Kotka, Finland. 10
2.8 Schematic illustration of offshore pipeline construction. 10

3.1 Landsat mission timeline with planned mission start of Landsat 9 in 2021. 12
3.2 Schematic illustration of whisk broom scanner. 13
3.3 Schematic illustration of a push broom scanner. 14
3.4 Atmospheric transmission of sunlight in percent over the wavelength in nm with spectral senitivity of each band of all scanners of the Landsat missions. 15
3.5 Atmospheric transmission of sunlight in percent over the wavelength in µm with spectral senitivity of each band of MSI scanner of the Sentinel-2 mission. 15
3.6 Reflection and absorbtion of near-infrared and visible light of dense vs. sparse vegetation with resulting NDVI. 17

4.1 Schematic illustration of an artificial neuron with x_1 to x_N as input and a as output. 20
4.2 Plot of Rectified Linear Unit. 21
4.3 Neural network with input, hidden and output layer. 21

4.4 Activation of layer L and $L-1$. 22
4.5 Schematic illustration of the training process of neural networks. . . . 24
4.6 Schematic illustration of the calculation of the dice coefficient. 25
4.7 Back-propagation of the errors δ through a neural network. 29
4.8 Three images of an eight of the MNIST dataset. 31
4.9 Schematic illustration of convolutional neural network (4.9a) with 3×3 Kernel (4.9b). 32
4.10 Kernel movement with stride of 2 px. 32
4.11 Schematic illustration of a convolution followed by a subsampling layer. 33
4.12 Schematic illustration of 2×2 max-pooling operation on a 4×4 matrix. The colored backgrounds indicate the area of each max operation. . . 34
4.13 Schematic illustration of a concatenation of two layers (red and green) along the concatenation axis (blue arrow). 35
4.14 Schematic illustration of the calculation of the Intersection over Union. 38
4.15 Schematic illustration of k-fold cross validation. The grey boxes represent the training data whereas the white represent the validation data in each split. 39

5.1 Course of NEL pipeline (red) in Northern Germany. 42
5.2 Map of energy infrastructure of National Grid UK. 43
5.3 Raw satellite image and corresponding mask. 44
5.4 Schematic illustration of the cutting process to create image-mask pair. The left side shows the satellite image. The right side shows the corresponding mask. 45
5.5 Process of training data generation with the raw data as input and the image-mask pairs as output. 45
5.6 Schematic illustration of the u-net-based model used in this thesis. The left part shows the encoder reducing the input image size from 64×64 px to 2×2 px. The right part shows the decoder increasing the image size to 64×64 px again. 46
5.7 Schematic illustration of training process using 5-fold cross-validation and subsequent testing on a separate dataset. 47
5.8 Overview of five augmentations used on the image-mask pairs during the training process. 48
5.9 The principle of grid shuffling, demonstrated with a picture of my dog "Sam". 48

6.1 Minimum loss and maximum IoU score averaged over all training splits for each learning rate with standard deviation. 52
6.2 IoU over number of epochs for split 2. 53

6.3 Loss over number of epochs for split 2. 54
6.4 Mean loss and IoU score of trained models applied to test dataset for each learning rate with standard deviation. 55
6.5 A selection of predictions of the trained model on the validation dataset. 57
6.6 A selection of predictions of the trained model on the test dataset. . 58

7.1 Overview of steps of proposed process for the automatic extraction of pipeline courses from satellite images. Ellipses represent inputs and outputs while rectangles represent process steps. 62

List of Tables

3.1 Overview of main scanner type, years of operation and spatial resolution for all Landsat missions and Sentinel 2. 14
3.2 Landsat 5 band overview with spectral range and spatial resolution. . 15

4.1 Confusion Matrix. 36
4.2 Confusion Matrix for two classifiers with same precision and recall but different classifier behaviors. 37

Chapter 1

Introduction

Gas transmission networks are pipeline systems that transport gas from production sites and import terminals to local distribution networks. Network models of gas transport networks form the basis for static and dynamic simulations to characterize important network properties. Furthermore, such simulations have a wide range of applications, such as the modeling of gas consumption, the identification of bottlenecks, network expansion, or the optimization of gas distribution strategies [1].

The European gas transmission network consists of a collection of sub-networks each operated by one Transmission System Operator (TSO). Each gas transmission network consists of gas terminals or gas entry points, compressor stations, pressure valves, and storage facilities connected by pipelines. Gas terminals feed gas into the network. They include Liquefied Natural Gas (LNG) terminals, entry points from on- and offshore gas fields, and interconnection pipelines from neighboring transmission networks. Interconnection points can also act as a gas exit and hence, enable the trade of gas between neighboring TSOs. The gas in the network is transported to storage sites and distribution networks through pipelines. Compressor stations direct the flow and maintain the pressure in these pipelines. Pressure valves lower the pressure for local distribution networks from the high-pressure gas transmission network. [2]

This thesis has been written in the framework of the SciGRID_gas project, which has the goal of creating an open-source network model for the European gas transport network. For the creation of the model, public data sources such as OpenStreetMap or press articles are currently used. These conventional data sources tend to be inaccurate or incomplete. For the correction of these inaccuracies and incompleteness, satellite images could play an important role.

The aim of this thesis is to prove that a pipeline dataset can be automatically generated from satellite images using machine learning methods. For this, a deep learning

model is trained to recognize pipeline courses on satellite images. Training data will first be generated using gas network data from Great Britain. Open-access satellite data will be used which is easily accessible, of adequate resolution, and is well documented. A suitable model will be trained with these data. The trained model will then be tested with satellite images from Northern Germany. The results will be evaluated and a possible future large-scale application of the model to the whole European area will be discussed.

To accomplish that goal Chapters 2 and 3 give the basis of the theoretical knowledge needed to understand the detection of pipelines on satellite imagery. Thereafter, deep learning tools for the processing of satellite images are portrayed followed by a chapter about the model and machine learning methods used in this thesis. In the subsequent chapter, the training process and the performance of the model on a test dataset are presented and discussed. Based on these findings, a fully automatic process for the extraction of pipeline courses from satellite images is presented in Chapter 7. Finally, the thesis is concluded with a summary of the content and an outlook on possible further enhancements of the model discussed in the prior chapters.

Chapter 2

Gas Transport Networks

This chapter describes the basic theoretical background of gas transport networks. It starts with the history of fuel gases in Europe, followed by the history and current state of the transport network in Europe for natural gas. Finally, the construction of pipelines for these transport networks is described.

2.1 Fuel Gases

Fuel gases are flammable gases or gas composites [3]. Today, fuel gases are an important energy source with a wide variety of applications. These applications mainly include electric power generation, heating and cooking in the residential and service sector and the industry (see Fig. 2.1) [4]. The most used fuel gas is natural gas [5]. Natural gas was formed through biogenic and thermogenic processes from trapped organic matter within sediments over the past 550 million years. It is a mixture of hydrocarbon and nonhydrocarbon constituents with its main component being methane. Some additional components which can be found in significant quantities are ethane, propane, butane, and pentane [6].

Natural gas can also be liquefied to Liquefied Natural Gas (LNG) by cooling it to approx. $-162\,^{\circ}\mathrm{C}$ at atmospheric pressure. This reduces it to $1/600$ of its original volume and allows for long-distance transport by ship. Also, remote gas fields that economically do not make sense to be accessed by pipeline can be used with LNG technology [7]. Comparing LNG with transportation by pipeline, it is cost effective for distances above 2500 km and coastal regions [8]. However, recent political tensions between e.g. Europe and the gas supplier Russia might further increase the demand for LNG from countries like the United States despite opposing economic factors

[8].

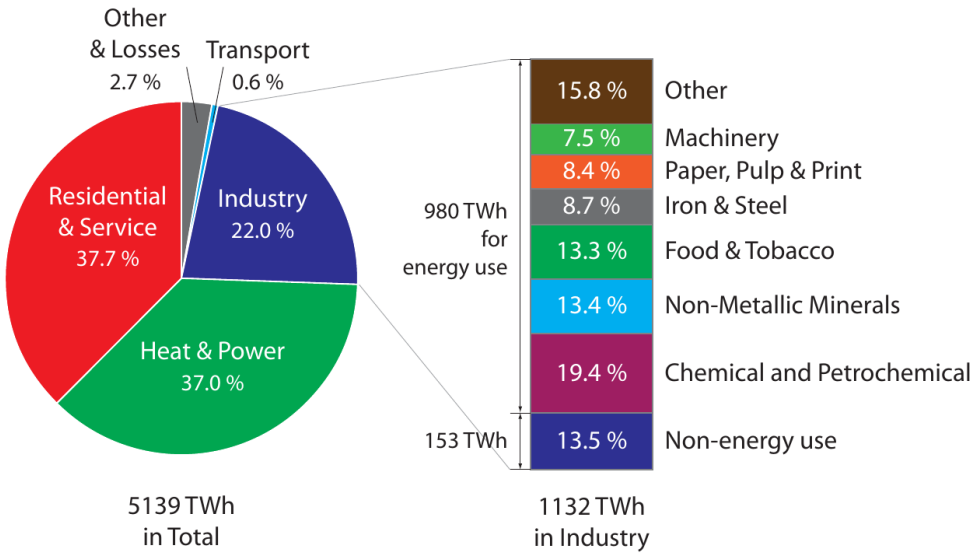


Figure 2.1: Natural gas use in Europe by sector (2010). [4]

In the 19th century, fuel gases rose to importance with the illumination of the streets at night. For this, mainly coal gas was used. Coal gas is obtained by the gasification of coal which produces a flammable gas mixture of hydrogen, carbon monoxide, methane, and ethylene [9]. During the course of the century and the beginning of the 20th century, its application extended to cooking and heating [10]. The transition to natural gas as the primary fuel gas began with the discovery of a natural gas field near Groningen in the Netherlands [11]. This marked the start of the construction of a gradually growing gas network in Europe, mainly consisting of pipelines in the gas-producing countries, which will be discussed in the following.

2.2 The European Gas Transport Network

Although many regions in Europe already implemented infrastructure for the use of natural gas it only played a minor role as a fuel source. However, after the two oil crises in 1973 and 1979, large imports of natural gas from Russia, Norway, and Algeria took over as the main fuel source in Europe in the 1980s. With the usage of natural gas as a primary fuel gas, large cross-national infrastructure for its distribution was needed. This resulted in a rapidly growing network of pipelines throughout Europe [11].

The pipeline network further extended with the introduction of natural gas in Greece, Portugal, and Ireland in the 1990s. After 2000 the focus laid on the connection of

the gas market of the United Kingdom (UK) to continental Europe and Norwegian gas fields. Additionally, new member states to the European Union (EU) were linked to the EU-integrated system and new import channels from North Africa and the Caspian Sea were added [11]. To further diversify the gas market in Europe, LNG terminals were built at several locations: at the Mediterranean Sea, the Atlantic coast, the North Sea, and the Baltic Sea [12] [11].

Today's gas infrastructure projects mainly focus on the reinforcement of the North-Africa-Southern-Europe connection, the connection of central Europe and Russia, and the connection of the Caspian region to central and Western Europe [11].

Pipelines are usually constructed by specialized contractors commissioned by a Transmission System Operator (TSO) [13]. The development of the European gas transport network is currently coordinated by the European Network of Transmission System Operators for Gas (ENTSOG) established by the European Commission in 2009. ENTSOG is an association of 44 European TSOs, 3 associated partners, and 9 observers (See Fig. 2.2 and Fig. 2.3). Its goal is to *"facilitate and enhance cooperation between national gas transmission system operators (TSOs) across Europe, to ensure the development of a pan-European transmission system in line with European Union energy goals"* [14]. To achieve this goal, TSO members of ENTSOG are obliged to publish information regarding capacity, nominations, flow rates and tariffs pertaining border points, which are interconnection points between the networks of TSOs [15].

Figure 2.2: Map of 44 TSO members, 3 associated partners, and 9 observers of ENTSOG (2020). [16]

Figure 2.3: Detailed map of the ENTSOG members in Austria, Germany, and Switzerland with TSO members and observers of ENTSOG (2020). [16]

The total length of the European gas pipeline network amounted to approx. 400 000 km in 2008 of which 112 000 km belonged to the gas transport network (see Fig. 2.4)

[17]. The pipe diameter ranges between 38 cm and 229 cm. The largest share of the transport network in the EU is in Germany which amounted to a length of 27 000 km in 2018 [18]. This already suggests the importance of the network to the European energy market. Using the European transport network, 21 % of the annual energy consumption of all members of the EU in the year 2018 was covered [19]. The main suppliers of natural gas of the same year were Russia with 38.8 % and Norway 27.0 % (see Fig. 2.5) [19]. LNG has a share of 12.4 % and is mainly supplied by Qatar (28 %), Russia (20 %), the United States (16 %), and Nigeria (12 %) [20].

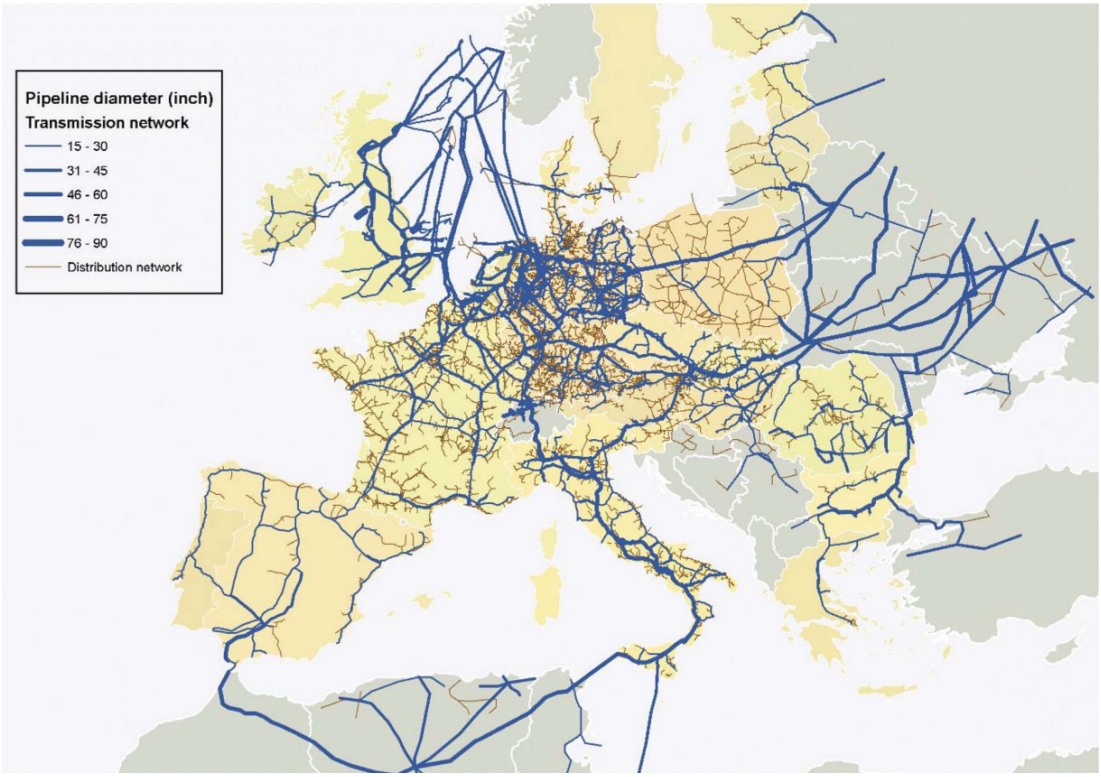


Figure 2.4: European gas transport network as in 2009. Background colors indicate members of the European Union. [17]

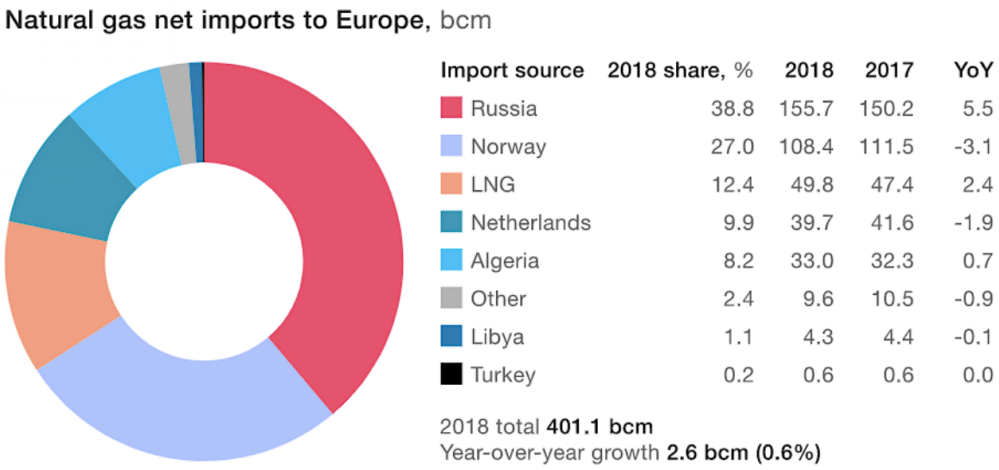


Figure 2.5: Composition of European natural gas imports of 2018 with year-over-year growth compared to 2017. [21]

2.3 The Construction of Pipelines

The European gas transport network consists of offshore and land pipelines. Offshore pipelines for example connect the gas market of Great Britain (GB) to Continental Europe or the Norwegian offshore gas fields to the mainland [11]. For the detection of pipelines on satellite images, it is important to understand the environmental changes created by their construction. The construction of offshore and land pipelines differ significantly and will be discussed in the following sections [13].

Before the start of the land pipeline construction, an area next to the planned pipeline, commonly called "right of way", needs to be acquired (see Fig. 2.6). The width of the right of way usually varies between 10 m and 50 m. Large right of ways are generally required for rougher terrain and larger pipeline diameters. It is mainly used during the construction phase and afterward, protects the pipelines from damage through the roots of trees and simplifies maintenance and leak detection. [13]

Figure 2.6: Construction of NEL gas pipeline near Hasenhäge, Germany. [22]

At the beginning of the construction phase, the right of way is cleared from all vegetation to make it maneuverable for construction vehicles. A ditch is dug, depending on the pipeline diameter and the local regulations. The pipe joints of the pipeline are strung along the ditch. After the joints are welded together, coating and wrapping to protect the pipeline from corrosion are added. The pipeline is lowered into the ditch which is then backfilled with the previously excavated soil. Finally, the line segment is tested for leaks and the right of way is cleaned up. [13]

The construction of offshore and onshore pipelines shares many operations. However, most of the operations are either done before the actual assembly of the pipeline or on the open sea. There are several methods for the construction of offshore pipelines. Most of them require the use of heavy marine gear. The most common method is the lay barge method which will be discussed in the following [13].

A lay barge is a marine vessel containing several welding stations, an inspection station, a coating station followed by a tensioner with a stinger. The individual pipe joints are stored on the vessel. Anti-corrosion coating is usually applied ashore leaving a section of bare pipe at both ends of the joint for welding (see Fig. 2.7). The individual joints are welded together and afterward the welding seams are radiographically inspected. The welding seams are coated with anti-corrosion coating. An additional layer of concrete coating is added to add negative buoyancy to keep the pipeline on the sea-floor. The extra layer also protects the anti-corrosion coating from damage by e.g. fishing gear. [23] [13]

Figure 2.7: Pipeline joint with concrete and anti-corrosion coating of Nord Stream for public display in Kotka, Finland. [24]

The completed pipeline segments are led of the rear of the barge continuously through a stinger by tensioners in a controlled manner (see Fig. 2.8). In combination with a system of anchors and anchor wenches at the barge, the pipeline is kept at tension to reduces stresses within the pipeline and ensure a successful laying of the pipeline on sea ground. Finally, the pipeline is anchored to the ground to prevent shifting due to currents and other forces. [23] [13]

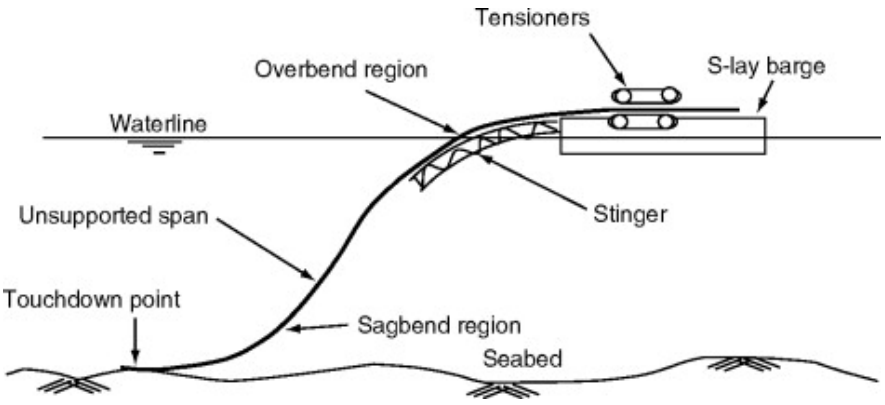


Figure 2.8: Schematic illustration of offshore pipeline construction. [23]

Chapter 3

Satellite Imagery of Pipelines

In this thesis, satellite imagery is used to detect pipelines. In the following sections, the essential characteristics and the main sources of satellite imagery are discussed.

Satellite imagery usually refers to images of Earth's surface of a light frequency range in the visual and infrared spectrum made by satellites. Satellite imagery is often divided into scenes of a specific size. A scene portrays a certain area at a particular timestamp. Scenes consist of a certain number of bands. Each band corresponds to the light's intensity measured by the satellites sensor sensitive to a particular wavelength range [25]. Multiple spectral bands are often more valuable than single or broadband images [25].

To the knowledge of the author there are two main sources for suitable open-access satellite imagery:

- The Sentinel-2 mission of the **Copernicus program** by the European Space Agency (ESA) [26]
- The **Landsat program** by the United States Geological Survey (USGS) and National Aeronautics and Space Administration (NASA) [27].

The Landsat program performs the systematic collection of land images with the longest observation span. It is a joint program by NASA and USGS which started with the first launch of the satellite Landsat 1 in 1972 (see Fig. 3.1). The latest launched satellite of the program is Landsat 8. All missions differ significantly with the onboard mounted optical sensory [27].

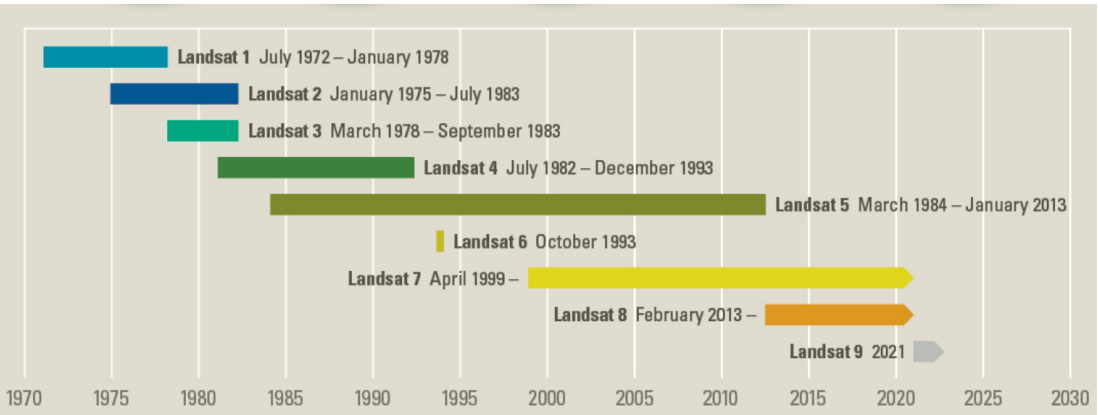


Figure 3.1: Landsat mission timeline with planned mission start of Landsat 9 in 2021. [28]

All spectral bands not only differ in resolution but also in spectral range and radiometric performance. The optical sensor used in the missions Landsat 1 to 3 was the Multi Spectral Scanner (MSS). The three missions span from 1972 to 1983. The missions Landsat 4 and 5 from 1982-1984 had an additional sensor called Thematic Mapper (TM). Landsat 7, which started in 1999, carried an upgraded version of the TM called Enhanced Thematic Scanner Plus (ETM+). The most recent Landsat 8 mission launched in 2013 uses the Operational Land Imager (OLI) - Thermal InfraRed Sensor (TIRS).

The Copernicus program is the earth observation program of the European Union. Managed by the European Commission with the European Space Agency (ESA). Its goal is a global, continuous, autonomous, high quality, and wide range Earth observation. To achieve this, the Copernicus program started several Earth-observing satellite missions called Sentinel. The Sentinel-2 mission consists of two satellites each mounted with the Multi Spectral Imager (MSI).

3.1 Sensors

The optical sensors MSS, TM, and ETM+ of the aforementioned satellites are whisk broom (WB) scanners. They consist of a rotating mirror scanning the ground track, deflecting light into a detector. The detector scans one pixel at a time, gradually building an image of a certain size (see Fig. 3.2). Scenes of the TM scanner cover an area of 185 km $\times$ 172 km with a resolution of 60 m px^{-1} with its thermal band and 30 m px^{-1} for the remaining bands. Scenes of the ETM+ scanner cover an area of 183 km $\times$ 170 km with a resolution of 15 m px^{-1} with its panchromatic band, 60 m px^{-1} for the thermal band and 30 m px^{-1} for the remaining bands. The scenes of the MSS

scanner cover an area of 185 km $\times$ 185 km with a resolution of 80 m px^{-1}.

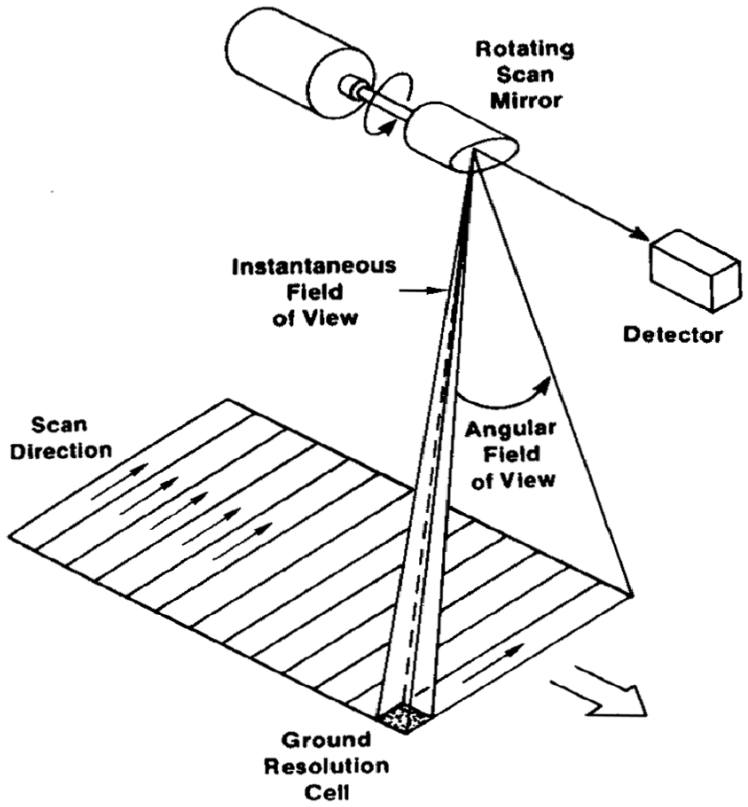


Figure 3.2: Schematic illustration of whisk broom scanner. [29]

The OLI-TIRS and the MSI sensors are push broom (PB) scanners. They consist of a linear array of detectors each of them scanning one line of pixels at a time (see Fig. 3.3). The OLI-TIRS outputs scenes which cover an area of 183 km $\times$ 170 km whereas the MSI scanner outputs scenes of the dimensions 290 km $\times$ 290 km. The MSI has a resolution of 60 m px^{-1} in the atmospheric correction bands, 20 m px^{-1} in its short wave infrared (SWIR) and red edge bands, and 10 m px^{-1} for the remaining bands. The OLI-TIRS scenes have a resolution of 15 m px^{-1} for the panchromatic, 100 m px^{-1} for the thermal and 30 m px^{-1} for the remaining bands in the visible and near-infrared (NIR) wavelength range. All missions with their relevant scanners are summarized in Tab. 3.1.

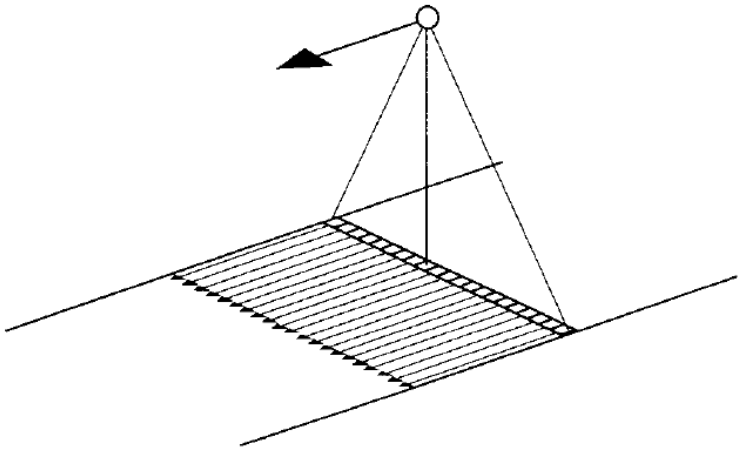


Figure 3.3: Schematic illustration of a push broom scanner. The arrows indicate the travel direction of the satellite. [30]

Table 3.1: Overview of main scanner type, years of operation and spatial resolution for all Landsat missions and Sentinel 2. [31]

Missions	Main scanner	Type	Years operation	Resolution
Landsat 1-3	MSS	WB	1972 - 1983	80 m px^{-1}
Landsat 4-5	TM	WB	1982 - 2011	30 m px^{-1} 60 m px^{-1} thermal
Landsat 7	ETM+	WB	1999 - now	15 m px^{-1} panchromatic 60 m px^{-1} thermal 60 m px^{-1} remaining
Landsat 8	OLI-TIRS	PB	2013 - now	15 m px^{-1} panchromatic 100 m px^{-1} thermal 30 m px^{-1} remaining
Sentinel-2	MSI	PB	2015 - now	10 m px^{-1} visible + NIR 20 m px^{-1} red edge + SWIR 60 m px^{-1} atm. corr.

Each band of a scene collected by a certain sensor corresponds to a specific wavelength range. The first band of each Landsat 5 TM scene for example corresponds to the wavelength range 0.45 µm to 0.52 µm or the color blue (see Tab. 3.2). An overview of the wavelength ranges for each band for all Landsat scanners can be seen in Fig. 3.4. The wavelength ranges for all Sentinel-2 bands can be seen in Fig. 3.5.

Table 3.2: Landsat 5 band overview with spectral range and spatial resolution. [32]

Band	Spectral range in µm	Resolution in m px^{-1}
Band 1 - blue	0.45 - 0.52	30
Band 2 - green	0.52 - 0.60	30
Band 3 - red	0.63 - 0.69	30
Band 4 - near-infrared (NIR)	0.76 - 0.90	30
Band 5 - short wave infrared (SWIR)	1.55 - 1.75	30
Band 6 - thermal	10.40 - 12.50	120
Band 7 - short wave infrared (SWIR)	2.08 - 2.35	30

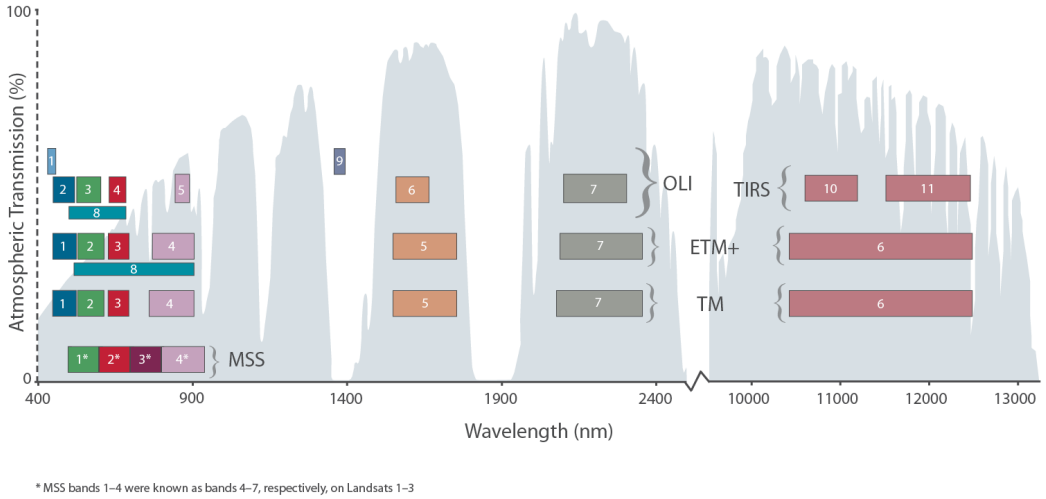


Figure 3.4: Atmospheric transmission of sunlight in percent over the wavelength in nm with spectral senitivity of each band of all scanners of the Landsat missions. [33]

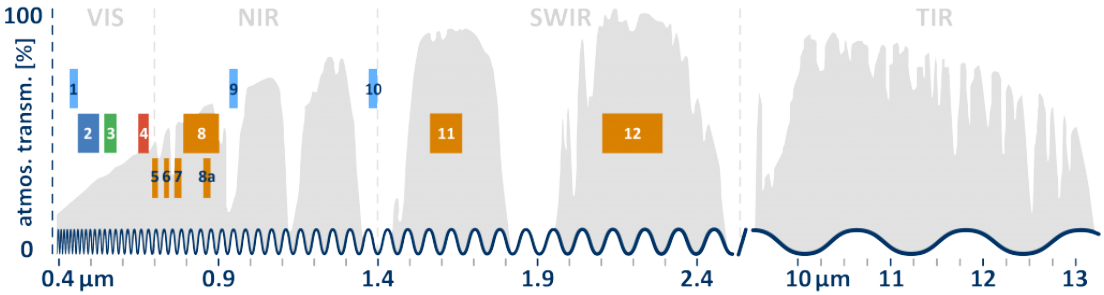


Figure 3.5: Atmospheric transmission of sunlight in percent over the wavelength in µm with spectral senitivity of each band of MSI scanner of the Sentinel-2 mission. [34]

3.2 Vegetation Changes

The construction of land pipelines causes temporal changes in vegetation density between the right of way of the pipeline and the surrounding area (see Sec. 2.3). Differences in vegetation density can be emphasized in remote sensing by observing the visible and near-infrared bands of satellite imagery. This is due to the difference in absorption and reflection of sunlight between different surfaces.

The Sun's spectrum closely corresponds to the radiation of a black body at 5800 K [35]. However, due to Earth's atmosphere, parts of the UV light is absorbed. Most of the spectrum covers wavelengths between 200 nm and 2500 nm [36]. In general, sunlight is absorbed and reflected by objects on the ground in specific wavelengths and is perceived as color. The pigment chlorophyll is highly present in vegetation. Chlorophyll strongly absorbs visible light (400 nm - 700 nm) to carry out photosynthesis. The cell structures of vegetation on the other hand strongly reflect near-infrared light (700 nm to 1100 nm). Areas of sparse vegetation however, reflect less near-infrared light and more visible light [37].

To quantify this difference in reflection and absorption in satellite imagery, the Normalized Difference Vegetation Index (NDVI) is introduced. The NDVI is the difference between the intensity of the reflected near-infrared light normalized by the sum of the near-infrared and visible light (see Eq. 3.1). It gives a number between -1 and 1. An NDVI value closer to 1 indicates a higher vegetation density whereas a value closer to -1 indicates lower density (see Fig. 3.6). [37]

$$NDVI = \frac{NIR - VIS}{NIR + VIS} \tag{3.1}$$

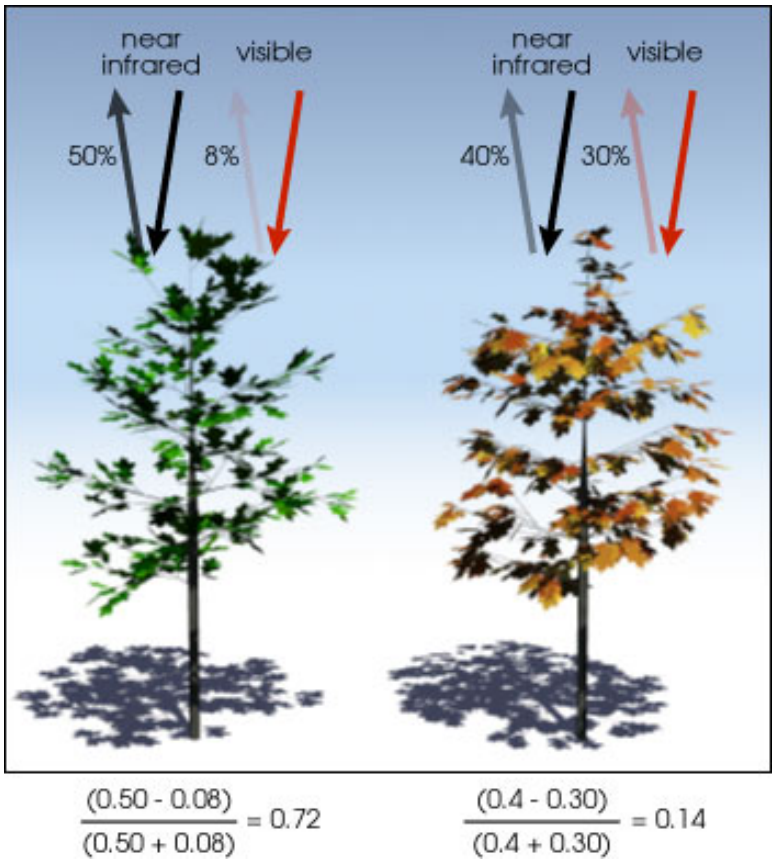


Figure 3.6: Reflection and absorbtion of near-infrared and visible light of dense vs. sparse vegetation with resulting NDVI. [37]

The calculation of the NDVI is made, based on the satellite specific sensor bands. For the TM sensor of the Landsat 5 mission band 3 and band 4 are used which correspond to the red and near-infrared respectively [38]:

$$NDVI = \frac{Band\,4 - Band\,3}{Band\,4 + Band\,3}. \tag{3.2}$$

Chapter 4

Machine Learning

Machine learning is the basis for the automated recognition of pipelines in satellite images. This chapter explains the relevant fundamental theory, starting with the basics for artificial neural networks. Based on this, it follows the theory for convolutional neural networks and their application on semantic segmentation tasks. The chapter concludes with the assessment methods used for the algorithms presented.

4.1 Neural Networks

Machine learning is a branch of artificial intelligence which automates solutions to complex problems which are hard to implement in a conventional way [39]. Using the conventional way to create an algorithm, the certain steps needed to arrive at a desired solution need to be specified. These steps then need to be implemented in a computer program. For many real-live examples these steps are hard to define, e.g. for the classic example of detecting handwritten digits on images. Despite the clear specifications of the problem, its hard to define a general rule on how the image relates to the digit, due to the various ways the digit can be drawn [40]. In the following, one way of solving these kinds of problems is presented, namely artificial neural networks. An artificial neural network "*is a mathematical model that is capable of solving and modeling complex data patterns and prediction problems. Neural network algorithms are developed by replicating and using the processing of the brain as a basic unit.*" [41] This section will discuss the basic units of artificial neural networks, the way these units interact and how neural networks are trained to create a desired output from a given input.

4.1.1 Neuron

Artifical neurons are the elementary units of artificial neural networks. They are a mathematical model of their biological counterpart [42]. Several inputs x_1 to x_N between 0 and 1 produce a single output a called activation (see Fig. 4.1). Each input x_m is first weighted with its corresponding weight w_m. All weighted inputs are summed up. This weighted sum minus a bias b serves as the input for a non-linear function φ, often called activation function. The bias guarantees an input to the function even if all other inputs are 0. The activation function φ produces the final output a of the neuron. This can be mathematically expressed by equation 4.1 [43, Chapter 5].

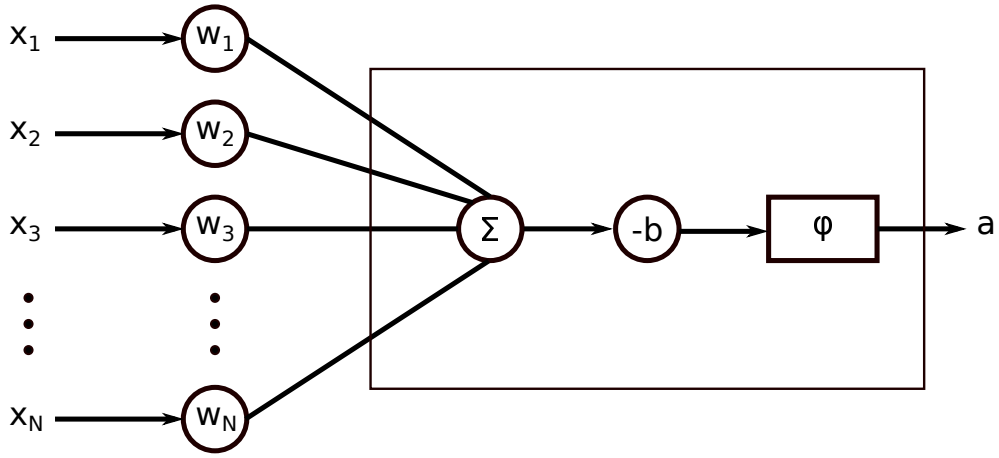


Figure 4.1: Schematic illustration of an artificial neuron with x_1 to x_N as input and a as output.

$$a = \varphi\left(\sum_{m=1}^{N} w_m \cdot x_m - b\right) \tag{4.1}$$

One common activation function is the Rectified Linear Unit (ReLU). As can be seen in Fig. 4.2, all negative inputs are mapped to 0, whereas all positive inputs are mapped linearly. This can be mathematically described by the following equation:

$$\varphi(x) = \max(0, x) \tag{4.2}$$

where x is the input. [44]

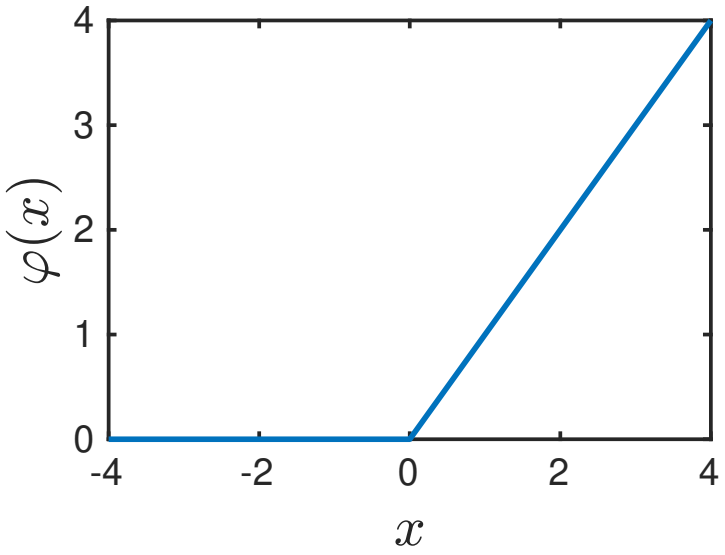


Figure 4.2: Plot of Rectified Linear Unit.

4.1.2 Structure of Neural Networks

Neural networks are composites of layers of artificial neurons. The weights and biases of each neuron in the composite can be adapted to fulfill certain tasks, e.g. image recognition. In general, neural networks consist of an input layer, an output layer and usually several hidden layers (see Fig. 4.3). The input layer reads in the inputs x_1 to x_N. Each input x_m reads a value between 0 and 1. The hidden and output layer consist of several neurons interconnected to the neurons of the layer before. The outputs a of the neurons in one layer serve as the inputs for the next layer.

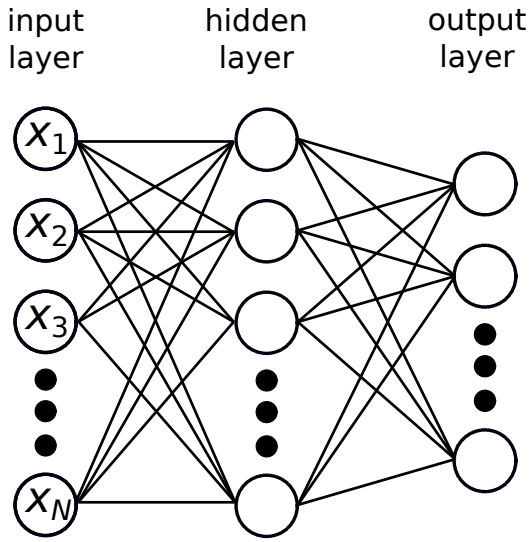


Figure 4.3: Neural network with input, hidden and output layer.

The activation $a_j^{(L)}$ of the j-th neuron in the L-th layer can be expressed as

$$a_j^{(L)} = \varphi\left(\sum_{k=1}^{N_{L-1}} w_{jk}^{(L)} a_k^{(L-1)} - b_j^{(L)}\right) \tag{4.3}$$

where N_{L-1} is the number of neurons in the $(L-1)$-th layer, $w_{jk}^{(L)}$ are the weights between the j-th neuron of the L-th layer and the k-th neuron in layer $(L-1)$ and $b_j^{(L)}$ is the bias of the j-th neuron in the layer L (see Fig. 4.4).

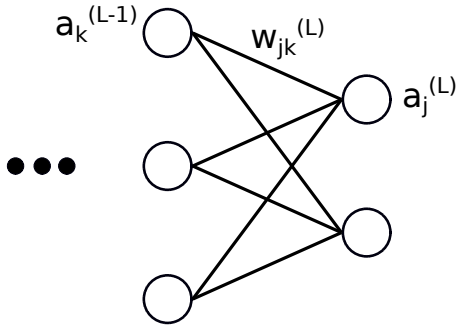


Figure 4.4: Activation of layer L and $L-1$.

This can be expressed more properly by summarizing all activations in the L-th layer and $(L-1)$-th layer, as well as all biases of the neurons in layer $(L-1)$ to one vector each. Additionally, all weights between these two layers can be combined to one weight matrix (see Eq. 4.4).

$$\begin{bmatrix} a_1^{(L)} \\ \vdots \\ a_{N_L}^{(L)} \end{bmatrix} = \varphi\left(\begin{bmatrix} w_{11}^{(L)} & w_{12}^{(L)} & \dots & w_{1N_{L-1}}^{(L)} \\ \vdots & \ddots & & \\ w_{N_L1}^{(L)} & & & w_{N_LN_{L-1}}^{(L)} \end{bmatrix} \begin{bmatrix} a_0^{(L-1)} \\ \vdots \\ a_{N_{L-1}}^{(L-1)} \end{bmatrix} - \begin{bmatrix} b_1^{(L)} \\ \vdots \\ b_{N_L}^{(L)} \end{bmatrix}\right) \tag{4.4}$$

This can be further summarized to

$$\boldsymbol{a}^{(L)} = \varphi\left(\boldsymbol{W}^{(L)}\boldsymbol{a}^{(L-1)} - \boldsymbol{b}^{(L)}\right). \tag{4.5}$$

One possible application for a neural network would be the classification of 64×64 pixel gray-scale image into two classes, class A and B. In this case, each input x_{m} corresponds to a pixel value of one pixel. Hence, the network has 4096 inputs. The output layer consists of one neuron. An output of the output neuron closer to 0 corresponds to an input image of class A, whereas an output closer to 1 corresponds to an input image of class B. The weights and biases of the hidden layers and the output layer are set in a manner that the input images are classified into the desired classes.

4.1.3 Training of Neural Networks

There are more than 200,000 weights and more than 4,000 biases for a fully connected neural network with 64×64 inputs, a hidden layer of 64 neurons and one output neuron. To produce the desired output for a given input, these weights and biases need to be adjusted by a process, which is called "learning".

To achieve learning, labeled training data is needed which represents the desired output for the neural network. For the example above, this could be gray-scale input images of size 64×64 pixels alongside labels which classifies the image into one of two classes. Using this data, the weights and biases are adjusted such that the performance of the neural network on this training data is maximized. A validation data set is then fed into the network. It includes more labeled data which has not been used in the training data set. The performance of the neural network can be assessed based on how many images of the validation data set it classified correctly.

The training process itself is illustrated in Fig. 4.5. In the most basic case, the weights and biases are initialized randomly. The images of the training data are fed into the neural network. A cost function is used to rate how close the outputs of the network are to the desired outputs. Cost functions will be treated in more detail in section 4.1.4. The cost function takes the output of the neural network and the desired output, the labels of the training data, as input. Based on this input, it produces a cost as an output. The higher the cost, the lower the performance of the network. This cost serves as the input for a back-propagation algorithm, which will be presented in detail in section 4.1.5. The algorithm calculates by how much and in which direction each weight and bias of the network needs to be changed to minimize the cost function. Finally, the output vector of the back-propagation algorithm is used to adjust the weights and biases accordingly. Each cycle of this process is called "epoch". A neural network is usually trained for a certain number of epochs.

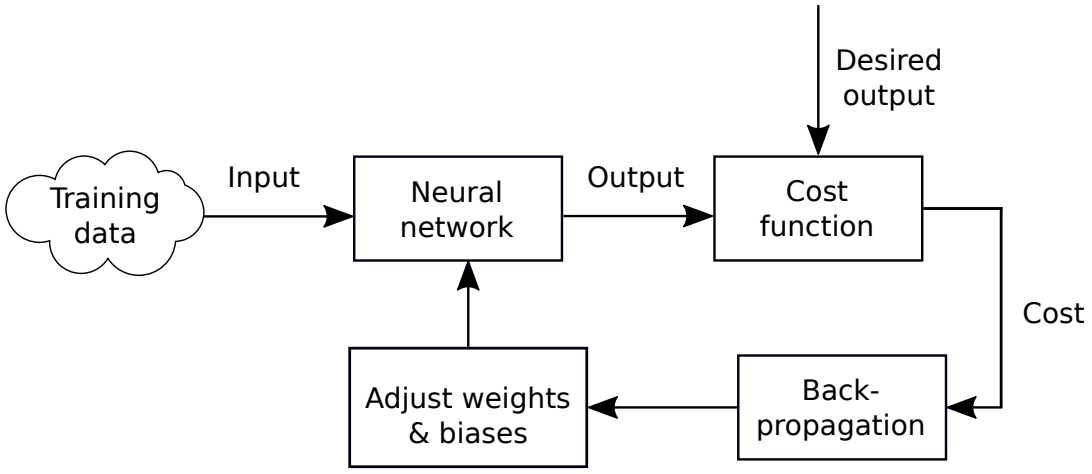


Figure 4.5: Schematic illustration of the training process of neural networks.

4.1.4 Cost Functions

A cost function assesses the accuracy of a neural network's output. It compares the actual output of the neural network with the desired output for the given data and returns a cost. The higher the cost, the lower the neural network performed on producing the desired output. The goal of training a network is to minimize the cost function for all training examples resulting in an increase of it's performance.

The problems to be solved are often multi-dimensional. One cost function which considers deviations in all dimensions to be equal is the Mean Squared Error (MSE) [43, Chapter 1]. It is the mean squared distance of the prediction $\boldsymbol{a}(\boldsymbol{W}, \boldsymbol{b}, \boldsymbol{x}_n)$ from the desired output $\boldsymbol{y}_n$ where N is the total number of samples:

$$\mathrm{C}(\boldsymbol{W}, \boldsymbol{b}) = \frac{1}{N} \sum_{n=1}^{N} \left(\boldsymbol{y}_n - \boldsymbol{a}(\boldsymbol{W}, \boldsymbol{b}, \boldsymbol{x}_n)\right)^2 . \tag{4.6}$$

MSE is best used for continuous problems, e.g. for regressions where the distance between predicted and expected point needs to be minimized. For classification however, other cost functions are more suitable.

Often times a classification of data into one of two classes is needed. These problems are called binary classification problems. These classes can be imbalanced. One cost function optimized for imbalanced binary classification tasks is Dice loss [45].

Dice loss is based on the Dice coefficient. The Dice coefficient is a measure of overlap between predicted and ground truth (see Fig 4.6 and Equ. 4.7). It ranges between 0 and 1, where 0 indicates no overlap between desired truth, also called ground truth, and predicted truth and 1 indicates a full overlap.

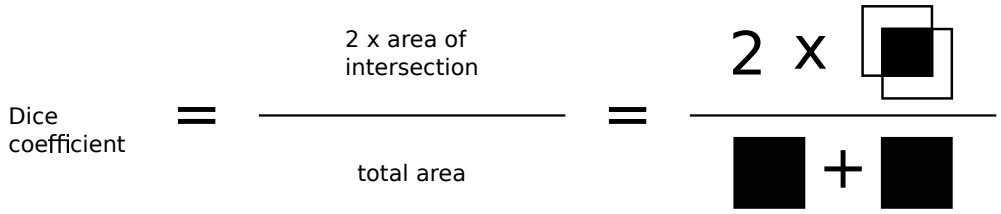


Figure 4.6: Schematic illustration of the calculation of the dice coefficient.

$$\mathrm{DC}\left(\boldsymbol{W}, \boldsymbol{b}\right) = \frac{2 \mid \boldsymbol{y}_n \cap \boldsymbol{a}\left(\boldsymbol{W}, \boldsymbol{b}, \boldsymbol{x}_n\right) \mid}{\mid \boldsymbol{y}_n \mid + \mid \boldsymbol{a}\left(\boldsymbol{W}, \boldsymbol{b}, \boldsymbol{x}_n\right) \mid} \tag{4.7}$$

Cost functions need to be minimizable. To make the optimal solution full overlap, the minimum Dice loss is defined as $\mathrm{C} = 1 - \mathrm{DC}$:

$$\mathrm{C}\left(\boldsymbol{W}, \boldsymbol{b}\right) = 1 - \frac{2 \mid \boldsymbol{y}_n \cap \boldsymbol{a}\left(\boldsymbol{W}, \boldsymbol{b}, \boldsymbol{x}_n\right) \mid}{\mid \boldsymbol{y}_n \mid + \mid \boldsymbol{a}\left(\boldsymbol{W}, \boldsymbol{b}, \boldsymbol{x}_n\right) \mid}. \tag{4.8}$$

4.1.5 Back-Propagation

Back-propagation is short for *backward propagation of error*. It was originally developed to calculate gradients of complex functions. Today, it is one of the most important algorithms in the training of neural networks. The algorithm calculates the gradient of a given error or loss function C with respect to the networks weights and biases. [43, Chapter 5]

The name stems from the method of calculating the gradients of the weights of the last layer first, and the gradients of the first layer weights last. Calculations of the partial gradients of one layer are used to calculate the partial gradients of the previous layer. The backwards propagation of the error calculation allows for an efficient method for the determination of the gradient of the cost function. [43, Chapter 5]

Before formally defining the back-propagation algorithm, the weighted sum of the j-th neuron in the L-th layer is defined as

$$z_j^{(L)} = \sum_{i=1}^{N_{L-1}} w_{ji}^{(L)} a_i^{(L-1)} - b_j^{(L)}. \tag{4.9}$$

For the activation of the same neuron it follows

$$a_j^{(L)} = \varphi\left(z_j^{(L)}\right) \tag{4.10}$$

with φ being the activation function of the neural network.

To simplify the calculations to come, the bias of the j-th neuron in the L-th layer is defined as the weight between the 0-th neuron of the prior layer and the j-th neuron in the L-th layer. Additionally the activation of the 0-th neuron in the $(L-1)$-th layer is set to 1 (See Equ. 4.11), so that

$$-b_j^{(L)} = w_{j0}^{(L)} \text{ with } a_0^{(L-1)} = 1. \tag{4.11}$$

Hence, for the weighted sum of the j-th neuron in the L-th layer it follows

$$z_j^{(L)} = \sum_{i=1}^{N_{L-1}} w_{ji}^{(L)} a_i^{(L-1)} - b_j^{(L)} = \sum_{i=0}^{N_{L-1}} w_{ji}^{(L)} a_i^{(L-1)}. \tag{4.12}$$

As mentioned above, the underlying principle of training a neural network is to minimize the cost function

$$C\left(\boldsymbol{W}, \boldsymbol{X}, \boldsymbol{Y}\right) = \sum_{n=0}^{N-1} C_n\left(\boldsymbol{W}, \boldsymbol{x_n}, \boldsymbol{y_n}\right) \tag{4.13}$$

with respect to the network weights $\boldsymbol{W}$ for N inputs, with $\boldsymbol{X}$ being all inputs, $\boldsymbol{Y}$ being all desired outputs and $\boldsymbol{x_n}$ and $\boldsymbol{y_n}$ being the n-th input and corresponding desired output. The gradient of the error function for a particular input n can be written as a vector containing all the partial derivatives with respect to all weights. For example, for a network with m layers and a number of N_m neurons in the m-th layer the gradient is defined as

$$\nabla C_n = \begin{bmatrix} \frac{\partial C_n}{\partial w_{00}^1} \\ \vdots \\ \frac{\partial C_n}{\partial w_{N_m N_{m-1}}^m} \end{bmatrix}. \tag{4.14}$$

The derivative of the cost function for a certain input n with respect to every weight in the network needs to be determined to calculate ∇C_n. To understand the mechanism behind these calculations, the partial derivative of C_n with respect to the weight w_{ji} in the L-th layer is examined. From section 4.1.4 and equations 4.9 and 4.10 one notices that C_n depends only on the weight $w_{ji}^{(L)}$ through the weighted sum $z_j^{(L)}$. Hence, applying the chain rule it follows for the partial derivative

$$\frac{\partial C_n}{\partial w_{ji}^{(L)}} = \frac{\partial C_n}{\partial z_j^{(L)}} \frac{\partial z_j^{(L)}}{\partial w_{ji}^{(L)}}. \tag{4.15}$$

The first term is usually called "error" for reasons which will be discussed below:

$$\delta_j^{(L)} = \frac{\partial C_n}{\partial z_j^{(L)}} \tag{4.16}$$

The second term can be determined using equation 4.9:

$$\frac{\partial z_j^{(L)}}{\partial w_{ji}^{(L)}} = \frac{\partial}{\partial w_{ji}^{(L)}} \left(\sum_{l=0}^{N_{L-1}} w_{jl}^{(L)} a_l^{(L-1)} \right) = a_i^{(L-1)}. \tag{4.17}$$

Thus, the partial derivative can be written as

$$\frac{\partial C_n}{\partial w_{ji}^{(L)}} = \delta_j^{(L)} a_i^{(L-1)}. \tag{4.18}$$

Therefore, for calculating the partial derivative one needs to multiply the error at the output end of a weight with the activation of the input end of the weight. Hence, to calculate the gradient vector ∇C_n for an input n the errors at each neuron need to be calculated.

In order to calculate the errors for each neuron, one has to start at the output layer of the neural network. Considering a neural network of M layers, the error of the j-th neuron in the last layer is

$$\delta_j^M = \frac{\partial C_n}{\partial z_j^M}. \tag{4.19}$$

Hence, the partial derivative of cost with respect to weight between the j-th neuron in last layer and i-th neuron in first to last layer is

$$\frac{\partial C_n}{\partial w_{ji}^M} = \frac{\partial C_n}{\partial z_j^M} a_i^{M-1}. \tag{4.20}$$

Traveling deeper into the neural network it becomes clear that the errors of a hidden layer L depend on the errors of the layer $L+1$. Hence, using the chain rule for multivariate functions again it follows for all layers $L \leq M$

$$\delta_j^{(L)} = \frac{\partial C_n}{\partial z_j^{(L)}} = \sum_{l=1}^{N_{L+1}} \frac{\partial C_n}{\partial z_l^{(L+1)}} \frac{\partial z_l^{(L+1)}}{\partial z_j^{(L)}}. \tag{4.21}$$

Considering that

$$\delta_l^{(L+1)} = \frac{\partial C_n}{\partial z_l^{(L+1)}} \tag{4.22}$$

equation 4.21 becomes

$$\delta_j^{(L)} = \frac{\partial C_n}{\partial z_j^{(L)}} = \sum_{l=1}^{N_{L+1}} \delta_l^{(L+1)} \frac{\partial z_l^{(L+1)}}{\partial z_j^{(L)}}. \tag{4.23}$$

Using equation 4.9

$$z_l^{(L+1)} = \sum_{j=1}^{N_{L+1}} w_{lj}^{(L+1)} a_j^{(L)} = \sum_{j=1}^{N_{L+1}} w_{lj}^{(L+1)} \varphi\left(z_j^{(L)}\right) \tag{4.24}$$

the second term of equation 4.21 can be written as

$$\frac{\partial z_l^{(L+1)}}{\partial z_j^{(L)}} = w_{lj}^{(L+1)} \varphi'\left(z_j^{(L)}\right). \tag{4.25}$$

Combining equation 4.21 and 4.25, it follows for 4.22

$$\delta_j^{(L)} = \sum_{l=1}^{N_{L+1}} \delta_l^{(L+1)} w_{lj}^{(L+1)} \varphi'\left(z_j^{(L)}\right) = \varphi'\left(z_j^{(L)}\right) \sum_{l=1}^{N_{L+1}} \delta_l^{(L+1)} w_{lj}^{(L+1)}. \tag{4.26}$$

This finally results to a partial derivative of the cost function with respect to a certain weight w_{ji} in the hidden layer L being

$$\frac{\partial C_n}{\partial w_{ji}^{(L)}} = \delta_j^{(L)} a_i^{(L-1)} = \varphi'\left(z_j^{(L)}\right) a_i^{(L-1)} \sum_{l=1}^{N_{L+1}} w_{lj}^{(L+1)} \delta_l^{(L+1)} \text{ for } 1 \leq L \leq M. \tag{4.27}$$

From equation 4.27 the concept of back-propagation becomes clear. To calculate the gradient vector ∇C, the partial derivatives $\partial C_n / \partial w_{ji}$ for all n inputs and all weights need to be determined. For this the errors in the networks final layer are calculated. These errors are used to calculate the errors in the first to last layer. This process is repeated until the first layer is reached.

In other words, the inputs are fed forward into the network until they reach the output layer. The back-propagation algorithm then propagates backwards through the network to determine the errors of all its neurons (see Fig. 4.7). [43, Chapter 5]

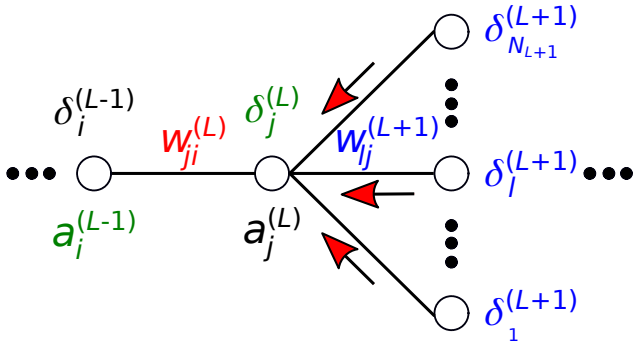


Figure 4.7: Back-propagation of the errors δ through a neural network.

4.1.6 Optimizer

The back-propagation algorithm gives an indication for the amount and direction the weights in the network need to be changed to minimize the cost function. The higher the gradient for a certain weight, the higher the needed adjustment to arrive at the optimal solution. Optimizers are algorithms to execute these adjustments optimally. These adjustments take place after a certain number of input-output pairs are presented to the neural network called "batch". The mean gradient $\frac{\partial C_t}{\partial w_{ji}^{(L)}}$ of the cost function for each weight for a batch t with a certain number of input-output pairs N is taken as input for the optimizer:

$$\frac{\partial C_t}{\partial w_{ji}^{(L)}} = \frac{1}{N} \sum_{n=1}^{N} \frac{\partial C_n}{\partial w_{ji}^{(L)}}. \tag{4.28}$$

Gradient descent is the simplest optimizer. The optimizer adjusts the weights of the network by the calculated gradient of the batch t times a set factor α called learning rate [46] [43, Chapter 5]:

$$\left(w_{ji}^{(L)}\right)_{t+1} = \left(w_{ji}^{(L)}\right)_t - \alpha \frac{\partial C_t}{\partial \left(w_{ji}^{(L)}\right)_t}. \tag{4.29}$$

In this case the learning rate stays the same for each weight. Sometimes however, some gradient information is more important for one weight than for the other. For these cases an individual and adjustable learning rate for each weight would be desirable. Optimizers with this functionality are called adaptive learning rate optimizers.

A common optimizer of this category is the Adaptive Moment Estimation (Adam) optimizer. [46]

The Adam optimizer considers past gradients for a certain weight to adjust its learning rate. For this the exponentially decaying averages of the past gradients m_t and the exponentially decaying averages of the squared past gradients v_t are kept (see Equ. 4.30 and 4.31). [46]

$$m_t = \beta_1 + m_{t-1} + (1 - \beta_1) \frac{\partial C_t}{\partial \left(w_{ji}^{(L)}\right)_t} \tag{4.30}$$

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) \left[\frac{\partial C_t}{\partial \left(w_{ji}^{(L)}\right)_t} \right]^2 \tag{4.31}$$

In machine learning the mean of a set of numbers is often called first moment, whereas the variance is often called second moment. m_t is an estimate for the mean (the fist moment) and v_t is and estimate for the variance (the second moment). Hence, the name for the optimizer. β_1 and β_2 are set numbers close to 1. The authors of the optimizer recommend $\beta_1 = 0.9$ and $\beta_2 = 0.999$. Since v_t and m_t are initialized as vectors of 0 they are biased to stay close to 0. To counteract this bias the values are bias-corrected (see Equ. 4.32 and 4.33). [47]

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t} \tag{4.32}$$

$$\hat{v}_t = \frac{v_t}{1 - \beta_2^t} \tag{4.33}$$

Using these values the update rules for Adam are defined as follows

$$\left(w_{ji}^{(L)}\right)_{t+1} = \left(w_{ji}^{(L)}\right)_t - \alpha \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon} \tag{4.34}$$

where ϵ is a small factor (the authors recommend it to be 10^{-8}) to prevent divisions by 0 and α is a set step size [47].

Like mentioned earlier, the weights of the network are update after each batch. A whole pass through the training set is called epoch. Neural networks are usually trained for several epochs.

4.2 Convolutional Neural Networks

Input variables are often distorted by one or more transformations. For image data these transformations include deformations, scaling or rotation of the target within the image (see Fig. 4.8). In many applications of pattern recognition it is necessary that predictions should be invariant under these transformations. To achieve this, a model is needed whose output stays invariant for the transformed input. One approach is to build invariance properties into the structure of the neural network. This concept is the basis for convolutional neural networks (CNNs). [43, Chapter 5]

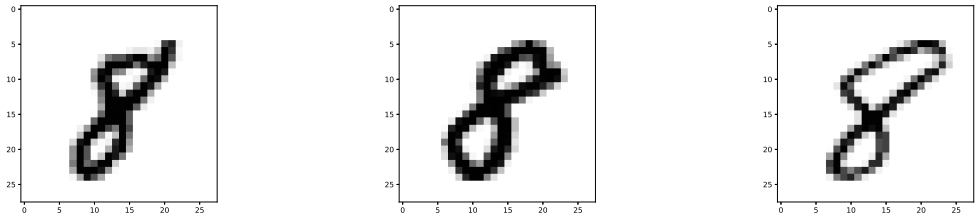


Figure 4.8: Three images of an eight of the MNIST dataset. [48]

Up to now, only fully connected neural networks were considered. For these networks a 2-D gray-scale image is transformed to a 1-D input vector to serve as an input. Naturally, this vectorization leads to the loss of some spatial information. It ignores a key property of images, which is that nearby pixels likely correlate more than distant pixels. Many computer vision technologies exploit this property by extracting local features from small subsections of the image. These local features are then merged to detect higher order features. Ultimately, these yield information about the image as a whole. Convolutional neural networks achieve this by three processes:

- local repetitive fields
- weight sharing
- sub sampling

Convolutional neural networks are organized into planes called feature maps. A subsection of an input image is connected to one unit in the feature map. In Fig. 4.9a, one unit of the feature map is connected to a 3×3 pixel section within the red window of the input image. The window corresponds to a set of weights (see Fig. 4.9b) called kernel. The pixels contained in the window are multiplied with their corresponding weight and summed up. The weighted sum is then fed into the unit in the feature

map and a bias is applied. The weighted sum is used as an input for the activation function. [43, Chapter 5] [49, Chapter 14]

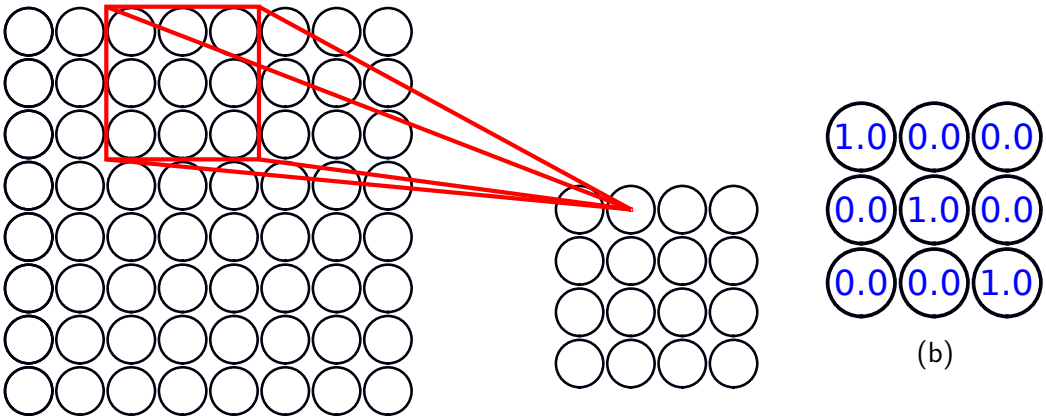


Figure 4.9: Schematic illustration of convolutional neural network (4.9a) with 3×3 Kernel (4.9b).

The same kernel is sled over the whole image (see Fig. 4.10) each time connecting a sub-region of the image to a unit of the feature map. The feature map can be understood as a feature detector, detecting one pattern at different locations of the image. For example, for the kernel depicted in figure 4.9b this would be a diagonal line from the upper left to the lower right corner. Due to the weight sharing of each unit in the feature map, the evaluation of its activations is equivalent to a convolution of the image pixel intensities with a kernel. [43, Chapter 5]

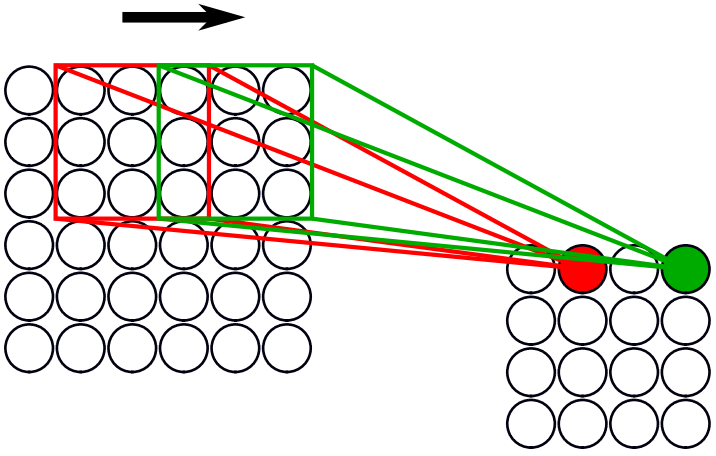


Figure 4.10: Kernel movement with stride of 2 px.

A shift of the input image would only result to a shift of the activations of the feature map while staying structurally the same. This is the basis for the above mentioned invariance to transformations of the input image. In addition to the first convolution

of the input image, convolutional neural networks usually consist of multiple feature maps. [43, Chapter 5]

The outputs of the feature maps serve as inputs the subsampling layer. There again, each unit is connected to a certain sub-region on the feature map via weights defined by a corresponding kernel. Unlike in feature maps, the sub-regions are chosen so that they do not overlap. [43, Chapter 5]

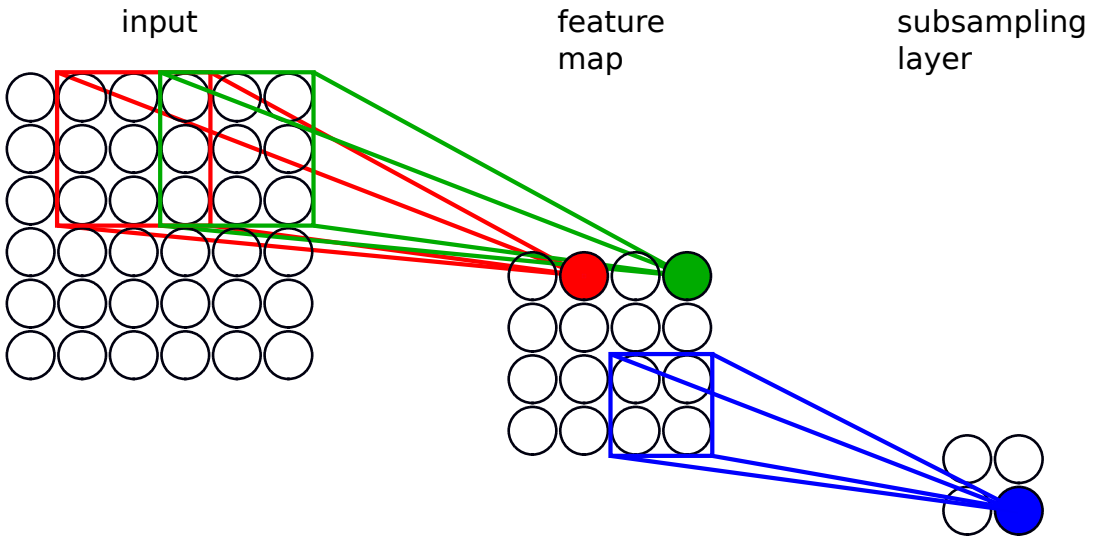


Figure 4.11: Schematic illustration of a convolution followed by a subsampling layer.

Normally, convolutional neural networks consist of several pairs of feature maps (convolutions) and subsampling layers. This leads to a decrease in spatial resolution which is compensated by an increase of the number of features. The last part of a convolutional neural network usually consists of two fully connected layers. These could for example use the extracted features to classify the image into several classes. Convolutional neural networks are trained by a slightly modified back-propagation algorithm. Due to the network structure, the number of overall weights and biases are lower than in fully connected neural networks. The additional constraints on the weights, e.g. due to the use of kernels, further reduces the number of parameters to be learned. This makes it superior to fully connected neural networks when using image data. [43, Chapter 5]

4.2.1 Max-Pooling

Like mentioned above, it is common to periodically insert subsampling layers in-between successive convolutional neural networks. The goal is the progressive reduction of the input's spatial resolution. This lowers the number of parameters to be

updated in the network and decreases the number of computations for the network's training. Eventually, this also controls for overfitting, which reduces the ability of the trained model to generalize to new inputs not included in the training data. [49, Chapter 14]

The most common subsampling layer is the max-pooling layer. It often comes in the form of a 2×2 filter with a stride of 2. It downsamples every slice of the input by 2 along the height and width (see Fig. 4.12). This is handled by the filter by taking the maximum out of 4 values each operation, which results in the discard of $75\,\%$ of activations while keeping the depth dimension the same. [49, Chapter 14]

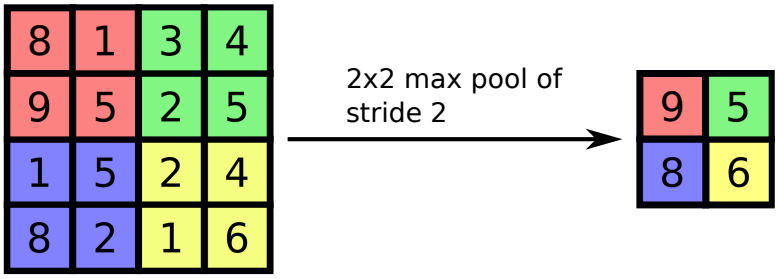


Figure 4.12: Schematic illustration of 2×2 max-pooling operation on a 4×4 matrix. The colored backgrounds indicate the area of each max operation.

4.2.2 Batch Normalization

In deep learning, the input data is usually normalized. This prevents features with a higher range from having a higher impact. The same can also be done for outputs of layers serving as input for hidden layers. In batch normalization, this is done by subtracting the batch mean from the output and dividing this by the mean's standard deviation, which limits the activation range. This leads to

- the prevention of overfitting,
- higher possible learning rates,
- better learning of different features, and
- ultimately the speedup of the training. [50]

4.2.3 Concatenation

For convolutional neural networks it is often desirable to share features between different layers along the network. One way of sharing these is the concatenation layer.

The concatenation takes two or more inputs and concatenates them along one specific concatenation axis (see Fig. 4.13). To accomplish this goal, the inputs need to have the same size except for the concatenation dimension. Assuming two inputs of the size $34 \times 34 \times 12$ and $34 \times 34 \times 6$, a concatenation along the 3rd axis would create an output of the dimension $34 \times 34 \times 18$. [51]

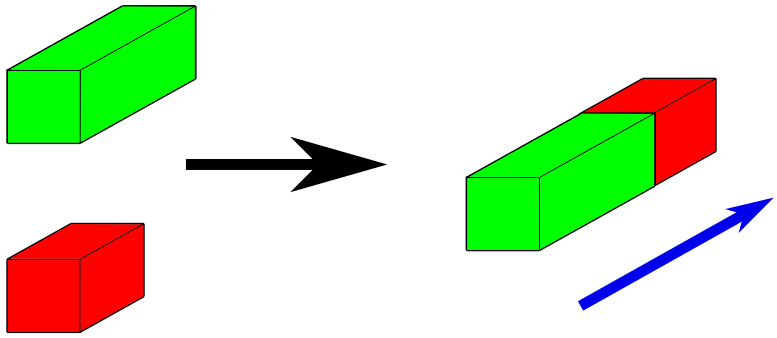


Figure 4.13: Schematic illustration of a concatenation of two layers (red and green) along the concatenation axis (blue arrow).

4.3 Semantic Image Segmentation

Image segmentation means the partition of images into regions of reasonable homogeneous visual appearance or regions corresponding to objects or parts of objects [43, Chapter 9]. For satellite images this could for example mean the partition into regions of cropland, urban areas and forests.

The objective is to extract parts of the images which need further analysis, e.g. the extraction of a road network from a gray-scale satellite image. Optimally, image segmentation produces an output which is more meaningful e.g. a binary map of pixels where white pixels represent roads and black pixels represent background area. [52]

Semantic image segmentation is the pixel-wise labeling of an image. Since the problem is defined at pixel-level, determining the class of a whole image or parts of the image is not acceptable. The localization of the class on the original image pixel resolution is necessary. [52] [53]

CNNs had various successes in challenges of semantic segmentation [54] [55]. CNNs in semantic image segmentation take images as input and produce a map of the same dimensions as an output. Each pixel of the output is assigned to one class. CNNs are provided with image-mask pairs to be trained for this task. In a two-class problem, the mask is a binary pixel map indicating which pixel in the image belongs to which

of the two classes. The mask is also called "ground truth". The ultimate goal of the training is to enable the model to produce a pixel map similar to the ground truth from a given input image.

4.4 Performance Assessment

Trained networks can not only be assessed by the cost function. There are several additional ways to assess the performance of a neural network. The following section will discuss some of these for binary classification models. The following two cases will be discussed: First, the case of balanced classes where class **A** and **B** take up roughly 50 % each for each instance. Second, the case of imbalanced classes where e.g. class **A** represents more then 90 % of each instance. The kind of assessment in binary classification problems is essentially based on the balance of the two classes. For balanced classes, it is common to use measures based on the confusion matrix. In cases where classes are highly imbalanced the Intersection over Union (IoU) is often used.

4.4.1 Confusion Matrix

Many measures of the performance of binary classification algorithms are based on the confusion matrix (see Tab. 4.1). The matrix summarizes the number of true positives (TP), false positive (FP), true negatives (TN) and false negatives (FN). In the case of a binary classification of elements into class **A** (positive or P) and class **B** (negative or N), a false positive would mean an element being classified as class **A** even though it belongs to class **B**. A true positive however, means an element which is classified correctly as class **A**.

Table 4.1: Structure of a confusion matrix. [56]

True class	Positive	Negative
Yes	TP	FP
No	FN	TN
	TP + FN = P	TN + FP = N

Many measures of the assessment of neural networks are based on the confusion matrix. Some examples are:

$$\text{accuracy} = \frac{TP + TN}{P + N} \tag{4.35}$$

$$\text{precision} = \frac{TP}{TP + FP} \tag{4.36}$$

$$\text{recall} = \frac{TP}{P} \tag{4.37}$$

$$\text{false alarm rate} = \frac{FP}{N} \tag{4.38}$$

These measures work well for balanced classes. However, problems arise for imbalanced ones. Table 4.2 shows two confusion matrix of different classifiers. Both have the same precision and recall. Still, both classifiers show different classifier behaviors. The left one shows a weak positive recognition rate and a strong negative recognition rate. The right one shows the opposite behavior. The accuracy is generally a good measure for the performance of classifiers. It fails however, for class imbalances.[56] This is especially true for semantic image segmentation. Classes in semantic image segmentation are usually highly imbalanced and most popular challenges do not use these metrics. Alternatively, IoU is often used in these challenges which looks at the similarity and diversity of sample sets. [53]

Table 4.2: Confusion Matrix for two classifiers with same precision and recall but different classifier behaviors. [56]

True class	Positive	Negative
Yes	200	100
No	300	400
	P=500	N = 500

True class	Positive	Negative
Yes	200	100
No	300	0
	P=500	N=100

4.4.2 Intersection over Union

Intersection over Union (IoU) is common measure of performance of binary classifiers for imbalanced classes. In semantic image segmentation it is the ratio of intersection between the pixel-wise predicted classification result with the mask or ground truth, to their union (see Fig. 4.14). In other words, it is the number of pixels prediction and ground truth share over pixels which are included in prediction and ground truth (see Fig. 4.14). [53]

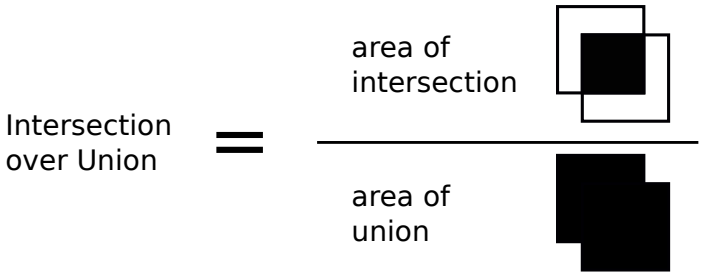


Figure 4.14: Schematic illustration of the calculation of the Intersection over Union.

or in more precise terms, similar to the Dice loss described in section 4.1.4

$$\begin{aligned} \mathrm{IoU} &= \frac{|\, \boldsymbol{y}_n \cap \boldsymbol{a}\,(\boldsymbol{W}, \boldsymbol{b}, \boldsymbol{x}_n)\,|}{|\, \boldsymbol{y}_n \cup \boldsymbol{a}\,(\boldsymbol{W}, \boldsymbol{b}, \boldsymbol{x}_n)\,|} \\ &= \frac{|\, \boldsymbol{y}_n \cap \boldsymbol{a}\,(\boldsymbol{W}, \boldsymbol{b}, \boldsymbol{x}_n)\,|}{|\, \boldsymbol{y}_n\,| + |\, \boldsymbol{a}\,(\boldsymbol{W}, \boldsymbol{b}, \boldsymbol{x}_n)\,| - |\, \boldsymbol{y}_n \cap \boldsymbol{a}\,(\boldsymbol{W}, \boldsymbol{b}, \boldsymbol{x}_n)\,|} \end{aligned} . \tag{4.39}$$

The IoU produces a number between 0 and 1. The higher this number, the better the prediction agrees with the ground truth. [53] An IoU score of greater than 0.5 is normally considered a "good" prediction [49, Chapter 14].

4.4.3 Cross-Validation

In the realm of machine learning, especially in deep learning, a large amount of training data is needed. Oftentimes however, the number of training data is limited. Splitting the available data into training data and validation data according to the usual scheme often leads to overfitting. This makes it necessary, to set aside a number of images for the final evaluation of the model called test dataset. This however, leads to a further reduction of the little available training or validation data. To build a good model, it is desirable to have as much data available for training as possible. Similarly, reducing the validation set will give a noisy prediction of the predictive performance of the model. To overcome this, k-fold cross-validation can be used.

K-folding is essentially taking the available data and partitioning it into k folds. In the simplest case, all k folds have the same size. One fold is used for validation while $k-1$ folds are used for training. This process is repeated for all k possible combinations (see Fig. 4.15). In k-fold cross-validation the average score of the k splits is then taken to assess the performance of the model. [43, Chapter 1]

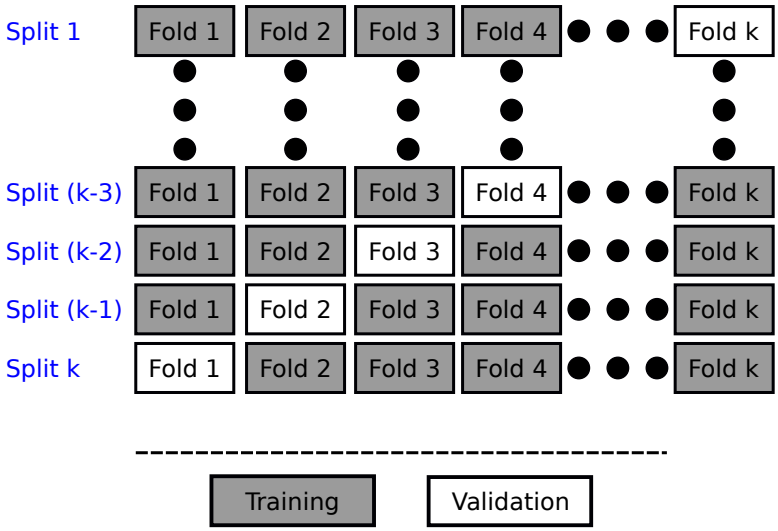


Figure 4.15: Schematic illustration of k-fold cross validation. The grey boxes represent the training data whereas the white represent the validation data in each split.

Chapter 5

Methods

In this chapter, all methods used to obtain the results discussed in Chapter 6 are described. It begins with the acquisition and pre-processing of data, followed by the preparation of the data. The chapter is concluded with the design and training of the model.

5.1 Data Acquisition and Pre-Processing

For the training of a neural network for image segmentation, training data is needed. To the knowledge of the author, there is no open-source training dataset available which would be well suited for the training of a model to detect pipelines on satellite images. Thus, a new dataset had to be created from scratch. In the following, the acquisition of the necessary raw data needed is described. Further, the pre-processing methods used to make this raw data readable for the model described later in this chapter.

5.1.1 Pipeline Course and Construction Date

For the creation of a training and validation dataset, geolocalized data on the course and dates of construction of pipelines are needed. The electricity and gas utility company *National Grid* operates the gas pipeline network in Great Britain. On its website geolocalized data on its pipelines is provided as a shapefile [57] (see Fig. 5.2). Construction dates can be found in the open-source database created by the non-governmental organization *Global Energy Monitor* [58]. To test the performance of the trained algorithm, georeferenced data is required. This data is obtained from

OpenStreetMap (OSM), a collaborative open-source project for the creation of a free editable map of the world [59]. OSM data can be downloaded, filtered, and converted to GEOJSON with the Python library esy-osmfilter [60]. The NEL (Nordeuropäische Erdgasleitung) pipeline in North Germany is chosen as a test case [61]. It has a diameter of 1.42 m, a total length of 441 km, and delivers 531 GW h of natural gas annually (see Fig. 5.1) [62]. Some information about its construction specifications can be found in this press article [63].

Figure 5.1: Course of NEL pipeline (red) in Northern Germany. [64]

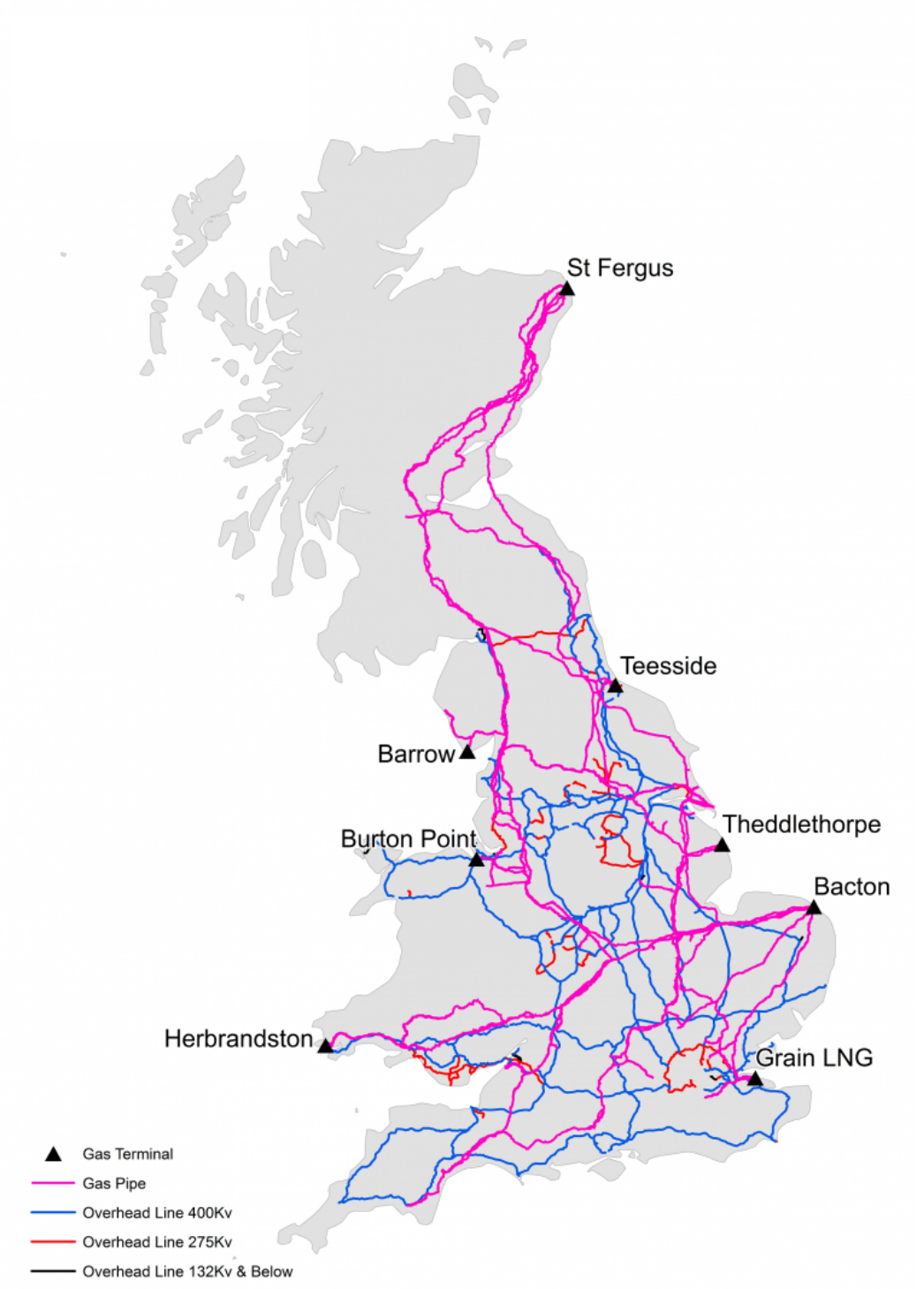


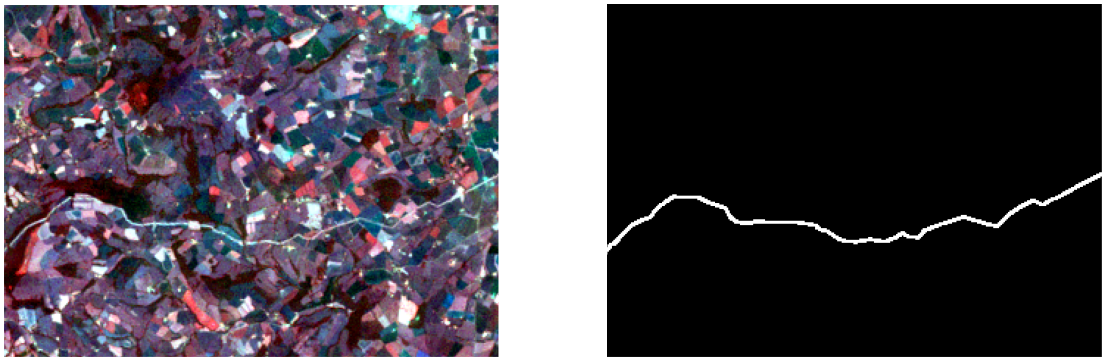
Figure 5.2: Map of energy infrastructure of National Grid UK. Gas pipelines are depicted in purple. [65]

The geolocalized data is loaded into Google Earth Engine (GEE) [66]. GEE is a platform for the processing of satellite imagery. Landsat 5 Tier 1 Surface Reflectance images of the construction dates of each pipeline in the dataset are loaded. The images are cropped down to an area of interest. The NDVI (see Chapter 3) is calculated. Finally, the images containing the bands 1, 2, 3, 7, and the NDVI are

downloaded in the Tagged Image File Format (TIFF). The course of the pipeline on each image is downloaded from GEE.

5.2 Data Preparation

A ground truth is needed to train an image segmentation algorithm using Landsat images. The ground truth is generated using the GEOJSON data on the pipeline in the following way: An all-black TIFF-image of the same size as the satellite image of the pipeline is created. The course of the pipeline is projected onto the raw image. The GEOJSON data has no information about the width of the pipeline construction site. Hence, each pixel in the range of 30 m around the course of the pipeline is tagged as a pixel containing the pipeline. The remaining pixels are tagged as background (See Fig. 5.3).



(a) Image (b) Mask

Figure 5.3: Raw satellite image and corresponding mask.

All bands of the image are normalized to a range from 0 to 1. For bands 1, 2, 3 and 7 the factor 0.0001 provided by the USGS is used [67]. The NDVI is normalized to a range from -1 to 1 by (see Chapter 3):

$$\mathrm{NDVI}_{\mathrm{norm}} = \frac{\mathrm{NDVI} + 1}{2}. \tag{5.1}$$

Pixels with invalid values, which are most likely caused by oversaturation, are replaced with the value 0. Finally, the processed images and corresponding masks are cropped into image-mask pairs of the dimensions 64×64 px (see Fig. 5.4). Only image-mask pairs are kept which contain more than 50 px tagged as pipeline. The full process of training data processing can be seen in Fig. 5.5.

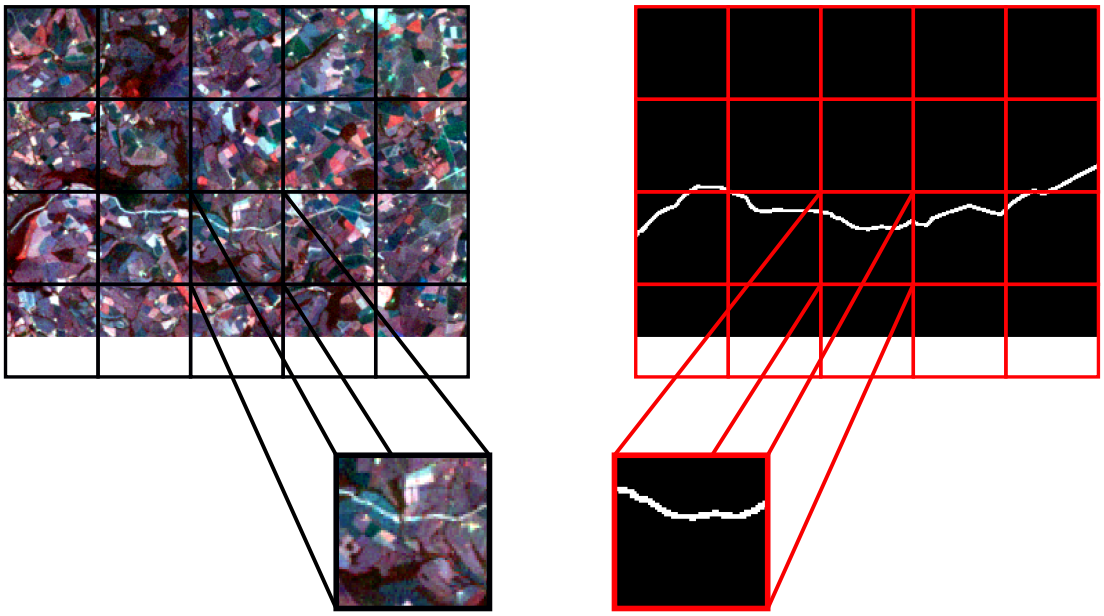


Figure 5.4: Schematic illustration of the cutting process to create image-mask pair. The left side shows the satellite image. The right side shows the corresponding mask.

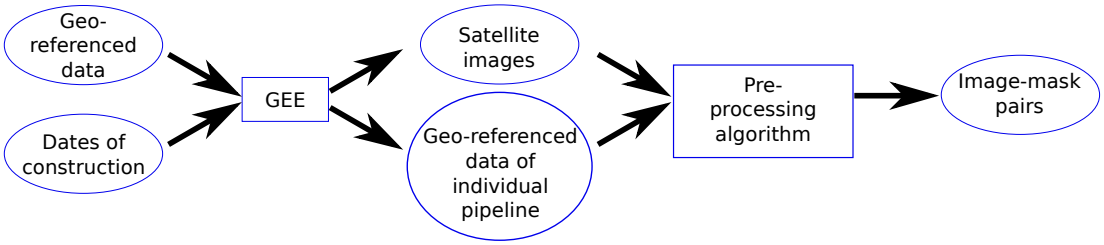


Figure 5.5: Process of training data generation with the raw data as input and the image-mask pairs as output.

5.3 Model Design and Training

As mentioned in section 4.3, CNNs had great success in various image segmentation challenges in the last years. One particular CNN architecture is U-Net which is also the foundation of the model used in this thesis [68]. The architecture was originally developed for multi-class image segmentation of biomedical images. However, its application left its original realm and was successfully applied to different image segmentation tasks [69] [70].

A schematic illustration of the model used in this thesis can be seen in Fig. 5.6. The architecture takes 64×64 px images with five channels (one for each band) as input. It yields a map of the same dimensions with one channel. The model is, like the name suggests, u-shaped. It can be divided into two parts: a contracting part (left) and an expansive part (right). The left part is the downsampling part, also called the

encoder. It is a convolutional network for the increase of feature information and the reduction of spatial information. This is achieved by a series of convolutions, followed by a ReLU layer and a max-pooling operation. The right part is the upsampling part, also called the decoder. Its goal is the combination of spatial and feature information through a series of up-convolutions.

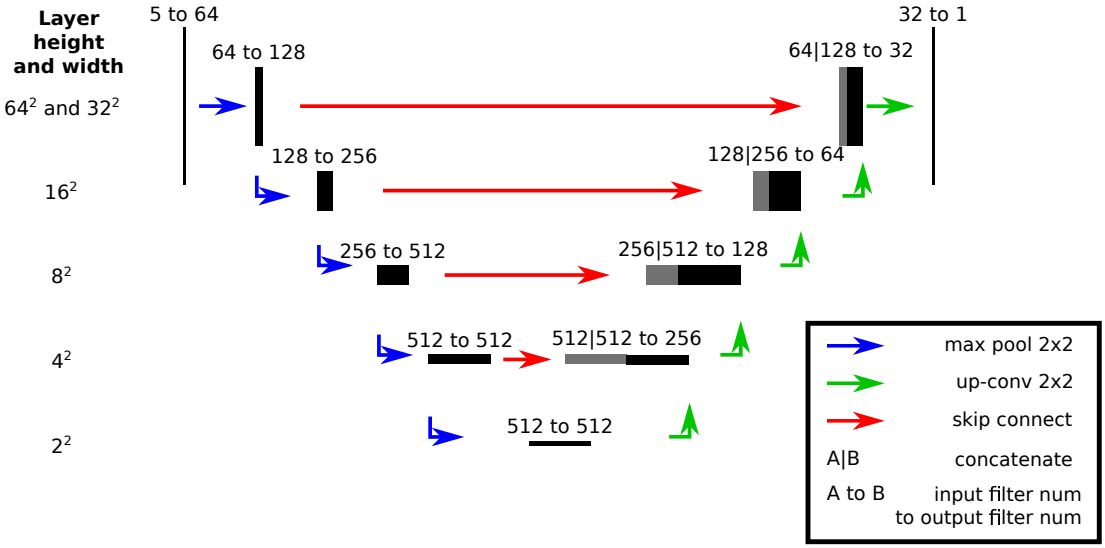


Figure 5.6: Schematic illustration of the u-net-based model used in this thesis. The left part shows the encoder reducing the input image size from 64×64 px to 2×2 px. The right part shows the decoder increasing the image size to 64×64 px again.

5.3.1 Encoder

The encoder can achieve its goal, the increase of feature information, by a series of 3×3 convolutional layers with ReLU activation (black boxes). After each series of convolutions the number of filter layers is increased, e.g. in the first step the dimensions of the image change from 5 to 64. Each series of convolutions is followed by a 2×2 max-pooling layer which reduces the image size by $75\,\%$. This leads to a reduction of the image size between the first and second step from 64×64 px to 32×32 px. The encoder reduces the spatial resolution from 64×64 px in the input layer to 2×2 px in the last layer. At the same time, the number of filter layers is increased from 5 to 512.

5.3.2 Decoder

As mentioned before, all the spatial information is lost at the last step of the encoder. The goal of the decoder is the reconstruction of the spatial information from the previously learned features. This is done by a series of 3×3 convolutions with

intermediate batch normalization layers, followed by 2×2 upsampling layers. This reduces the feature layers from 512 in the last layer of the encoder to 1 in the last layer of the decoder. Simultaneously, the spatial dimensions are increased from 2×2 px to 64×64 px again. The batch normalization layers further speed up learning.

5.3.3 Skip Connections

Skip connections execute a concatenation between distant layers. Implemented symmetrically between encoder and decoder, they help to share fine-grained feature information between encoder and decoder. This supports the decoder with detailed spatial information to better reconstruct the image from the learned features. The skip connections are indicated with the red arrows in Fig. 5.6.

5.3.4 Training

The model was implemented using the Python library *Segmentation Models* which itself is based on the machine learning and neural network libraries *Tensorflow* and *Keras*. The problem at hand can be considered as a semantic image segmentation problem. Hence, Dice loss is chosen as the cost function. The model is trained for four different learning rates: $1e{-}3$, $1e{-}4$, $1e{-}5$ and $1e{-}6$. The available data is limited to approx. 300 image-mask pairs. Thus, 5-fold cross-validation is used (see Fig. 5.7). Each run of each cross-validation is tested on a separate test dataset.

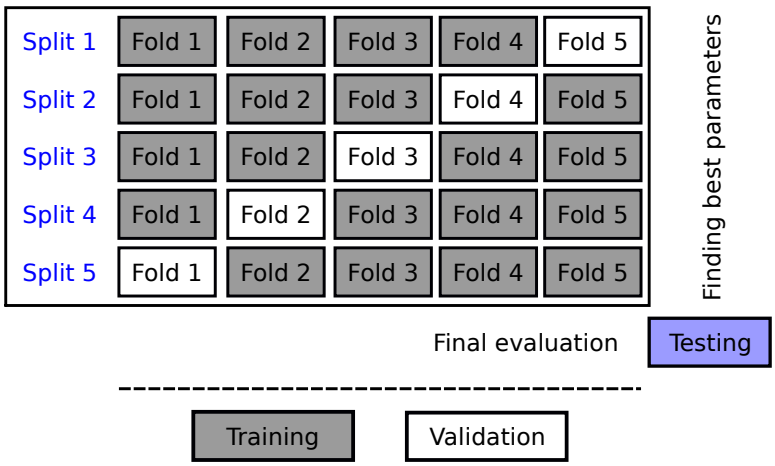


Figure 5.7: Schematic illustration of training process using 5-fold cross-validation and subsequent testing on a separate dataset.

Additional augmentation is used on the training data. This further increases the number of available training examples. Five different augmentations were used on

the image-mask pairs (see Fig. 5.8):

- a horizontal flip
- a vertical flip
- a rotation of 90°
- a transpose or a
- grid shuffle (see Fig. 5.9)

Each augmentation is applied with a probability of 50 %. The application of multiple augmentations is possible.

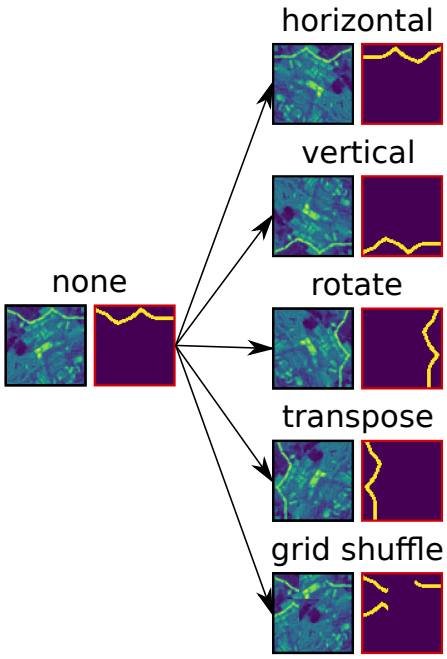


Figure 5.8: Overview of five augmentations used on the image-mask pairs during the training process.

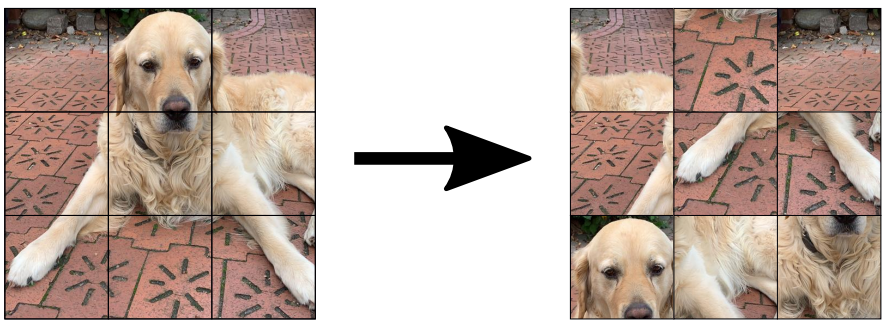


Figure 5.9: The principle of grid shuffling, demonstrated with a picture of my dog "Sam".

The model is trained for a maximum of 200 epochs. The learning rate is reduced by a factor of 0.1 if the validation IoU does not change for 15 epochs. The training is stopped if the validation IoU does not change for 20 epochs. The model with the best validation IoU score of each cross-validation run is saved.

Chapter 6

Results and Discussion

The learning rate has a high impact on the training of neural networks. Learning rates which are too small result in slow training or non-convergence of the loss. However, learning rates that are too high lead to high fluctuations of the loss in training which again can cause non-convergence. To investigate the influence of the learning rate on the training of the neural network at hand the network is trained with different learning rates. The goal is to find the learning rates at which the network training still converges and a smooth training process is ensured.

The model was trained with learning rates of $1\mathrm{e}{-03}$, $1\mathrm{e}{-04}$, $1\mathrm{e}{-05}$ and $1\mathrm{e}{-06}$. The dataset described in chapter 5 was used with 5-fold cross-validation. Each split is trained for a maximum of 200 epochs. The learning rate was further reduced by a factor of 0.1 if the maximal validation IoU did not change for 15 epochs. The training was stopped if the maximal validation IoU did not change for 20 epochs. The weights of the maximal validation IoU of each split are saved. The maximal validation IoU and minimal validation loss of each split were averaged for each learning rate.

The mean maximal validation IoUs and minimal validation losses for each learning rate can be seen in Fig. 6.1. The training and validation IoU scores of each training epoch of the second split are plotted over the number of epochs alongside the mean maximal validation IoU score of the corresponding learning rate in Fig. 6.2a to 6.2d. Additionally, the training and validation losses of each training epoch of the second split are plotted over the number of epochs alongside the mean minimal validation loss of the corresponding learning rate in Fig. 6.3a to 6.3d.

The mean maximal validation IoU scores and minimal validation losses of the learning rates of $1\mathrm{e}{-03}$ and $1\mathrm{e}{-04}$ both have similar values within the standard deviation of one another. The mean max. val. IoU score and the mean min. val. loss for $1\mathrm{e}{-05}$ are slightly lower or higher, respectively. This indicates the convergence of these

measures during the training processes for $1e{-}03$ and $1e{-}04$ and training processes close to convergence for $1e{-}05$. Figure 6.2a to 6.2c confirm this assumption despite showing high fluctuations of the validation IoU score between each training epoch. The validation loss was also subject to high fluctuations for these learning rates (see Fig. 6.3a to 6.3c). These fluctuations can be explained by the relatively low number of training data of 375 image-mask pairs compared to e.g. the Massachusetts Roads dataset containing more than 1000 image-mask pairs 1000×1500 px each [71]. Figure 6.2d and 6.3d show smaller fluctuations but no trend of convergence. The mean maximum validation IoU score of the learning rate $1e{-}06$ is only $8.4\,\%$ of the mean maximum validation IoU score of the learning rate of $1e{-}05$. The mean minimum validation loss of learning rate $1e{-}06$ is 4.6 times higher than the one of the learning rate $1e{-}05$. This, in combination with the small standard deviation, suggests non-convergence for all splits for the learning rate of $1e{-}06$.

The authors of the Adam optimizer recommend an initial learning rate of $1e{-}03$ [47]. The training processes are still able to converge for learning rates of $1e{-}04$ and $1e{-}05$. For a learning rate of $1e{-}06$ a convergence is not possible. The model was trained for images from Great Britain. For the detection of pipelines in Europe, a similar performance of the model for images of all counties in Europe would be desirable.

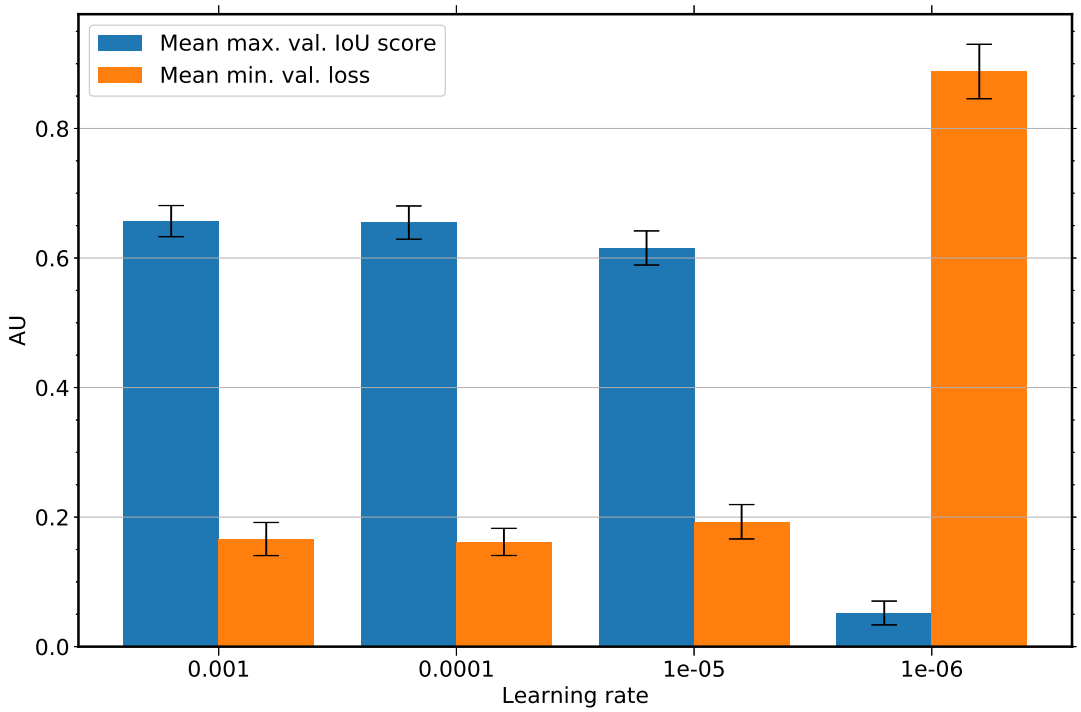
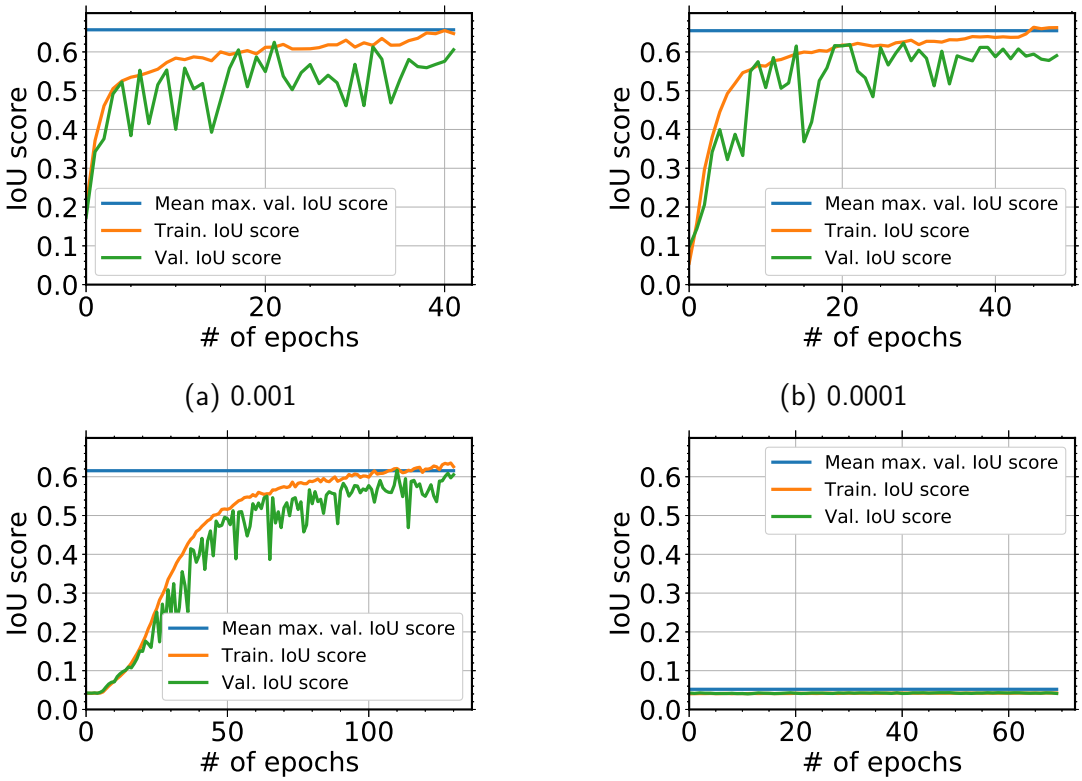


Figure 6.1: Minimum loss and maximum IoU score averaged over all training splits for each learning rate with standard deviation.



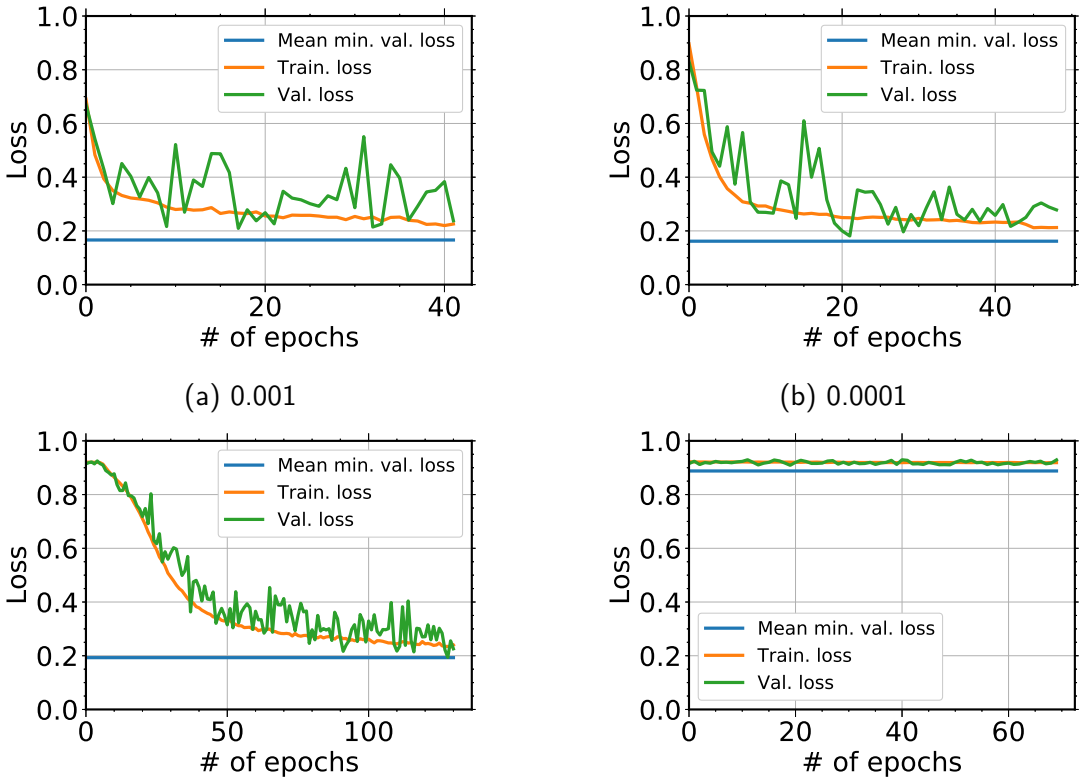
(a) 0.001

(b) 0.0001

(c) 1e-05

(d) 1e-06

Figure 6.2: IoU over number of epochs for split 2.



(a) 0.001

(b) 0.0001

(c) 1e-05

(d) 1e-06

Figure 6.3: Loss over number of epochs for split 2.

To test the ability of the model to generalize to unknown data, all fully or nearly converging models were tested with a dataset of images of the NEL pipeline in Northern Germany. The dataset consisted of 37 images of the same properties as the training dataset. Each model trained on each split for each learning rate was evaluated on the test dataset. The IoU scores and the losses were averaged for each learning rate (see Fig. 6.4). The highest mean IoU and lowest mean loss of 0.57 ± 0.01 and 0.31 ± 0.01 respectively were measured for the learning rate of $1e{-}04$. The lowest IoU of 0.48 ± 0.03 and highest loss of 0.38 ± 0.03 were reached for the learning rate of $1e{-}05$. The best performing split of the learning rate of $1e{-}04$ was found to be split 5 with an IoU score of 0.58 and a loss of 0.29. This makes it still performing 0.07 worse in terms of IoU score than the mean maximum validation IoU score during training. To get a better picture of possible causes for that, the performance of the trained model for the best performing split (5th split of learning rate of $1e{-}04$) was further visually analyzed.

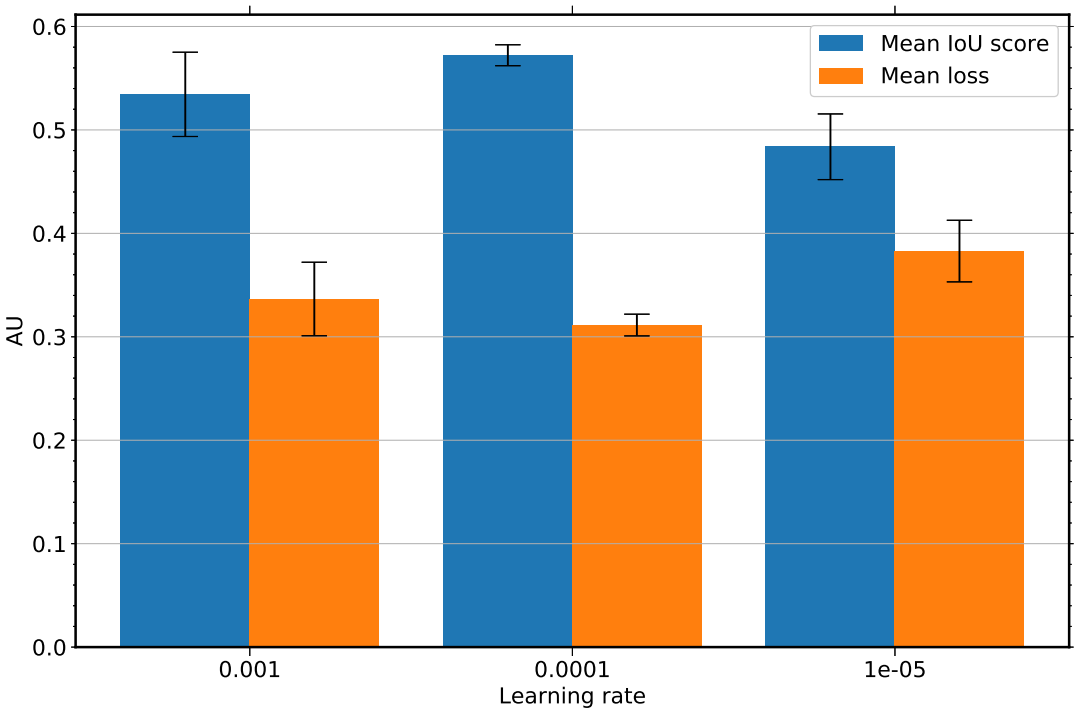


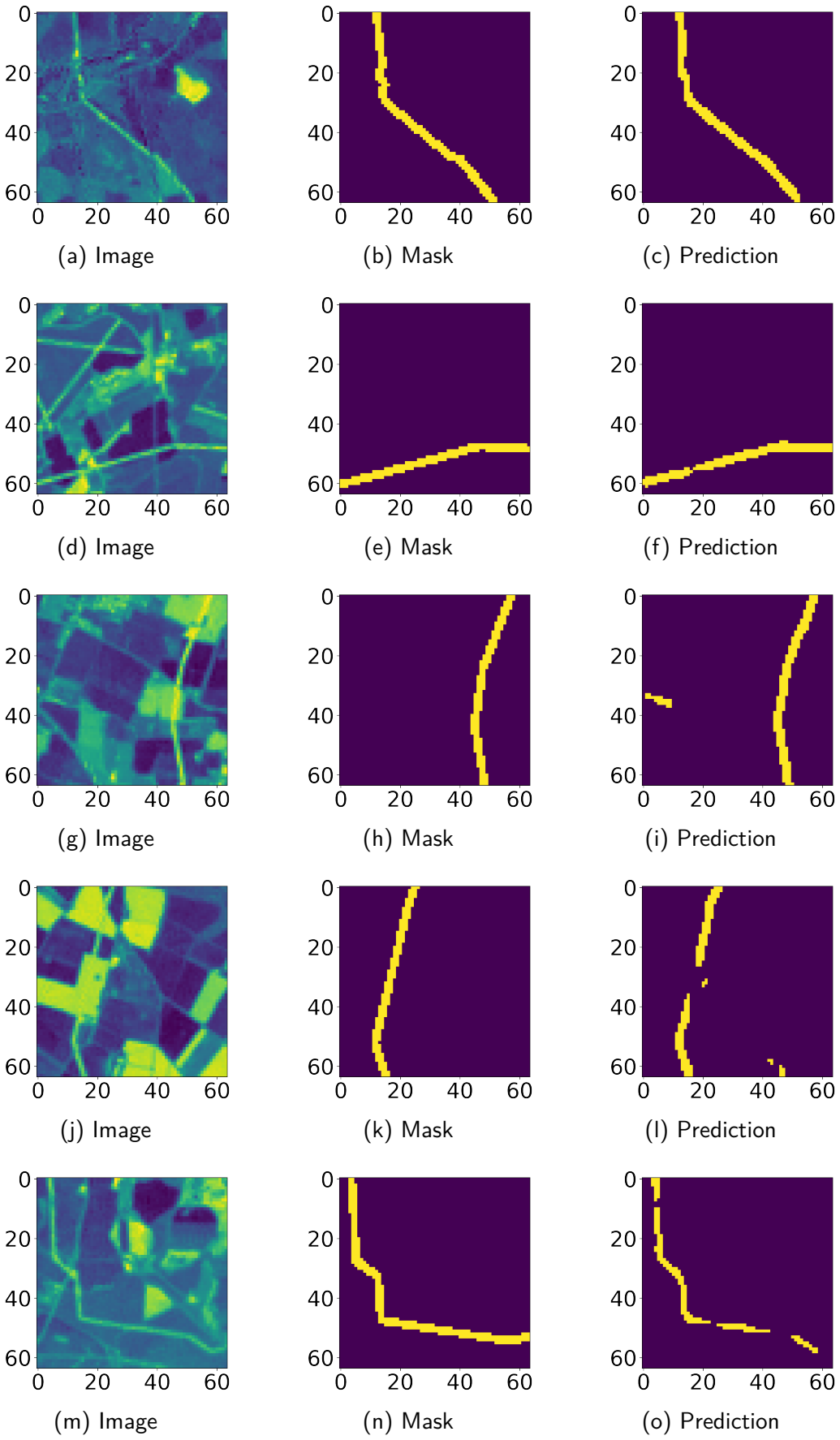
Figure 6.4: Mean loss and IoU score of trained models applied to test dataset for each learning rate with standard deviation.

The visual performance assessment was first done on the validation dataset of the 5th split of the $1e{-}04$ learning rate. This assessment was repeated on the test dataset unknown to the model. Figure collection 6.5 shows a collection of input images, the corresponding mask, and the prediction of the model. For the predictions, only pixels above $90\,\%$ certainty are shown. Figure collection 6.6 shows images, masks, and predictions in the same manner for the test dataset.

Fig. 6.5a to 6.5c show the performance on input images with a low density of pipeline-like structures. The model can predict the pipeline course with high accuracy. Fig. 6.5d shows an input image with a high number of pixels belonging to infrastructure which is visually similar to the pipeline course depicted in 6.5e. The model is still able to accurately predict the course of the pipeline without incorrectly labeling the infrastructure as pipeline. This, however, is not always the case. Fig. 6.5i shows the prediction of the pipeline course for Fig. 6.5g. Parts of the image that are visually close to the pipeline are incorrectly labeled as pipeline. However, it is not conclusive why only this small part is labeled as pipeline and not areas which are similar. Fig. 6.5l shows the opposite case. Parts of the image shown in Fig. 6.5j are falsely labeled as background. This, however, can be explained by the visual similarities between the pipeline course and the background. The model is also able to predict pipeline courses which are wrongly annotated (see Fig. 6.5m to 6.5o). This also means the model was punished for correct predictions in the training process which reduced the

IoU score.

The trained model behaves similarly with the test dataset as input. In cases of backgrounds with little pipeline-like structures, the model reliably predicts the pipeline course (see Fig. 6.6c). This is also true for backgrounds, which contain visually similar structures like the one depicted in Fig. 6.6d. However, this is not always the case. In Fig. 6.6i a road and parts of a field path were falsely labeled as pipeline. Falsely negative labeling also took place. Similarly to the validation dataset, the model had problems labeling pipeline courses with low contrast to the background (see Fig. 6.6j and 6.6l). As in the validation dataset, not all images were labeled correctly. The model was still able to correctly predict the course of the pipeline for these images like can be seen in Fig. 6.6m to 6.6o.



(a) Image (b) Mask (c) Prediction

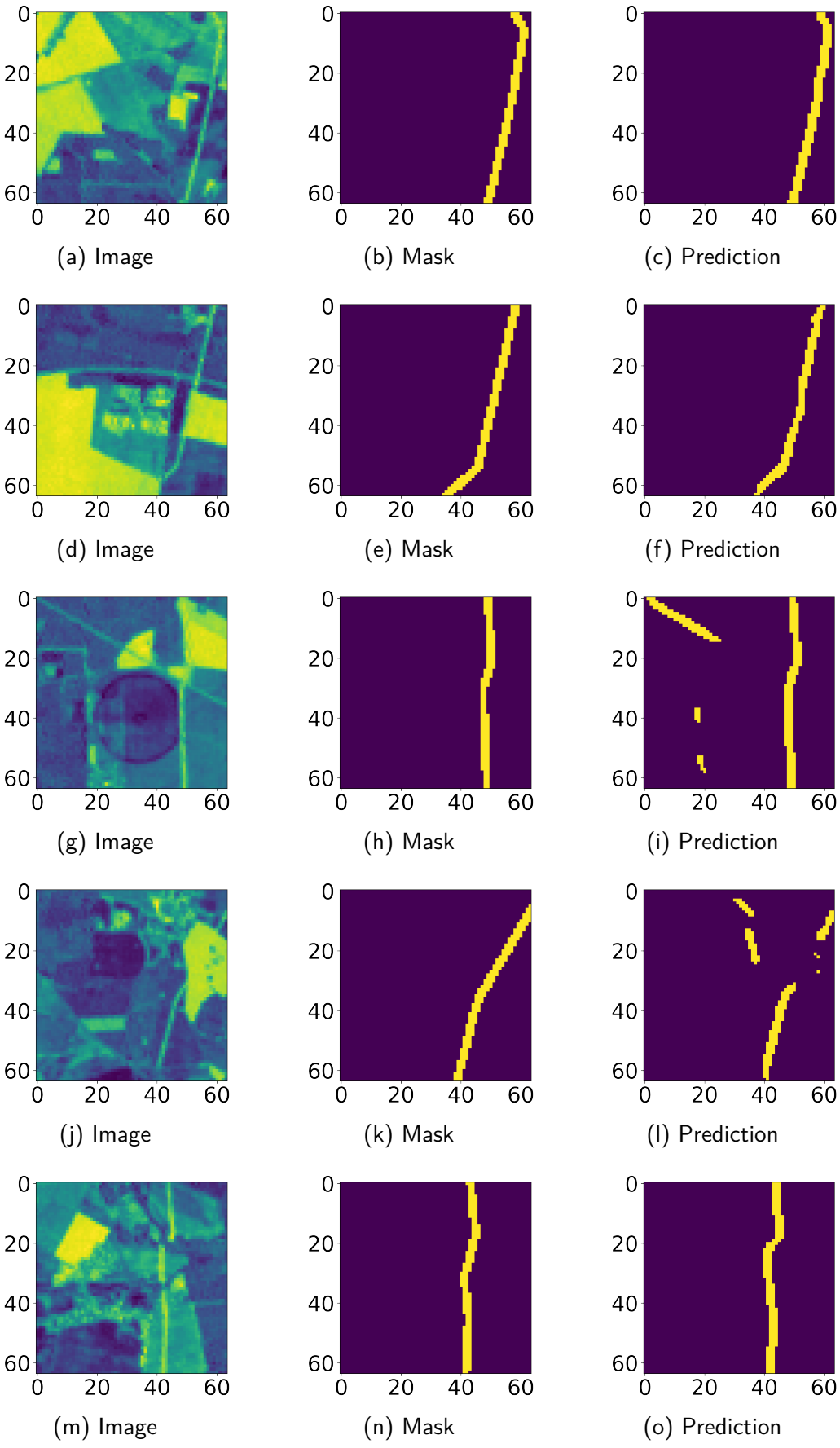
(d) Image (e) Mask (f) Prediction

(g) Image (h) Mask (i) Prediction

(j) Image (k) Mask (l) Prediction

(m) Image (n) Mask (o) Prediction

Figure 6.5: A selection of predictions of the trained model on the validation dataset.



(a) Image (b) Mask (c) Prediction

(d) Image (e) Mask (f) Prediction

(g) Image (h) Mask (i) Prediction

(j) Image (k) Mask (l) Prediction

(m) Image (n) Mask (o) Prediction

Figure 6.6: A selection of predictions of the trained model on the test dataset.

The lower IoU score can be attributed to multiple factors. The geo-referenced data for the test dataset originated from OpenStreetMap and not from the TSO operating the pipeline. This could have lead to more errors in the labeling process, which resulted in incorrect ground truths. Since the ground truths are the basis for the assessment of the model's performance, this could be the reason for the IoU being lower for the test dataset. Another factor could be attributed to the different regions of the training and test dataset. The course of a pipeline appears visually similar throughout Europe. The background, however, is subject to sometimes dramatic changes due to e.g. the change of vegetation area, infrastructure, or type of agriculture. Since both datasets are located in different regions of Europe new background features were introduced with the test dataset. The model was not trained on the test dataset and was therefore not able to learn those features. This resulted in more false positive labels and ultimately in a lower IoU score.

From the results, it can be concluded that the extraction of pipeline courses from satellite images using a u-net-like deep learning model is possible. The model was successfully trained (IoU > 0.5) with initial learning rates of $1\mathrm{e}{-03}$, $1\mathrm{e}{-04}$, and $1\mathrm{e}{-05}$. The training produced a mean maximum validation IoU score of 0.66 ± 0.02, 0.65 ± 0.03 and 0.61 ± 0.03 for these learning rates respectively using a 5-fold cross-validation. All training runs had high fluctuations of the validation loss and IoU score between each training epoch. This can be explained by the low number of training data. Poorly labeled data could also have influenced the training process. The models trained with the learning rate $1\mathrm{e}{-04}$ performed best on a dataset unknown to the model. The evaluation produced an IoU score of 0.57 ± 0.01 which is 0.08 lower than the mean maximum validation IoU for this learning rate. A visual performance assessment led to the conclusion that this difference is caused by two main factors. The test data was labeled differently which might have caused ground truths of lower quality for this dataset. Additionally, the data was collected in different regions which introduce new background features the model could not be trained on.

The resolution of the images could also play a significant role in the performance of the model. This could mean better performance of a model trained with images of higher resolution. Thus, pipelines build after the launch of Landsat 4 in 1982, which had the same sensor set as Landsat 5, could be detected on satellite images. However, prior Landsat satellites had a maximum resolution of $80\,\mathrm{m\,px^{-1}}$, which is greater than the maximum width of the right of way in pipeline construction. Hence, pipelines build before 1982 could be difficult to detect on satellite images.

The models were only trained with images of gas transmission pipelines which have a diameter of at least 45 cm, and tested with data of the NEL pipeline which has a

diameter of 141 cm. The pipelines on the satellite images are detected through their right of way created during construction. The width of the right of way scales with the diameter of the pipe. Thus, pipelines of smaller diameter e.g. in other countries or of distribution networks could be hard to detect on satellite images.

In conclusion, it can be noted that the process has several downsides. Although it is quite accurate on low noise data it has some difficulties with images containing structures that are similar to pipelines and areas of low contrast to the pipeline course. This can be partly compensated with the collection of more training data. The model can only be applied on data that contains the pipeline during construction. This either means that the date of construction of the pipeline is known to the user or the model is only part of a larger process. One suggestion for such a process will be discussed in the following chapter.

Chapter 7

Proposed Process

In the previous chapter, it was discussed that the model can only be part of a larger process to create a meaningful way to detect pipelines on satellite images. Only using the trained model of the previous chapters, pipeline images from the construction phase are required to produce an output containing a reasonable part of the pipeline course. This is not practical since the date of construction is usually not known. This chapter proposes an automatic process using the model. For reasons already discussed in chapter 3 the process will be based on Landsat images.

To detect pipeline courses manually on satellite images, a region of interest needs to be defined. The region of interest denotes an area where a pipeline was likely installed. These areas can be found e.g. by gaps in available data sources like OSM. Using the region of interest, a list of images depicting this region needs to be accessed. The images can be used as input for a trained model. The model will always falsely detect pixels as pipeline. The number of pipeline pixels detected in an image before and during a pipeline construction should differ. Thus, the images need to be successively fed into the model, starting at the oldest image. A sudden change in the number of pipeline pixels indicates a pipeline construction. Hence, these images need to be flagged. Georeferenced data about the pipeline course needs to be extracted from the flagged images. For this, the generated pixels maps from the model created by the flagged images can be fed into a post-processing algorithm. An overview of the proposed process can be seen in Fig. 7.1. Each step will be discussed in detail in the following sections.

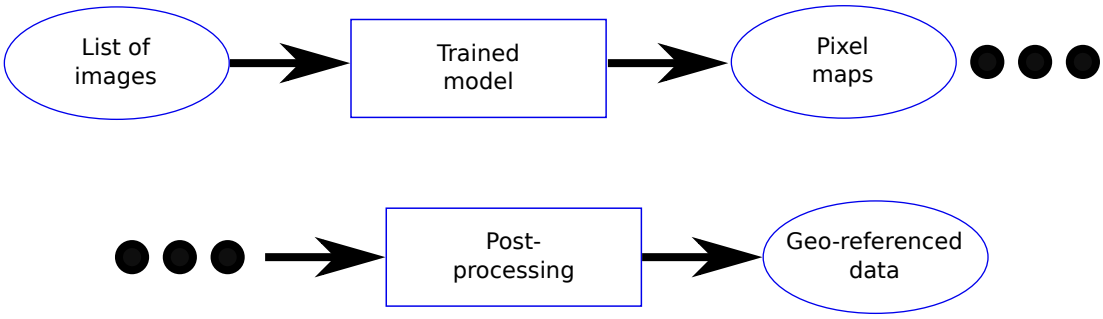


Figure 7.1: Overview of steps of proposed process for the automatic extraction of pipeline courses from satellite images. Ellipses represent inputs and outputs while rectangles represent process steps.

7.1 Definition of Region and Generation of Image List

A great number of satellite images are needed to detect pipelines in an area. Thus, an image source needs to be accessed to generate a list of images. This can be done through application programming interfaces (APIs). One example of an API accessing Landsat images through Python is *landsatexplore* [72]. It takes the coordinates of the region of interest, the desired minimal cloud cover of the images and a time frame to return a list of Landsat images. Clouds can block the view on pipeline construction sites. Thus, the cloud cover needs to be minimal. The time frame required as input for the API can either be kept as wide as possible or can be specified if further knowledge of the pipeline construction in the region was available. Different satellites have different spectral sensitivities. Hence, images from different satellites have other characteristics. The model the images are fed into was likely trained on images of one satellite to achieve higher performance. Therefore, the output list from the API has to be additionally filtered by satellite. The resulting list contains all images of a certain satellite of a region of interest over some time.

7.2 Pipeline Detection

Now that a filtered list of images of the region of interest is available, the images containing the pipeline construction need to be found. This is done by successively downloading the actual image using the API and feeding it into a model trained on images from the specific satellite. For this, the images likely need pre-processing like normalization and NDVI calculation (see chapter 5). Like mentioned before, the

numbers of detected pipeline pixels will peak between an image before and during the construction. Images of a high pipeline pixel number need to be flagged. The pixel maps generated by the model from the flagged images can be used for post-processing in the next step.

7.3 Post-Processing

Georeferenced data needs to be extracted from the images to make a dataset usable for gas networks. The pixel maps produced by the model using images of pipeline constructions normally contain small patches of false positives. Additionally, they contain larger patches of true positives interrupted by areas of false negatives (see Chapter 6, Figs. 6.5 and Figs. 6.5). The goal of the post-processing is to filter out the false positives and connect the patches of true positives. This creates a pixel map that can be used to extract georeferenced data about the pipeline course.

Small patches of false positives can be filtered out by factors like e.g. proximity to other patches. Additionally, other datasets like road datasets can be used to filter out roads falsely labeled as pipeline. Larger patches of true positives can be connected using e.g. snakes. Snakes or active contour models are for delineating objects. They use deformable lines that are adjusted to fit features of interests. These were already proven to work on similarly shaped structures like roads [73]. The true positives of the improved pixel map need to be exported to a georeferenced data format like GeoJSON or shapefile. Pixels in satellite images are georeferenced. As long as the structure of the pixel map does not differ from the original image, the true positives can be read out from it. The output data can then be used in a gas network.

Chapter 8

Conclusion and Outlook

In the course of this thesis the following goals were achieved:

- A training and test dataset for the training of machine learning models for the detection of pipelines on satellite images was created. The training dataset contains 375 images of the size 64×64 px of gas transport pipelines in Great Britain. The test dataset contains 37 images of the same dimensions of the NEL pipeline in Northern Germany.
- An U-net like deep learning model was successfully trained with the above-mentioned training dataset with initial learning rates between $1\mathrm{e}{-03}$ and $1\mathrm{e}-06$. Models trained with the learning rates of $1\mathrm{e}{-03}$ and $1\mathrm{e}{-04}$ were found to be converging at a mean max. val. IoU score of 0.66 ± 0.02 and 0.65 ± 0.03 respectively. Models trained with the initial learning rate of $1\mathrm{e}-05$ were found to be close to convergence at a mean max. val. IoU score of 0.62 ± 0.03 whereas for the initial learning rate of $1\mathrm{e}-06$ no convergence was determined.
- The ability of the model to generalize to a different region in Europe was tested. For this, the converging models or models close to convergence were evaluated on the above-mentioned dataset. The best performing models were found to be the ones trained with an initial learning rate of $1\mathrm{e}{-04}$ performing with a mean IoU score of 0.57 ± 0.01. This makes it perform 0.08 worse in terms of IoU score than the mean max. val. IoU score during the training process for this learning rate. A visual performance assessment confirms a reduction in performance. However, the trained models were still found to be able to detect pipelines on images.

Based on the findings of this thesis the successful detection of pipelines on satellite images with resolutions of $30\,\mathrm{m\,px^{-1}}$ using machine learning methods was proven.

With IoU scores greater than 0.5 this was also proven for different vegetation zones. However, the current process was found to be far from fully automatic. The images have to be selected and loaded manually into the model, and the output of the model needs further manual post-processing to make it usable for gas networks. Thus, a process for the automatic detection and extraction of pipeline courses from satellite images based on the model considered in this thesis was proposed in Chapter 7. Nonetheless, there are some more immediate steps which can be taken to improve the performance of the model:

- The collection of more training data throughout regions of Europe
- The correction of mislabeled data
- The collection of image data with higher resolution

Additionally, there are the following questions to be clarified:

- How does the trained model perform on data from vastly different vegetation zones like the Mediterranean scrubland in Spain?
- Is it even sensible to train the model not only for images of one satellite at a time but also for one vegetation zone?
- How does the trained model perform on thinner pipelines than used in this thesis?
- Can a model be trained with data with lower resolution, e.g. with data from Landsat 1-3?

Bibliography

[1] DLR Institute of Networked Energy Systems. SciGRID_gas - General information. Accessed: 10.10.2020. URL: https://www.gas.scigrid.de/.

[2] Meysam Qadrdan, Muditha Abeysekera, Jianzhong Wu, Nick Jenkins, and Bethan Winter. Fundamentals of natural gas networks. In *The Future of Gas Networks*, pages 5–22. Springer, 2020.

[3] H. Müller. Fuels | gaseous. In Paul Worsfold, Alan Townshend, and Colin Poole, editors, *Encyclopedia of Analytical Science (Second Edition)*, pages 505 – 511. Elsevier, Oxford, second edition edition, 2005. URL: http://www.sciencedirect.com/science/article/pii/B0123693977002119, doi:https://doi.org/10.1016/B0-12-369397-7/00211-9.

[4] Stefan Heyne. *Bio-SNG from Thermal Gasification - Process Synthesis, Integration and Performance*. PhD thesis, 05 2013.

[5] Bob Dudley et al. Bp statistical review of world energy. *BP Statistical Review, London, UK, accessed Aug*, 6:2018, 2018.

[6] Saeid Mokhatab and William A Poe. *Handbook of natural gas transmission and processing*. Gulf professional publishing, 2012.

[7] Saeid Mokhatab and William A. Poe. Chapter 1 - natural gas fundamentals. In Saeid Mokhatab and William A. Poe, editors, *Handbook of Natural Gas Transmission and Processing (Second Edition)*, pages 1 – 42. Gulf Professional Publishing, Boston, second edition edition, 2012. URL: http://www.sciencedirect.com/science/article/pii/B9780123869142000017, doi:https://doi.org/10.1016/B978-0-12-386914-2.00001-7.

[8] Sergey Paltsev. Economics and geopolitics of natural gas: Pipelines versus lng. In *2015 12th International Conference on the European Energy Market (EEM)*, pages 1–5. IEEE, 2015.

[9] Leslie Tomory. The environmental history of the early british gas industry, 1812–1830. *Environmental History*, 17(1):29–54, 2012.

[10] FW Goodenough. Coal gas as a fuel for domestic purposes. lecture ii. *Journal of the Royal Society of Arts*, 61(3170):898–906, 1913.

[11] Jens Bjørnmose, Ferran Roca, Tatsiana Turgot, and D Smederup Hansen. An assessment of the gas and oil pipelines in europe. *European Parliament, Brussels*, 2009.

[12] Maciej Gucma and Chł. The impact of a liquefied natural gas terminal on the gas distribution and bunkering network in poland. pages 154–160, 01 2018. doi:10.17402/278.

[13] John L. Kennedy. *Oil and gas pipeline fundamentals*. Pennwell books, 1993.

[14] ENTSOG. Mission. Accessed: 02.10.2020. URL: https://www.entsog.eu/mission.

[15] EU Regulation. Regulation (eu) 2018/1999 of the european parliament and of the council of 11 december 2018 on the governance of the energy union and climate action, amending regulations (ec) no 663/2009 and (ec) no 715/2009 of the european parliament and of the council. Technical report, Directives 94/22/EC, 98/70/EC, 2009/31/EC, 2009/73/EC, 2010/31/EU, 2012/27 ..., 2018.

[16] ENTSOG. Members. Accessed: 02.10.2020. URL: https://entsog.eu/sites/default/files/2020-06/entsog_members_map_Apr%202020.jpg.

[17] Rui Carvalho, Lubos Buzna, Flavio Bono, Eugenio Gutiérrez, Wolfram Just, and David Arrowsmith. Robustness of trans-european gas networks. *Physical review E*, 80(1):016106, 2009.

[18] Central Intelligence Agency. The World Factbook - Pipeline, Feb 2018. Accessed: 10.09.2020. URL: https://www.cia.gov/library/publications/the-world-factbook/fields/383.html.

[19] Eurostat. Where does our energy come from? Accessed: 04.10.2020. URL: https://ec.europa.eu/eurostat/cache/infographs/energy/bloc-2a.html.

[20] European Commission. Liquefied Natural Gas. Accessed: 04.10.2020. URL: https://ec.europa.eu/energy/topics/oil-gas-and-coal/liquefied-natural-gas-lng_en.

[21] Edouard Jérôme Robert LOTZ. *Could US LNG challenge Russian Gas within the European Union in the long term?* PhD thesis, ESCP Europe, 2020.

[22] Niteshift. Hasenhäge NEL-Erdgas-Pipeline Bau 2011-05-15 009.JPG, 2011. Accessed: 31.07.20. URL: https://commons.wikimedia.org/wiki/File:Hasenh%C3%A4ge_NEL-Erdgas-Pipeline_Bau_2011-05-15_009.JPG.

[23] Boyun Guo, Shanhong Song, Ali Ghalambor, and Tian Ran Lin. Chapter 11 - pipeline installation methods. In Boyun Guo, Shanhong Song, Ali Ghalambor, and Tian Ran Lin, editors, *Offshore Pipelines (Second Edition)*, pages 135 – 146. Gulf Professional Publishing, Boston, second edition edition, 2014. URL: http://www.sciencedirect.com/science/article/pii/B978012397949000011X, doi:https://doi.org/10.1016/B978-0-12-397949-0.00011-X.

[24] Vuo. Nord Stream pipe in Kotka.jpg, 2017. Accessed: 31.07.20. URL: https://commons.wikimedia.org/wiki/File:Nord_Stream_pipe_in_Kotka.jpg.

[25] *Remote sensing : models and methods for image processing*. ScienceDirect. 3. ed. edition, 2007.

[26] European Space Agency. Sentinel 2. Accessed: 03.09.2020. URL: https://sentinel.esa.int/web/sentinel/missions/sentinel-2.

[27] National Aeronautics and Space Administration. Landsat Science. Accessed: 03.09.2020. URL: https://landsat.gsfc.nasa.gov/.

[28] United States Geological Survey. Landsat Missions Timeline. Accessed: 03.09.2020. URL: https://www.usgs.gov/media/images/landsat-missions-timeline.

[29] Darryl Keith. *Coastal and Estuarine Waters: Optical Sensors and Remote Sensing for Management*. 08 2014. doi:10.1201/9780429441004-5.

[30] Robert A Schowengerdt. *Remote sensing: models and methods for image processing*. Elsevier, 2006.

[31] Matthias Drusch, Umberto Del Bello, Sébastien Carlier, Olivier Colin, Veronica Fernandez, Ferran Gascon, Bianca Hoersch, Claudia Isola, Paolo Laberinti, Philippe Martimort, et al. Sentinel-2: Esa's optical high-resolution mission for gmes operational services. *Remote sensing of Environment*, 120:25–36, 2012.

[32] United States Geological Survey. Landsat Missions - Landsat 5. Accessed: 16.09.2020. URL: https://www.usgs.gov/core-science-systems/nli/landsat/landsat-5.

[33] NASA Landsat Science. Visual comparison of Landsat spectral bands . Accessed: 06.09.2020. URL: https://landsat.gsfc.nasa.gov/wp-content/uploads/2012/12/all_Landsat_bands.png.

[34] FU Berlin Remote Sensing and Geoinformatics staff. Remote Sensing Data Analysis - Sentinel 2 bands. Accessed: 04.09.2020. URL: https://blogs.fu-berlin.de/reseda/files/2018/05/sentinel_2_channels.png.

[35] Markus Aschwanden. *Physics of the solar corona: an introduction with problems and solutions*. Springer Science & Business Media, 2006.

[36] HW Wu, A Emadi, G De Graaf, J Leijtens, and RF Wolffenbuttel. Design and fabrication of an albedo insensitive analog sun sensor. *Procedia Engineering*, 25:527–530, 2011.

[37] John Weier and David Herring. Measuring vegetation (ndvi & evi). *NASA Earth Observatory*, 20, 2000.

[38] United States Geological Survey. Landsat Normalized Difference Vegetation Index. Accessed: 04.09.2020. URL: https://www.usgs.gov/land-resources/nli/landsat/landsat-normalized-difference-vegetation-index?qt-science_support_page_related_con=0#qt-science_support_page_related_con.

[39] Gopinath Rebala, Ajay Ravi, and Sanjay Churiwala. *An introduction to machine learning*. Springer, 2019.

[40] Witold Pedrycz and Shyi-Ming Chen. *Deep Learning: Concepts and Architectures*. Springer, 2020.

[41] baeldung. Genetic Algorithms vs Neural Networks. Accessed: 03.10.2020. URL: https://www.baeldung.com/cs/genetic-algorithms-vs-neural-networks.

[42] Yung-Yao Chen, Yu-Hsiu Lin, Chia-Ching Kung, Ming-Han Chung, I Yen, et al. Design and implementation of cloud analytics-assisted smart power meters considering advanced artificial intelligence as edge analytics in demand-side management for smart homes. *Sensors*, 19(9):2047, 2019.

[43] Christopher M Bishop. *Pattern recognition and machine learning*. springer, 2006.

[44] Anurag Bhardwaj, Wei Di, and Jianing Wei. *Deep Learning Essentials: Your hands-on guide to the fundamentals of deep learning and neural network modeling*. Packt Publishing Ltd, 2018.

[45] Shruti Jadon. A survey of loss functions for semantic segmentation. *arXiv preprint arXiv:2006.14822*, 2020.

[46] Richard Elderman. *Exploring Improvements for Gradient Descent Optimization Algorithms in Deep Learning*. PhD thesis, 2019.

[47] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[48] Yann LeCun, Corinna Cortes, and Christopher JC Burges. The mnist database. *URL http://yann. lecun. com/exdb/mnist*, 1998.

[49] Aurélien Géron. *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow: Concepts, tools, and techniques to build intelligent systems*. O'Reilly Media, 2019.

[50] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[51] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708, 2017.

[52] Linda Shapiro. *Computer vision and image processing*. Academic Press, 1992.

[53] Irem Ulku and Erdem Akagunduz. A survey on deep learning-based architectures for semantic segmentation on 2d images. *arXiv preprint arXiv:1912.10230*, 2019.

[54] University of Freiburg LMB. Our U-net wins two Challenges at ISBI 2015. Accessed: 12.10.2020. URL: `https://lmb.informatik.uni-freiburg.de/people/ronneber/isbi2015/`.

[55] E. Goceri. Challenges and recent solutions for image segmentation in the era of deep learning. In *2019 Ninth International Conference on Image Processing Theory, Tools and Applications (IPTA)*, pages 1–6, 2019.

[56] Nathalie Japkowicz and Mohak Shah. Performance evaluation in machine learning. In *Machine Learning in Radiation Oncology*, pages 41–56. Springer, 2015.

[57] National Grid UK. Network route maps. Accessed: 10.09.2020. URL: https://www.nationalgrid.com/uk/gas-transmission/land-and-assets/network-route-maps.

[58] Global Energy Monitor. National Transmission System. Accessed: 16.09.2020. URL: https://www.gem.wiki/National_Transmission_System.

[59] OpenStreetMap contributors. Planet dump. Accessed: 12.10.2020. URL: https://planet.osm.org.

[60] Adam Pluta and Ontje Lünsdorf. esy-osmfilter–a python library to efficiently extract openstreetmap data. *Journal of Open Research Software*, 8(1), 2020.

[61] NEL Gastransport GmbH. NEL. Die Nordeuropäische Erdgasleitung. Accessed: 10.09.2020. URL: https://www.nel-gastransport.de/netzinformationen/die-nordeuropaeische-erdgasleitung/.

[62] NEL Gastransport GmbH. The North European Natural Gas Pipeline. Accessed: 04.09.2020. URL: https://www.nel-gastransport.de/en/our-network/the-north-european-natural-gas-pipeline.

[63] NEL Gastransport GmbH. NEL goes on stream on time, Aug 2012. Accessed: 12.09.2020. URL: https://www.nel-gastransport.de/en/press/press-releases/press-releases/nel-goes-on-stream-on-time.

[64] NEL Gastransport GmbH. The NEL Network. Accessed: 04.09.2020. URL: https://www.nel-gastransport.de/fileadmin/bilder/nel/NEL_Infrastructure.png.

[65] National Grid UK. National Transmission System. Accessed: 16.09.2020. URL: https://www.nationalgrid.com/uk/gas-transmission/land-and-assets/network-route-maps.

[66] Noel Gorelick, Matt Hancher, Mike Dixon, Simon Ilyushchenko, David Thau, and Rebecca Moore. Google earth engine: Planetary-scale geospatial analysis for everyone. *Remote Sensing of Environment*, 2017. URL: https://doi.org/10.1016/j.rse.2017.06.031, doi:10.1016/j.rse.2017.06.031.

[67] Kristi Sayler and Karen Zanter. Landsat 4-7 collection 1 (c1) surface reflectance (ledaps) product guide. 2020.

[68] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on*

Medical image computing and computer-assisted intervention, pages 234–241. Springer, 2015.

[69] Zhengxin Zhang, Qingjie Liu, and Yunhong Wang. Road extraction by deep residual u-net. *IEEE Geoscience and Remote Sensing Letters*, 15(5):749–753, 2018.

[70] Yanbing Bai, Erick Mas, and Shunichi Koshimura. Towards operational satellite-based damage-mapping using u-net convolutional network: A case study of 2011 tohoku earthquake-tsunami. *Remote Sensing*, 10(10):1626, 2018.

[71] Volodymyr Mnih. *Machine Learning for Aerial Image Labeling*. PhD thesis, University of Toronto, 2013.

[72] Yann Forget. landsatxplore. URL: `https://pypi.org/project/landsatxplore/`.

[73] Rickard Sirefelt. Road extraction from aerial images. Master's thesis, 2015.

Acknowledgments

I would like to thank my supervisor Adam Pluta for guiding me through the thesis processes and the weekly meetings where we could explore and discuss ideas. I would also like to thank my other colleagues for their support and help.

Erklärung

Hiermit versichere ich an Eides statt, dass ich diese Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe. Außerdem versichere ich, dass ich die allgemeinen Prinzipien wissenschaftlicher Arbeit und Veröffentlichung, wie sie in den Leitlinien guter wissenschaftlicher Praxis der Carl von Ossietzky Universität festgelegt sind, befolgt habe.

Oldenburg, 04.11.2020 J. Dasenbrock

Ort, Datum Jan Dasenbrock