



**Institut für Chemie und Biologie der Meeres (ICBM) & Institut für Vernetzte
Energiesysteme am Deutschen Zentrum für Luft- und Raumfahrt (DLR)**
Studiengang Umweltmodellierung M. Sc.

Masterthesis

**PIPELINE DETECTION USING
UNCERTAINTY-DRIVEN MACHINE
LEARNING**

Hendrik-Pieter Tetens
Matrikelnr.: 5906917

10. Dezember 2021

1. Gutachter: Prof. Dr. Carsten Agert
2. Gutachter: Dr. Adam Pluta
1. Betreuer: Dr. Adam Pluta
2. Betreuer: Matthias Zech

Contents

List of Figures	I
List of Tables	III
1 Introduction	1
1.1 The Process of Building Pipelines	4
1.2 Satellite Imagery	4
2 Semantic Image Segmentation	9
3 Deep Learning and Convolutional Neural Networks	12
3.1 The Perceptron	12
3.2 Multi-Layer Perceptron	15
3.2.1 Activation Functions	17
3.2.2 Training of Artificial Neural Networks	20
3.3 Convolutional Layers	22
3.4 Max-Pooling	26
3.5 Batch Normalization	28
3.6 Transposed Convolutions	28
4 Uncertainty in Deep Learning	30
4.1 Bayesian Deep Learning	31
4.2 Deep Ensembles	32
4.3 Active Learning	33
5 Model and data prerequisites	35
5.1 Choosing a Model Architecture and Encoder-Backbone	36
5.1.1 U-Net	37
5.1.2 LinkNet	37
5.1.3 Backbone Architectures	40
5.1.4 Comparisons	43
5.2 Revision: The Great Britain and Northern Germany Dataset	47

5.3	Dataset extension: The Spanish Dataset	50
6	Hypotheses, Evaluation and Results	54
6.1	Hypothesis 1: Model Generalizability	54
6.1.1	Model Generalizability: Training of a GBNEL Deep Ensemble	55
6.1.2	Model Generalizability: Results	57
6.2	Hypothesis 2: Efficiently Estimating Model Uncertainties	63
6.2.1	Efficiently Estimating Model Uncertainties: The Wasserstein Distance and the KS-Test	63
6.2.2	Efficiently Estimating Model Uncertainties: Results	65
6.3	Hypothesis 3: Active Learning	69
6.3.1	Active Learning: Procedure	69
6.3.2	Active Learning: Results	71
7	Discussion	78
8	Conclusion & Outlook	83
References		

List of Figures

1.1	Exemplary output of the model trained by Dasenbrock et al.	2
1.2	Right-of-way during and after the construction of a pipeline	5
1.3	high-resolution Right-of-ways OPAL and NEL near Lubmin	6
1.4	low-resolution right-of-ways of OPAL and NEL pipelines near Lubmin	7
2.1	Image segmentation in 1979	9
2.2	Subcategories in Image Segmentation	10
3.1	The perceptron	13
3.2	Multi-layer perceptron with two layers	15
3.3	Multi-layer perceptron modeling logical XOR	16
3.4	Sigmoid activation function	17
3.5	Tanh activation function	18
3.6	ReLU activation function	19
3.7	Weight update	21
3.8	Convolution operation at first filter position	23
3.9	Convolution operation at second filter position	24
3.10	Comparison stride=1 and stride=2	25
3.11	Zero padding	26
3.12	Downsampling with max-pooling	27
3.13	Max-pooling operation	27
3.14	Transposed convolution	29
3.15	Stride and padding in transposed convolutions	29
4.1	Model uncertainties with softmax activation	30
4.2	Active learning: sample efficiency	34
5.1	U-Net architecture	38
5.2	LinkNet architecture	39
5.3	Google Trends comparison for search entrys <i>tensorflow</i> (red) and <i>py-torch</i> (blue)	40
5.4	ResNet building block	41

5.5	Intersection over union score	44
5.6	Creation of image-mask pairs	48
5.7	Automatically created image-mask pair	49
5.8	Overview map of Europe	50
5.9	Available OSM pipeline data in Spain	51
5.10	Landsat 5 imagery download and preparation	52
5.11	Pipeline segments in final dataset	53
6.1	Exemplary comparison of different regions	55
6.2	Right-of-way visibility in different regions	56
6.3	Validation IoU for GBNEL and Spanish test datasets	58
6.4	Boxplots of GBNEL and ES sample variance	59
6.5	Selected model outputs for GBNEL pipeline pathways	60
6.6	Selected model outputs for Spanish pipeline pathways	62
6.7	Cityscape street scenes	63
6.8	Wasserstein distance in 2-D	65
6.9	Wasserstein distances of sample uncertainty estimates	66
6.10	Comparison of individual sample uncertainty estimates	67
6.11	Comparison of ensemble validation IoUs	72
6.12	Comparison of sample uncertainty estimates	73
6.13	Comparisons of IoU GBNEL and ES	74
6.14	Low pipeline-background contrasts in GBNEL and ES data	76
6.15	Two uncertainty estimate series for typical GBNEL and problematic ES data	77

List of Tables

1.1	Landsat 5 bands	8
3.1	Outputs of perceptron model for logical AND	14
3.2	Outputs of perceptron model for logical OR	14
3.3	Output table of logical XOR	14
5.1	Architecture-backbone comparisons	45
6.1	GBNEL ensemble member performances	57

1 Introduction

Natural gas plays a vital role in the European energy mix (European Parliament, 2017). It is not only the most important energy carrier on the heat market but also extensively used in other branches like the chemical industry. As a flexibility option that allows for electricity generation and energy storage, it is regarded as a possible compensation storage for renewable energies in the future. Compared to other fossil fuels, it has a lower CO₂ footprint making it a sensible energy carrier during the transition into a carbon-free free economy (Federal Ministry for Economic Affairs and Energy, 2019). An extensive pipeline-based gas transmission grid serves as the backbone of the European gas economy (Carvalho et al., 2009). It is used to import high gas volumes from, e.g., Russia, Norway, and Algeria (European Parliament, 2017) but also to enable fast transmission of imported and self-produced natural gas throughout the countries.

Another development emphasizing the importance of the gas transmission grid stems out of the current unprecedented political and business momentum of hydrogen-based energy solutions. Molecular hydrogen (H₂) promises, inter alia, to be a versatile energy carrier, like natural gas, can be produced by means of renewable energies and is suitable to be transported in pipelines (IEA, 2019). Currently, a lot of effort is put into the investigation on how to transport H₂ in the existing gas infrastructure due to the different properties of H₂ in comparison to natural gas. Ideas range from blending hydrogen with natural gas like in (Isaac, 2019) to a complete reassignment of pipelines (Cerniauskas, Jose Chavez Junco, Grube, Robinius, & Stolten, 2020). The importance of the European gas transmission grid and the future challenges make its modeling to assist profound decisions regarding its development crucial. Therefore, the three year project SciGRID_{gas} has been initiated in the beginning of 2018 and was funded by the *Federal Ministry for Economic Affairs and Energy* (Medjroubi, Diettrich, Pluta, & Dasenbrock, 2021). The goal of the project was to create the first open-source gas transport data model for the European gas infrastructure. It provides future users – e.g., researchers in the field of energy system modeling – with tools and data to generate data models, which can then be used for solving a wide variety of modeling-, simulation-, and optimization tasks. This

makes SciGRID_{gas} an important addition to the zoo of models and tools aimed at the German and European energy transition towards a renewable energy-based system.

Within the project novel data collection techniques were explored. One of these techniques was the use of modern machine learning algorithms on historical satellite imagery to detect pipeline pathways. The results were published in (Dasenbrock, Pluta, Zech, & Medjroubi, 2021). The authors showed that the method detects pipelines in an adequate manner in order to eventually extract the precise location of individual pipeline segments, which is important to the modeling of the European gas transmission infrastructure. Because this was a proof-of-concept study, the trained machine learning algorithm was only trained on and applied to selected pipeline pathways in Great Britain and Northern Germany (Dasenbrock et al., 2021). Figure 1.1 shows an example output of the trained model.

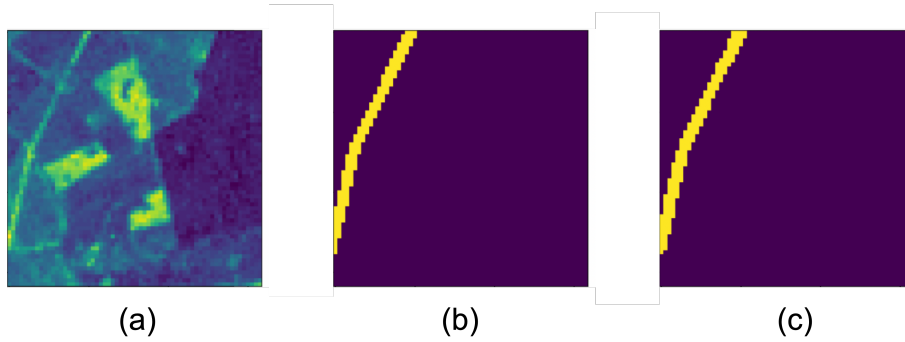


Figure 1.1 **Exemplary output of the model trained by Dasenbrock et al.**

(a) shows the $64 \text{ px} \times 64 \text{ px}$ satellite image provided to the model. (b) depicts the ground truth where the actual pipeline path is marked in yellow. (c) is the model output segmenting the input into pipeline (yellow) and background (dark purple). Pictures from (Dasenbrock et al., 2021).

The aim of this master thesis is to further develop the approach in order to make it potentially applicable in an efficient and reliable manner to the entirety of Europe. Therefore, the following research questions have been prepared:

- Is a model, respectively, deep learning algorithm trained on satellite imagery from Great Britain and Northern Germany able to sufficiently detect pipeline pathways in other regions that feature different vegetational and geographic properties?
- Can an *active learning* approach that uses estimated sample uncertainties to actively choose training samples of high curiosity to the model benefit the adaption of the model to other regions? The benefits include a more sample-efficient training of the model and a better pipeline detection accuracy overall.

- Because an active learning approach will make use of so-called *deep ensembles* that combine the output of multiple member *neural networks*, it is essential to find an efficient model architecture and a sufficient number of model members. It is asked if there is a more suited model architecture than used in (Dasenbrock et al., 2021) and how many ensemble members are needed to provide high-quality uncertainty estimates.

Along with the research questions, additional insights, for example, sources of error related to particular satellite image features, are analyzed. Answering these questions will likely provide a profound basis for a subsequent deployment of the model to progressively detect pipeline pathways in Europe.

The thesis is structured as follows: First, in order to understand the general approach of detecting pipeline pathways on satellite imagery, the process of building pipelines and satellite imagery as a data source is explained. Next, the task at hand is located within the field of *computer vision*. As we will see, modern deep learning algorithms and especially *convolutional neural networks* are the state-of-the-art solution to this kind of problem. Therefore, the theory behind these models is presented: From a simple one unit *perceptron* toward a *multi-layer perceptron* – a type of neural network – and its training, convolutional neural networks and different techniques within these networks are introduced. Prepared with the theoretical foundations, the topic of *model uncertainties* and their estimation is derived, providing the theoretical groundwork for the *active learning* approach, which is described afterward. The selection of an appropriate model architecture and the process of generating training and test data for Great Britain, Northern Germany, and Spain are presented in the next chapter. This covers all prerequisites to be able to answer the research questions of this thesis. Each question is represented by a derived hypothesis, the specific approach on how to evaluate each hypothesis is described, and the consequent results are presented. Next, the obtained results are discussed. The work ends with a conclusion and an outlook illustrating possible working packages in order to deploy the developed model in future projects.

1.1 The Process of Building Pipelines

A key challenge of detecting European onshore transmission pipelines originates out of the fact that their vast majority runs underground (Diettrich, Pluta, & Medjroubi, 2021). The burial of pipelines has several advantages: The pipeline is better protected against damage – be it through single incidents or long term factors like tough weather conditions –, the temperature in the ground is more stable and the landscape can in areas with low tree cover mostly recover to its original state after the pipeline has been constructed (Alkazraji, 2008). In order to overcome the challenge of detecting underground pipelines and to introduce the approach followed in this work, the pipeline construction process is briefly outlined.

Given that the pipes themselves are already produced and available for installation, the soil where the final pipeline will be buried has to be prepared. Therefore, a working corridor – the so-called *right-of-way* (*ROW*) – is established. The right-of-way needs to be of sufficient width – usually 9 to 46 meters (Johnson, Gagnolet, Ralls, & Stevens, 2011) – to allow for an easy access of construction machinery. Additionally, it has to leave enough space for the *stringing* of the single pipes along the ROW (Menon, 1978). It is cleared from all obstacles like trees, fences, and other entities and the topsoil is stripped. The trench that will contain the final pipeline is excavated. The individual pipeline segments are then welded together and transferred into the trench. Afterward the removed soil is backfilled and the topsoil replaced (Alkazraji, 2008). The right-of-way is maintained even after the construction of the pipeline. This is done to enable prolonged access to the pathway to provide pipeline maintenance and to protect it from damages which can, for example, be caused by tree roots (Menon, 1978). Figure 1.2 shows a right-of-way during and after the construction of a pipeline in a densely forested area.

1.2 Satellite Imagery

The previously described process of constructing a pipeline creates several options to detect pipeline pathways on satellite imagery. One option is to use high-resolution satellite images where the right-of-way can, in many cases, easily be identified even years after the pipeline construction itself. Figure 1.3 provides the current *Google earth* high-resolution image consisting of satellite imagery from 08.06.2018 and newer (Gorelick et al., 2017). The image is located near Lubmin – the location where the Nordstream pipelines are entering Germany – and depicts the point where the OPAL pipeline, which is in operation since 2011 (EUGAL, 2021), and the NEL pipeline, which is in operation since 2012 (NEL, 2013), split. Both right-of-ways are easily



Figure 1.2 **Right-of-way during and after the construction of a pipeline**

The left image shows the ROW during the construction of the pipeline. The topsoil has been stripped, and individual pipe segments lie along the ROW to be welded together. In the right picture, the environment starts to recover after the pipeline has been installed. Reforestation is not possible in order to maintain easy access to the pipeline and to protect it from growing tree roots. Picture from (Orhan Degermenci, 2019).

identifiable even years after their establishment. In forested areas, the ROWs are even more exposed due to the necessary deforestation along their path.

Even though the use of high-resolution imagery possibly allows for a reliable detection of pipeline pathways, it has the disadvantage of not providing additional meta information like the year of construction for pipelines that have been built before the deployment of the respective satellite. However, this information can be of high interest for future projects where further pipeline specifics could be derived from their year of construction. Therefore the use of historical low-resolution satellite imagery has been evaluated in (Dasenbrock et al., 2021). Figure 1.4a shows a low-resolution image also from 2018 depicting approximately the same area as Figure 1.3. Neither the ROW of the OPAL nor the NEL pipeline is identifiable. Figure 1.4b shows another low-resolution image of the same location but shot closely after the construction of the NEL pipeline in 2011. The right-of-way is clearly visible and most likely to be detected by sophisticated machine learning algorithms.

(Dasenbrock et al., 2021) proved that low-resolution imagery is sufficient to detect pipeline pathways if it shows the right-of-way during or shortly after the construc-



Figure 1.3 **high-resolution Right-of-ways OPAL and NEL near Lubmin**

Current Google earth satellite image at location $54^{\circ}06'08''\text{N}$ $13^{\circ}36'34''\text{E}$ (WGS 84). The satellite imagery is from 08.06.018 or newer and provided by GeoBasis-DE/BKG (Gorelick et al., 2017). The Right-of-ways are framed by red lines. The ROW running from north to south-west is the NEL pipeline. The pipeline running from north to south is the OPAL pipeline.

tion of a pipeline. For the evaluation, they used imagery taken by the Landsat 5 satellite, whose mission lasted from 1984 to 2013 (USGS, 2021e). Thus, it potentially captured the construction of many European pipelines over a span of 29 years. Additionally, the Landsat 5 data is easily accessible and publicly available for example through EARTHExplorer (USGS, 2021a) or GOOGLE EARTH ENGINE (Gorelick et al., 2017). Another advantage of using this data is that its low resolution decreases the computational demand compared to the use of high-resolution data. This is because fewer image pixels have to be analyzed by a machine learning algorithm per unit area. For these reasons Landsat 5 satellite imagery is also used in this work.

To better understand the Landsat 5 data, it is described in more detail: With

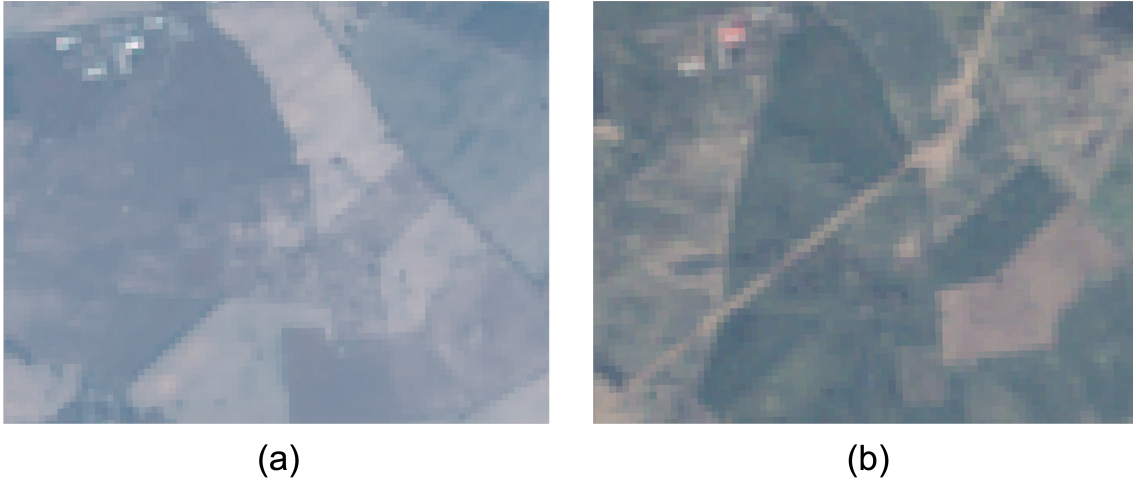


Figure 1.4 **low-resolution right-of-ways of OPAL and NEL pipelines near Lubmin**
(a) shows an image taken on 11.10.2018 by the Landsat 8 satellite with a resolution of 30 m px^{-1} (USGS, 2021c). Compared to the high-resolution image Figure 1.3 depicting roughly the same area, no ROWs can be identified. (b) shows a satellite image from the Landsat 5 mission with a resolution of 30 m px^{-1} (USGS, 2021e) shot closely after the construction of the NEL pipeline (22.04.2011). The ROW is clearly visible.

29 years of service, Landsat 5 set the Guinness World record for *Longest Operating Earth Observation Satellite*. It travels at an altitude of 705 km and circles the earth every 99 minutes resulting in 150,000 earth revolutions during its lifetime. The repeat cycle is 16 days (USGS, 2021e), meaning that the satellite is always traveling along the same path and will be at the initial position again after 16 days (Liew, 2001). Overall over 2.9 million satellite images were captured using the Worldwide Reference System-2 path/row system.

The Landsat 5 satellite was equipped with two sensors: The Multispectral Scanner (MSS) and the Thematic Mapper (TM). In this work, only the TM-generated images are considered because of the drastically lower resolution of the MSS sensor of $57 \times 79 \text{ m px}^{-1}$ and its only four bands (green, red, and two near-infrared bands). The Thematic Mapper failed in 2011 so there is no data available past 2011. The imagery captured by the TM sensor provides the bands and resolutions specified in Table 1.1 (USGS, 2021e):

In order to keep the pixel sizes consistent, only the bands 1–5 and 7 are considered for the rest of this work. It is estimated that thermal images with a resolution of 120^2 m px^{-1} do not benefit a model used to detect pipeline pathways. Looking back on Figure 1.4b, where only the bands 3 (red), 2 (green), and 1 (blue) were used to generate an image that follows the human intuition of color, the adopted bands 1–5 and 7 should be sufficient to detect pipeline pathways reliably. This was also shown

Table 1.1 **Landsat 5 bands**

Band	Wavelength [μm]	Spectrum	Resolution [m px^{-1}]
1	0.45 - 0.52	blue	30^2
2	0.52 - 0.60	green	30^2
3	0.63 - 0.69	red	30^2
4	0.76 - 0.90	near-infrared	30^2
5	1.55 - 1.75	near-infrared	30^2
6	10.40 - 12.50	thermal	120^2
7	2.08 - 2.35	mid-infrared	30^2

in (Dasenbrock et al., 2021). Of particular importance could be the interaction between bands 3 and 4 that are also used to calculate the so-called *normalized difference vegetation index (NDVI)*. The NDVI is used to quantify the density of living greens in a certain area such as a pixel of an image (Weier & Herring, 2000). Therefore bands 3 and 4 could be of particular interest when discriminating between a newly established right-of-way and its surroundings on a satellite image. The overall impact of each band on the detection of pipeline pathways is not further elaborated in this work but left implicit in the trained machine learning models themselves.

The available Landsat 5 data is deviated into different datasets depending on the data quality and applied preprocessing steps. In this work, only the *Landsat 5 TM Collection 1 Tier 1 calibrated top-of-atmosphere (TOA) reflectance* dataset is used. Tier 1 represents the highest quality of imagery that can also be used for time-series analysis because of its high image-to-image consistency regarding the pixel locations (USGS, 2021d).

This concludes the brief chapter on the process of building pipelines and satellite imagery. The following chapters will introduce the theory associated with neural networks that will guide the methods used to pursue the goals of this work.

2 Semantic Image Segmentation

Inside of machine learning, the task at hand – detecting pixels classified as *pipeline* in an input image – belongs to the category of *image segmentation*. In comparison to *image classification*, image segmentation does not provide a single class label for the whole image – e.g., *cat* or *dog* but pixel-wise class labels. Therefore, image segmentation is capable of detecting objects on an image and locates them within the provided image. Today the field of image segmentation is looking back on a rather long history, starting with the first algorithms developed in 1979 (Minaee et al., 2020). The algorithm segments simple grayscale images by finding optimal thresholds for gray levels, maximizing the separability between the different levels (Otsu, 1979). Figure 2.1 shows a result from 1979, segmenting an input image into two classes.

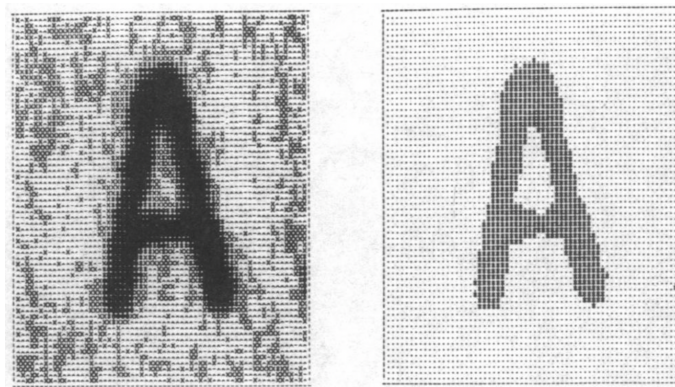


Figure 2.1 **Image segmentation in 1979**

An input image (left) is segmented into two classes (right) by finding an optimally discriminating threshold for grayscale values (Otsu, 1979).

Until today plenty of other approaches – many related to the field of classic machine learning – have been proposed and deployed, like *k-means clustering*, *active contours*, and *sparsity-based methods* (Minaee et al., 2020).

On a more granular level, the topic of image segmentation can be further divided into the following three subcategories (Kirillov, He, Girshick, Rother, & Dollár, 2019):

- **Semantic Image Segmentation**

The task in *semantic image segmentation* is to map each pixel in an input

image towards distinct object classes (Liu, Deng, & Yang, 2018) like, for example, *car*, *tree*, or *pedestrian* (see Figure 2.2b). There is no differentiation between single entities of an object class.

- **Instance-level Segmentation**

Instance-level segmentation can be viewed as an extension of semantic image segmentation and discriminates between single instances of an object class, making them countable (Z. Zhang, Fidler, & Urtasun, 2016). See Figure 2.2c for an instance segmentation of the cars and pedestrians seen in Figure 2.2a.

- **Panoptic Segmentation**

The last subcategory is *panoptic segmentation*, a combined approach of both the beforehand discussed subcategories. Instead of marking each pixel with either an object class or object instance, it provides each pixel with a *class label* and an *instance id* (Kirillov et al., 2019).

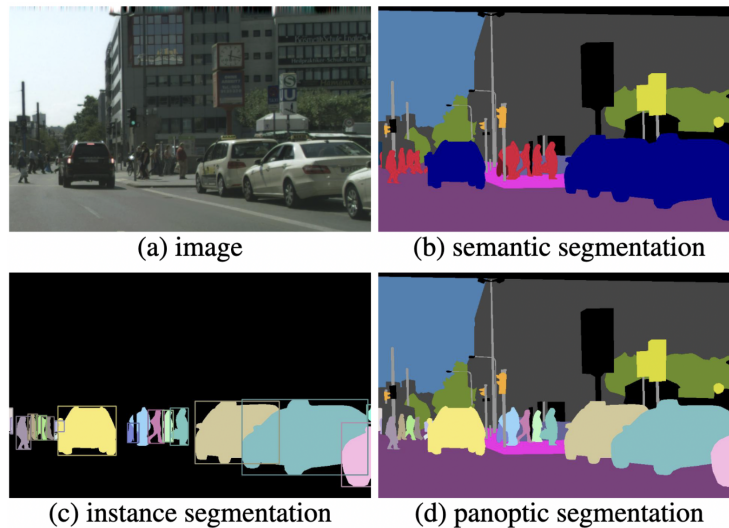


Figure 2.2 **Subcategories in Image Segmentation**

The original street scenery (a) as model input which is either segmented into the different object classes (b), segmented into different instances of an object class (c), or provided with both – the object class labels and instance-level ids (d) (Kirillov et al., 2019).

In the last three decades, the development and advancements of *Convolutional Neural Networks* (CNN) have been of special importance. This led to a substantial performance increase in the field of computer vision regarding classification and segmentation tasks (Sultana, Sufian, & Dutta, 2020). Nowadays, the application of these modern neural network architectures ranges over a wide variety of use cases like medical image analysis, self-driving cars, and video surveillance (Minaee et al.,

2020).

The detection of pipelines as approached in this master thesis belongs into the category of semantic image segmentation. Because of the great success of deep learning models, especially CNNs, in this category, the model used for the later detection of pipelines will also ground on a CNN. In order to provide a deeper understanding of the later proposed and compared deep learning architectures for the task at hand, the following chapter 3 will address the most important concepts that are extensively used in those architectures.

3 Deep Learning and Convolutional Neural Networks

This chapter will cover the most relevant concepts and ideas in deep learning that are required to understand the later discussed implementations of deep neural networks used in semantic image segmentation.

Therefore, an elementary neural network unit will be elaborated: The *perceptron*. Next, perceptrons will be accumulated into a stack of multiple layers called *multi-layer perceptron*, and the importance of non-linear *activation functions* will be discussed. Later the training process is described in detail. This serves as a starting point for the subsequent explanation of *convolutional layers*, which led to a breakthrough in computer vision in 2012 (Yamashita, Nishio, Do, & Togashi, 2018). Thereafter, the concepts *max-pooling*, *batch normalization*, and *transposed convolutions*, which have all been developed to increase model performances in different ways, will be presented.

In the following, a consistent denomination for scalar values (a), vectors (\vec{a}), matrices (A), and tensors (\mathbf{A}) is used.

3.1 The Perceptron

The perceptron was introduced by Frank Rosenblatt in his work *The Perceptron: A probabilistic model for information storage and organization in the brain* in 1958 (Rosenblatt, 1958) and is a simple artificial neural network (Spektrum, 2017). Figure 3.1 shows the basic structure of a single perceptron. In literature, the term *perceptron* is used in two different ways: In some sources, it is already used for one single element like it is depicted in Figure 3.1, e.g. (Kramer, 2020), and in other sources, e.g. (Géron, 2019), the single element is called an *threshold logic unit* (TLU) and the arrangement of multiple TLUs in parallel sharing the same inputs is called a perceptron. In this work, everything from 1 up to n artificial neurons in parallel will be referred to as *perceptron*. Many scholars also use the terms *neuron* and *perceptron* interchangeably.

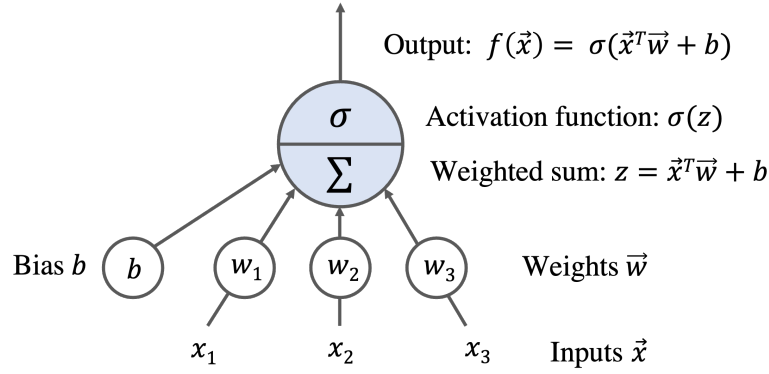


Figure 3.1 **The perceptron**

An input vector \vec{x} is mapped from \mathbb{R}^3 to \mathbb{R} by the means of calculating the weighted sum $\vec{x}^T \vec{w} + b$ and passing it through an activation function σ . Figure inspired by (Géron, 2019).

A single unit perceptron takes in an input vector \vec{x} of dimension d , calculates the dot product of this input with a weight vector \vec{w} of dimension d , and usually adds a bias term b , which is scalar. In the case of a perceptron with multiple units, the weight vector \vec{w} becomes a weight matrix W , and the bias becomes a bias vector \vec{b} containing a bias term for every single unit. The weighted sum, given by $\vec{x}^T \vec{w} + b$, is then fed into an activation function σ , calculating the final output. The motivation behind activation functions will be discussed further in section 3.2.

An easy practical example for using a perceptron is the modeling of the logical components AND and OR with two inputs. A logical AND can be modeled by setting the weights w_1 and w_2 for each input to 0.5. The bias will be set to -0.6 . As activation function, a *Heaviside step function* given by the following term is chosen:

$$x \mapsto \begin{cases} 0 : & \vec{x}^T \vec{w} + b < 0 \\ 1 : & \vec{x}^T \vec{w} + b \geq 0 \end{cases} \quad (3.1)$$

This way an input vector $\vec{x} = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$ will result in

$$\vec{x}^T \vec{w} + b = \begin{pmatrix} 1 & 0 \end{pmatrix} \times \begin{pmatrix} 0.5 \\ 0.5 \end{pmatrix} - 0.6 = -0.1 \quad , \quad (3.2)$$

which is < 0 and the result will be 0. Table 3.1 depicts the results of all input combinations, showing that this model perfectly reproduces a logical AND.

Table 3.1 **Outputs of perceptron model for logical AND**

		\mathbf{x}_1	
		0	1
\mathbf{x}_2	0	0	0
	1	0	1

To model a logical OR we only need to adjust the bias term to -0.1 , yielding the results given in Table 3.2.

Table 3.2 **Outputs of perceptron model for logical OR**

		\mathbf{x}_1	
		0	1
\mathbf{x}_2	0	0	1
	1	1	1

The above example shows the capability to model different underlying functions using a simple one-unit perceptron. Another use case of a single perceptron could be the classification of iris flowers using only their petal length and width. Although the method works great in linearly separable classes, it fails at other simple tasks like modeling an EXCLUSIVE OR (XOR), which is not linearly separable (Table 3.3) and will be elaborated in section 3.2. This was stated by Marvin Minsky and Seymour Papert in 1969 and has led to a harsh decrease in fundings for research activities related to neural networks (Géron, 2019). Some sources like (Wikipedia, 2020) refer to this period as the first *AI winter*, meaning research fundings were cut drastically for years only until new discoveries and methods have been made/developed that gathered new interest.

Table 3.3 **Output table of logical XOR**

		\mathbf{x}_1	
		0	1
\mathbf{x}_2	0	0	1
	1	1	0

Later, it has been shown that the XOR problem can be solved by stacking multiple layers of perceptrons on top of each other, leading to the introduction of a new class of artificial neural networks: The *Multi-Layer Perceptron* (Géron, 2019). This concept will be elaborated in the following.

3.2 Multi-Layer Perceptron

The stacking of multiple layers of perceptrons with multiple units is called a *multi-layer perceptron* (MLP). In its most basic form, it consists of an input layer X , one *hidden layer* H , and an output layer O (A. Zhang, Lipton, Li, & Smola, 2021). Figure 3.2 depicts a MLP with three input units, four hidden units in the hidden layer, and two output units in the output layer. A setup like this is referred to as a

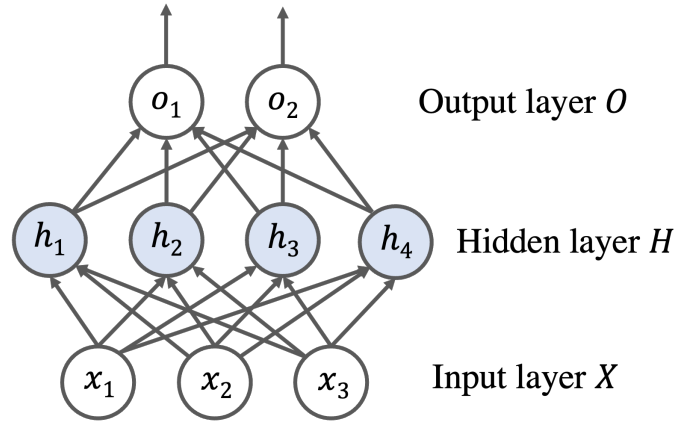


Figure 3.2 **Multi-layer perceptron with two layers**

An MLP with three input units $[x_1, x_2, x_3]$, a hidden layer H with four hidden units $[h_1, \dots, h_4]$ and an output layer O with two output units $[o_1, o_2]$.

two-layer MLP because only two layers – namely the hidden layer and output layer – do any calculations. Note that all neurons in the hidden layer are connected to all neurons of the previous layer (input) – such layers are also called *dense layers* or *fully connected layers* (Géron, 2019). The same applies to the output layer. If we neglect the activation functions in all the neurons, adding hidden layers, respectively, stacking multiple layers does not enable the model to depict non-linear relationships. The reason is that the hidden units in H are only an affine function of its inputs given by $\vec{h} = \vec{x}W_1 + \vec{b}_1$, and the outputs are only an affine function of the hidden units $\vec{o} = \vec{h}W_2 + \vec{b}_2$. Therefore we can rewrite the mapping from \vec{x} to \vec{o} in the following way:

$$\vec{o} = (\vec{x}W_1 + \vec{b}_1)W_2 + \vec{b}_2 = \vec{x}W_1W_2 + \vec{b}_1W_2 + \vec{b}_2 = \vec{x}\mathbf{W} + \vec{b} \quad (3.3)$$

This is again a simple linear single layer model, and we gained nothing with the introduction of additional layers. (A. Zhang et al., 2021)

To introduce non-linearity to the model, a non-linear activation function σ – e.g., the Heaviside step function – is reapplied elementwise to each neuron in the hidden layer

H . The output of our simple MLP is now given by $\vec{h} = \sigma(\vec{x}W_1 + \vec{b}_1)$ and $\vec{o} = \vec{h}W_2 + \vec{b}_2$ which cannot be reformulated back into a simple one-layer linear model. The model can now depict non-linear functions like the logical XOR (Géron, 2019). Figure 3.3 shows a possible MLP architecture to solve the XOR problem.

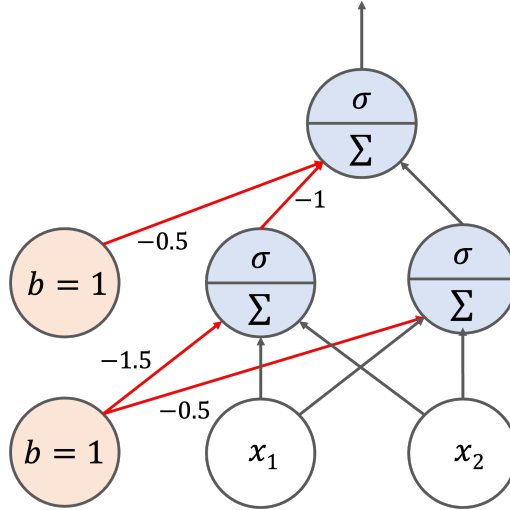


Figure 3.3 **Multi-layer perceptron modeling logical XOR**

Activation function σ is given by the Heaviside step function (Equation 3.1). Logical inputs are denoted with x_1, x_2 , and bias b . Grey connections are associated with a weight of 1. Red connections are associated with different weights shown next to them. Table 3.3 depicts all outputs for all input combinations. Figure inspired by (Géron, 2019)

MLPs are universal approximators, meaning that, in theory, a single-hidden-layer model with enough nodes can represent any possible function (A. Zhang et al., 2021). Practitioners and theorists refer to the addition of more hidden layers as making an artificial neural network *deeper*. The term *deep learning*, therefore, refers to neural networks with a large number of hidden layers (Goodfellow, Bengio, & Courville, 2016). A specific number of hidden layers that discriminates between *deep learning* and *traditional* neural networks is not defined.

This will finish the explanations on the different kinds of perceptrons. The following subsection 3.2.1 will introduce some of the most common activation functions in deep learning.

3.2.1 Activation Functions

To better understand activation functions, which provide the non-linearity in neural networks, and their different characteristics, a few of them which are popular in machine learning will be briefly described in a listed fashion. The most important one is the *ReLU* function for its broad usage and convenient properties. For example, the later architecture that will be used for the prediction of pipelines is exclusively using activations calculated with the ReLU function (Chaurasia & Culurciello, 2017). The *softmax* function that is often applied to the output neurons to yield the final prediction scores of a model will take up the last part of this section.

- **Sigmoid function**

From a historical perspective, the motivation behind the first neural networks was to resemble biological neurons, which either *fire* or *not fire*. This encouraged thresholding functions like the *Heaviside step function*, which can be designed to output either TRUE (*fire*) or FALSE (*not fire*). Later, "when attention shifted to gradient-based learning, the sigmoid function was a natural choice because it is a smooth, differentiable approximation to a thresholding unit" (A. Zhang et al., 2021). In short, the *gradient-based learning* approach differentiates the model errors in respect to every weight and propagates the obtained gradients back through the network to update its weights. This will be described in the following subsection 3.2.2 in far more detail. However, it is important to understand the idea behind *back-propagation* to then understand the problem of *vanishing gradients*, which is a drawback of the sigmoid function as an activation function (Nwankpa, Ijomah, Gachagan, & Marshall, 2018).

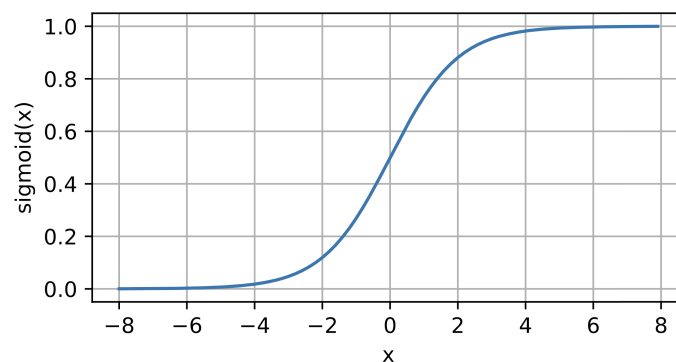


Figure 3.4 **Sigmoid activation function**

The sigmoid activation function, as shown in (A. Zhang et al., 2021).

The sigmoid function is given by

$$\sigma(x) = \frac{1}{1 + \exp(-x)} \quad (3.4)$$

(Goodfellow et al., 2016). It maps inputs from the space of real numbers \mathbb{R} to numbers between 0 and 1 and is centered at 0.5. Figure 3.4 shows that large positive and negative inputs return numbers close to 0 or 1 with gradients approaching 0. This means that only very small numbers, prone to numerical errors, will be propagated back to preceding layers of the network. These numbers are also often getting increasingly lower as the back-propagation algorithm reaches the lower layers, which results in neglectable weight updates and no convergence to good model solutions, describing the heart of the vanishing gradients problem (Géron, 2019).

- **Tanh function**

The Tanh function maps its inputs in a similar fashion to the sigmoid function by compressing the input which comes from the space of real numbers \mathbb{R} into a range between -1 and 1 (Figure 3.5) (A. Zhang et al., 2021). Furthermore, it is also smooth, differentiable, and an approximation of a thresholding unit. It is given by

$$\text{Tanh}(x) = \frac{1 - \exp(-2x)}{1 + \exp(-2x)} \quad (3.5)$$

(A. Zhang et al., 2021).

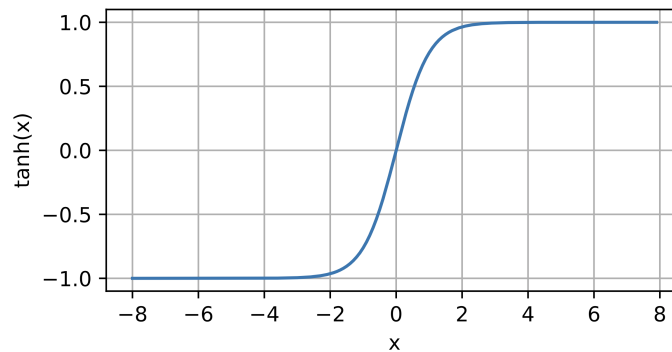


Figure 3.5 **Tanh activation function**

The tanh activation function, as shown in (A. Zhang et al., 2021).

It differs from the sigmoid function in its centering at 0 and its wider output range, which comparatively improves the training of multi-layer neural networks. Therefore, it is preferred as an activation function but also suffers from the problem of vanishing gradients (Nwankpa et al., 2018).

- **ReLU function**

ReLU stands for *rectified linear unit* and is given by

$$\text{ReLU}(x) = \max(0, x) \quad (3.6)$$

(Kramer, 2020). Figure 3.6 shows a plot of this simple nonlinear transformation. It is easy and fast to compute. In comparison to the Sigmoid and Tanh activation functions, it has the advantage that the gradients are well behaved because for all output values > 0 , the derivative is 1, and for all output values < 0 , the derivative is 0, which mitigates the problem of vanishing gradients.

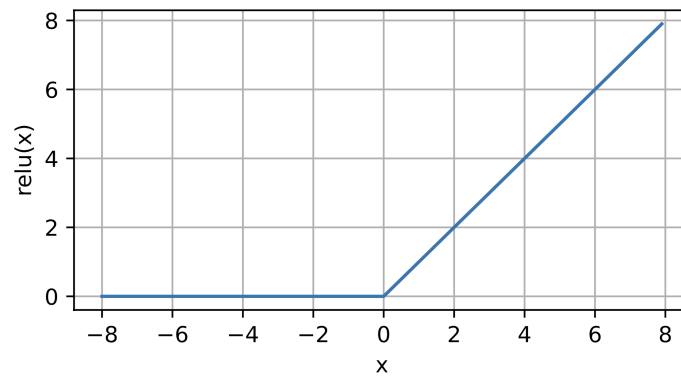


Figure 3.6 **ReLU activation function**

The ReLU activation function, as shown in (A. Zhang et al., 2021).

In the special case of $\text{ReLU}(x) = 0$, which is not differentiable, the gradient defaults to 0. These properties – fast and easy computation and mitigation of vanishing gradients – make ReLU the most popular activation function in the field of machine learning (A. Zhang et al., 2021). There are further activation functions closely related to ReLU, like, for example, *leaky ReLU* and *ELU*, (Géron, 2019), just to mention a few.

- **Softmax function**

The softmax function is only used in classification problems and applied to the output of the final layer of an artificial neural network. It transforms the output of the last C neurons, where C equals the number of possible classes, into a range between 0 and 1 with the property that the values of all classes combined add up to 1. Thus, it can be interpreted as the probability for each class (Nwankpa et al., 2018). Its formula is given by

$$\text{softmax}(x_i) = \frac{\exp(x_i)}{\sum_j \exp(x_j)} \quad , \quad (3.7)$$

where x_i represents the output of a single neuron in the output layer and $\sum_j \exp(x_j)$ the sum over the exponentials of all outputs in the final layer (Goodfellow et al., 2016).

3.2.2 Training of Artificial Neural Networks

The goal of training neural networks in the field of semantic image segmentation is to minimize the divergence between the prediction of a neural network and the *ground truth* on a pixel-wise level (A. Zhang et al., 2021). The ground truth includes the labels, respectively, classification for all pixels and inputs in the training and test data set like *pipeline* and *background* or *road*, *building*, *car* and *pedestrian*. Often the ground truth is generated by hand, which is laborious work (A. Zhang et al., 2021). In some cases, automated methods can be applied, for example, when pipeline courses are provided as shapefiles and only have to be merged with the respective satellite data. Of course, the quality of the shapefiles has to be inspected beforehand.

The divergence between a model's prediction and the the ground truth can be quantified with a loss function L . A common loss function in classification problems is the cross-entropy loss given by

$$L(\vec{x}, \text{class}) = -\log\left(\frac{\exp(\vec{x}[\text{class}])}{\sum_j \exp(\vec{x}[j])}\right) . \quad (3.8)$$

It is a function of input \vec{x} , which are the unnormalized scores for each class coming from the neural network, and its respective true *class* index, which is between $[0, C - 1]$, where C is the number of possible classes (Torch Contributors, 2019). If the model outputs a high score for the true class and small scores for the wrong classes, the ratio between the numerator and denominator of $\exp(\vec{x}[\text{class}]) / \sum_j \exp(\vec{x}[j])$ tends towards 1, which results in a L value close to 0. Given that the model predicts the wrong class label – meaning high scores for wrong classes and a low score for the true class – the fraction will tend towards 0 resulting in a high L value.

Training a neural network now means that we adapt its weights \vec{w} to minimize L . The most popular method for this is to perform gradient descent on \vec{w} with respect to L . The gradients $\nabla L(\vec{w})$ have the following form:

$$\nabla L(\vec{w}) = \begin{pmatrix} \frac{\partial L(\vec{w})}{\partial w_1} \\ \frac{\partial L(\vec{w})}{\partial w_2} \\ \vdots \\ \frac{\partial L(\vec{w})}{\partial w_k} \end{pmatrix} , \quad (3.9)$$

where k is the number of all weights (Kramer, 2020). The calculation of the gradients is done using the *back-propagation* algorithm. It very efficiently computes the chain rule for a computational graph, like a neural network, which is needed to determine the gradient of each weight in regard to all following nodes. The algorithm is called back-propagation because it propagates the calculated loss back through the network to calculate the derivatives, respectively, gradients in contrast to forward propagation, which is performed when the model evaluates an input (Goodfellow et al., 2016). The calculated gradients are then applied in a weight update to \vec{w} with $\vec{w}' = \vec{w} + \Delta\vec{w}$. $\Delta\vec{w}$ is given by $-\eta\nabla L(\vec{w})$, where η is the *learning rate* (Kramer, 2020), which is an important *hyperparameter* determining the training characteristics. Figure 3.7 provides a visual explanation of why the weight update is directed against the gradient ($-\eta$).

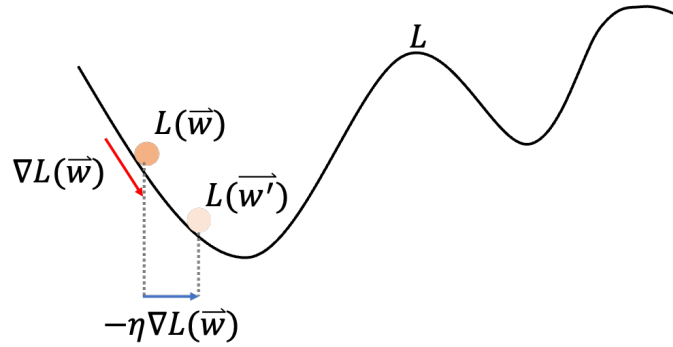


Figure 3.7 **Weight update**

The gradient at \vec{w} is indicated by the red arrow and is going downwards (negative). In order to make a sensible weight update, we have to move the weights against the gradient in the positive direction, implemented by the addition of the *minus* sign in $-\eta\nabla L(\vec{w})$.

The above-explained gradient descent method is often referred to as *vanilla gradient descent*. A well-established and powerful update to this method is *Adam*. Adam stands for *Adaptive momentum estimation* and was introduced in 2014 by Diederik P. Kingma and Jimmy Ba. Instead of only taking $\nabla L(\vec{w})$ into consideration when updating the weights of a neural network, Adam makes use of adaptive estimates of lower-order moments (Kingma & Ba, 2017), redefining $\Delta\vec{w}_i$ for each weight as

$$\Delta\vec{w}_i = -\frac{\eta m'_i}{\sqrt{v'_i + \epsilon}} \quad (3.10)$$

m' is an approximation of the first-moment estimate of gradients and given by $m' = \beta_1 + (1 - \beta_1)\nabla L(\vec{w})$. Analyzing the formula, we see that Adam includes the present $\nabla L(\vec{w})$ and takes advantage of past gradients multiplied with exponential decay rate

β_1 , making the updates more stable. v' is calculated by $v' = \beta_2 + (1 - \beta_2)\nabla L^2(\vec{w})$ including the uncentered variance (second moment) of $\nabla L(\vec{w})$. The decay rate β_2 is by recommendation set close to 1, limiting the impact of past gradient variances strongly. $\epsilon \in \mathbb{R}^+$ is introduced to avoid divisions by 0 (Kramer, 2020).

To conclude this section, the terms *epoch*, *stochastic gradient descent*, and *mini-batch* are introduced: An *epoch* represents the presentation of all training samples in a training dataset to a neural network. In order to converge, models have to be trained for multiple epochs, meaning each sample is presented multiple times. In the past, some training strategies only performed weight updates after each epoch resulting in a prolonged but stable convergence of the models. Contrasting this practice, the *stochastic gradient descent* performs weight updates after each training sample. This results in efficient training routines (Goodfellow et al., 2016). An often used compromise is the *mini-batch* mode, where the model parameters are updated after the presentation of every e.g., 32 or 64 training samples. An exact number of how many samples a mini-batch should consist of is not available and depends on the individual application. Mini-batch has numerous advantages over the other two approaches because it is fast, stable, and can also be used to optimize memory and computational efficiency (Ioffe & Szegedy, 2015).

3.3 Convolutional Layers

As mentioned beforehand, it is possible to represent every function with a single layer MLP. However, this approach would be extremely costly in terms of memory and computation requirements. To counteract this problem, it is sensible to take advantage of the domain knowledge we have about the functions we are trying to capture with our models: In the field of computer vision the subject of matter are exclusively images. This has led to the development of so-called convolutional layers, which were first successfully applied in (LeCun et al., 1989). They provided a breakthrough in computer vision (Kramer, 2020) due to their high gains in prediction performance and parameter efficiency compared to traditional neural networks (Krizhevsky, Sutskever, & Hinton, 2012). To the author's knowledge, every modern state-of-the-art architecture aimed at vision tasks makes extensive use of convolutional layers.

Convolutional layers are translation invariant feature detectors, meaning that their output does not change regardless of the position/shift of an image feature, like for example a human eye. They are applied in a sliding window fashion (see Figure 3.8 and Figure 3.9) across an entire input volume (Kramer, 2020) of dimensions

$H' \times W' \times C'$, where H' is the height in px, W' the width in px, and C' the number of channels. An RGB image would, for example, have 3 channels in the first layer.

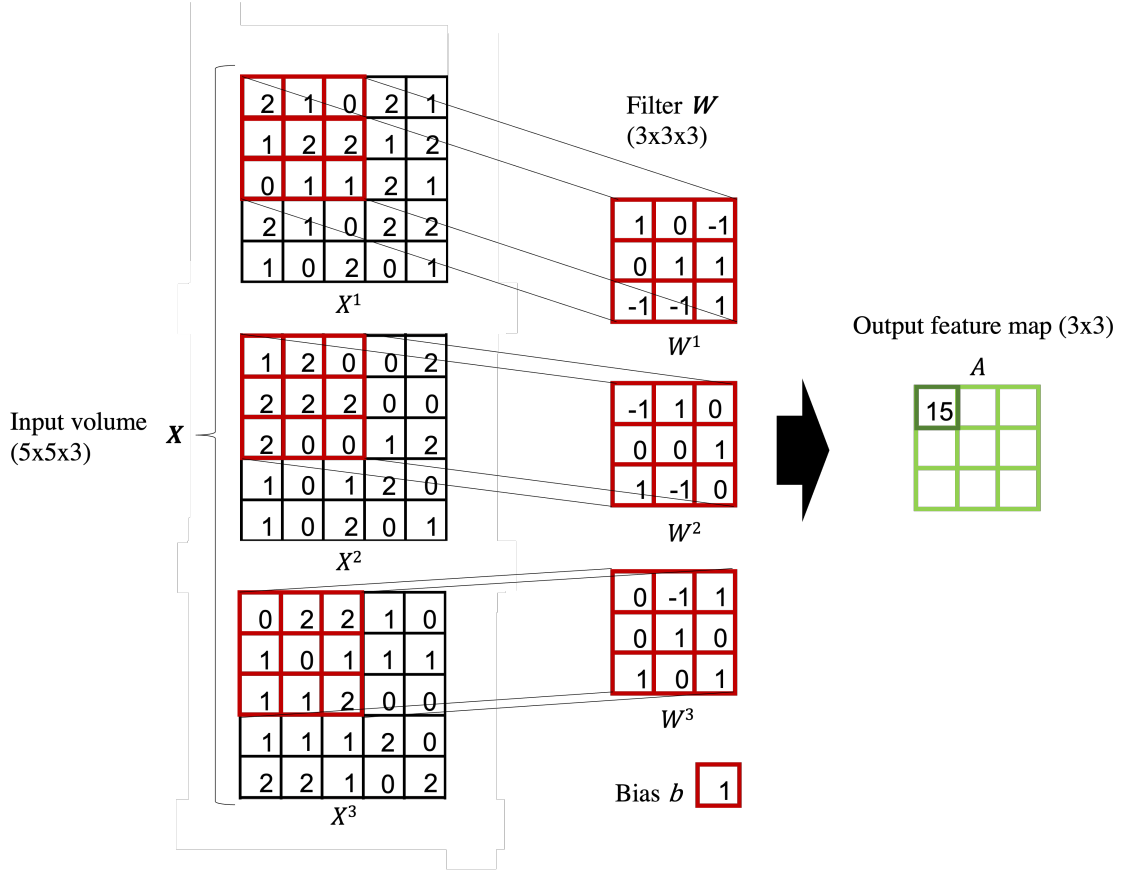


Figure 3.8 **Convolution operation at first filter position**

The red square represents the filter position $(i,j) = (1,1)$ over the input volume. Input values and weights are depicted by the numbers inside the squares. The output of the convolutional process at the current filter position is depicted in dark green on the right.

A convolutional layer, therefore, maps an input $\mathbf{X} \in \mathbb{R}^{H' \times W' \times C'} = [X^1, X^2, \dots, X^{C'}]$ to an output $\mathbf{A} \in \mathbb{R}^{H \times W \times C} = [A^1, A^2, \dots, A^C]$, which is then referred to as *feature maps*. This is done by using C filter kernels \mathbf{W} . Each filter kernel $[\mathbf{W}_1, \mathbf{W}_2, \dots, \mathbf{W}_C]$ consists of C' weight matrices $\mathbf{W}_c = [W_c^1, W_c^2, \dots, W_c^{C'}]$ where each matrix W_c^s is a two dimensional kernel in case of *2D convolution*. The activations of a single output A_c is given by

$$A_c = \mathbf{W}_c * \mathbf{X} = \sum_{s=1}^{C'} W_c^s * X^s \quad (3.11)$$

and $*$ being the convolutional operation

$$a_c(i,j) = \sum_{k=1}^m \sum_{l=1}^m w_c^s(k,l) \times x^s(i+k-1, j+l-1) \quad (3.12)$$

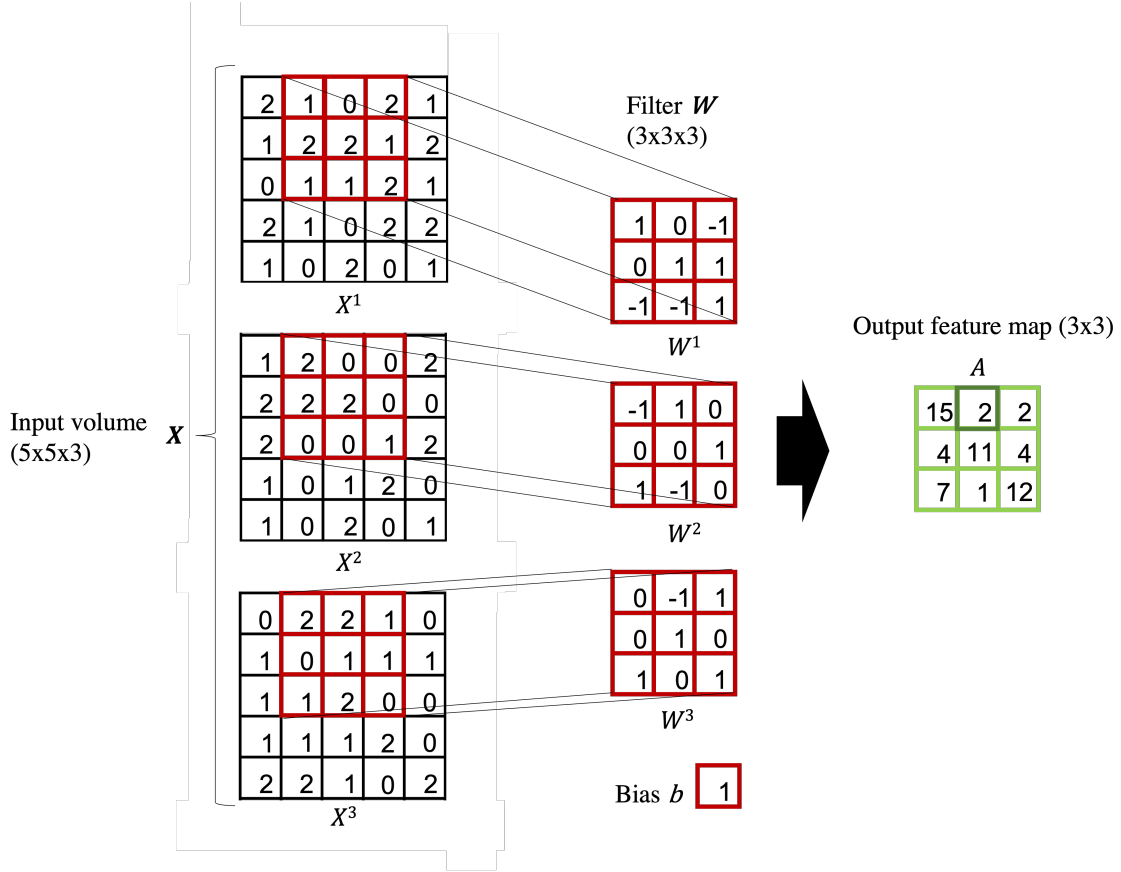


Figure 3.9 **Convolution operation at second filter position**

The red square represents the filter position $(i,j) = (2,1)$ over the input volume. Input values and weights are depicted by the numbers inside the squares. The final output of the whole convolutional process is depicted in green on the right for completeness. The result of the current convolutional process is marked with dark green.

with the $m \times m$ filter kernel W_c (Kramer, 2020). To better understand the convolutional operation, the examples in Figure 3.8 and Figure 3.9 are provided. To calculate the first entry of the output feature map marked with dark green ($a_c(1,1)$) in Figure 3.8, we have to calculate the dot product of the flattened input array at the position covered by the filter kernel and the flattened filter kernel itself to get the result 15. Note that the dot product is equal to $\sum_{k=1}^m \sum_{l=1}^m w_c^s(k,l) \times x^s(i+k-1, j+l-1)$ if the input matrices get flattened. Next, we calculate the second entry ($a_c(1,2)$) of the output feature map shown in Figure 3.9 by sliding the filter kernel one step to the right and performing the same operation. This gets repeated until the whole output feature map A_c is calculated. The addition of further filter kernels would create additional output feature maps.

Typically, the number of filter kernels is doubled in each convolutional layer. For example, an input with dimensions (28,28,128) would be transformed into an output

of (26,26,256). This results in a decrease in spatial resolution of the input image but an increase in feature space, allowing the model to capture more complex image features.

Two important hyper parameters when performing convolutions are *stride* and *padding*. *Stride* determines by how many elements or pixels a filter is shifted after each operation and can be split up into vertical and horizontal stride. If only one number is given, the horizontal and vertical stride are the same. Figure 3.10 provides a visual example. Usually, the stride is set to 1, meaning the filters are only shifted by one unit at a time, which works well in practice. Using a bigger stride allows the model to create spatially smaller output volumes (Stanford CS231, 2021) and shrinks the number of computational operations needed in the following layers. Some architectures use this technique as an alternative to *pooling layers*, which will be explained in the next section.

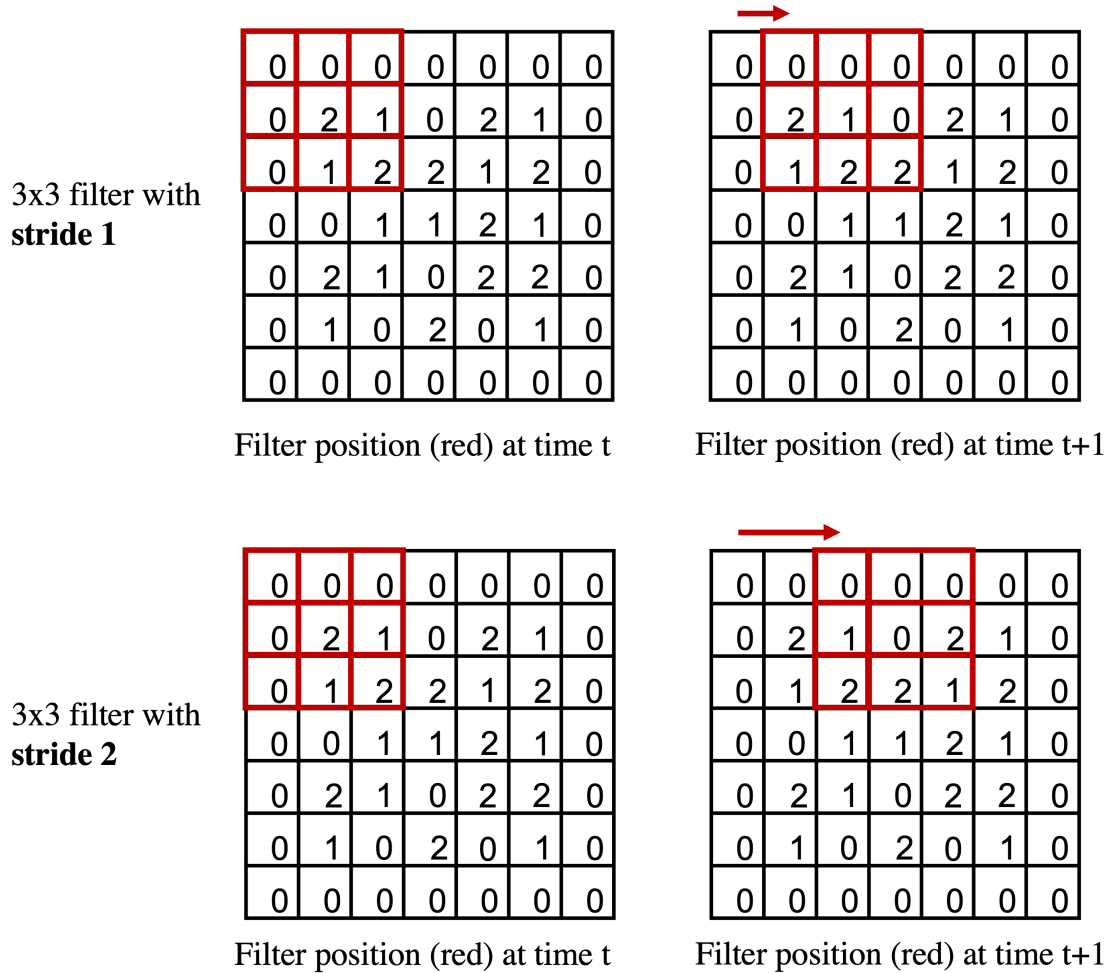


Figure 3.10 **Comparison stride=1 and stride=2**

The second mentioned hyper parameter, *padding*, is used to determine the number

of pixels added around the borders of an input (see Figure 3.11). This counteracts the decrease in spatial output size and loss of information at the borders of an image when applying a convolutional filter to a volume. For example, when making use of 3×3 convolutions, it can be advantageous to add a pixel with value 0 – which is called *zero padding* with $\text{padding} = 1$ – around the borders of an image, resulting in the same input and output size (Stanford CS231, 2021).

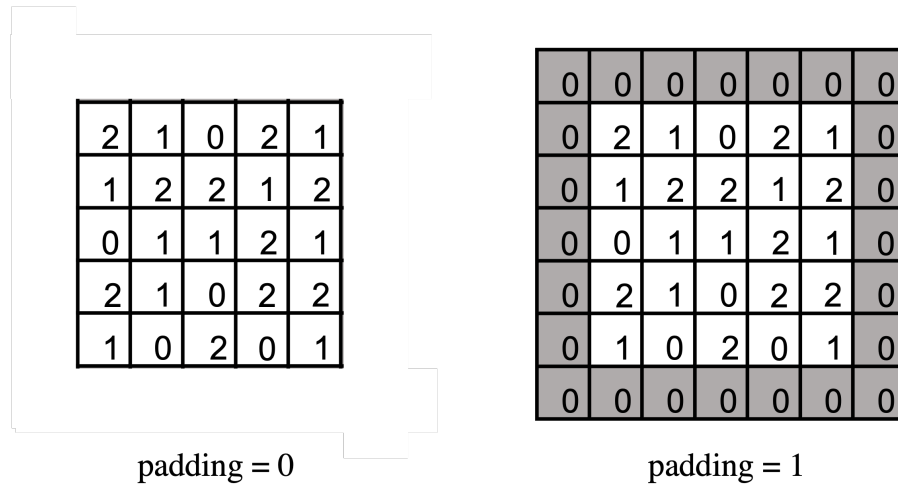


Figure 3.11 **Zero padding**

Left picture shows the input image with no padding and right picture shows the same input image with padding equal to 1 in grey.

The following sections will introduce additional concepts heavily used in artificial neural network and especially in convolutional neural networks.

3.4 Max-Pooling

In order to progressively decrease the spatial size of an input volume – which reduces the number of parameters needed in a network and counteracts the problem of overfitting to the training data – many modern deep learning architectures make use of *pooling* operators.

These operators take an input I with dimensions $W_1 \times H_1 \times C$, where W is the width, H the height, and C the number of channels, and downsample it to $W_2 \times H_2 \times C$ (Figure 3.12). Comparably to convolutional filter kernels, a *pooling layer* applies an $n \times n$ filter in a sliding window fashion with stride s to the input I . In the case of *max-pooling*, this filter then only extracts the highest value inside its scope to output it to the next layer. The dimensions of the downsampled input are given by $W_2 = (W_1 - n)/s + 1$ and $H_2 = (H_1 - n)/s + 1$. The number of channels

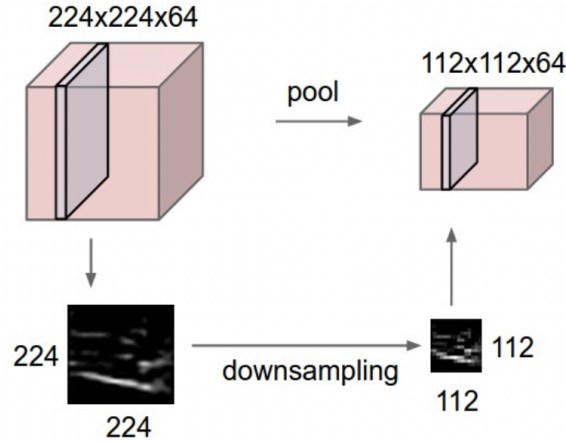


Figure 3.12 **Downsampling with max-pooling**

An input I of dimensions $W_1 \times H_1 \times C = 224 \times 224 \times 64$ is downsampled to $W_2 \times H_2 \times C = 112 \times 112 \times 64$, quartering the number of inputs to the next layer. As shown in (Stanford CS231, 2021).

C stays unchanged. In many practical cases, n will be set to 2 and the stride s to 2, as shown in Figure 3.13 (Stanford CS231, 2021).

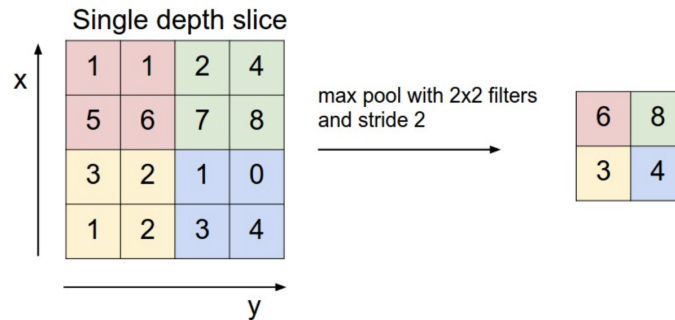


Figure 3.13 **Max-pooling operation**

A 2×2 max-pooling filter with stride 2 is applied to a 4×4 input. The output dimensions are given by $W_2 = (W_1 - n)/s + 1 = (4 - 2)/2 + 1 = 2$ and $H_2 = (H_1 - n)/s + 1 = (4 - 2)/2 + 1 = 2$. As shown in (Stanford CS231, 2021).

Another pooling operation would be the *average pooling* which computes the average of all values in the scope of the pooling filter.

Regarding the process of back-propagation, the maximum activation indices generated in the forward passes are kept to then route the gradients back to their respective weights during training (Stanford CS231, 2021).

3.5 Batch Normalization

Batch normalization was introduced by Sergey Ioffe and Christian Szegedy in 2015. Their paper was aimed at the problems that arise out of the changing input distributions of each layer during training "as the parameters of the previous layers change" – namely low learning rates and careful parameter initialization resulting in longer training times (Ioffe & Szegedy, 2015). Their strategy was to implement a normalization step preceding all hidden layers that takes in all activations over a mini-batch y_i , calculates their mean μ_y and standard deviation σ^2 to then determine the normalized y_i , which is denoted as y'_i :

$$y'_i = \frac{y_i - \mu_y}{\sqrt{\sigma^2 + \epsilon}} \quad . \quad (3.13)$$

ϵ is some small positive error term. The obtained activations will be scaled by $\gamma \in \mathbb{R}^+$ and shifted with $\beta \in \mathbb{R}$, which are both learned during training. The final result of the batch normalization is given by

$$y''_i = \gamma \times y'_i + \beta \quad . \quad (3.14)$$

This stabilizes the layer inputs and aids the training process. A positive side effect of the strategy is that due to the shifting, the biases can be neglected, resulting in fewer network parameters (Kramer, 2020). It was also shown that batch normalization regularizes the model and mitigates the problem of overfitting to training data, which would otherwise result in deteriorated generalization capabilities toward the test data (Ioffe & Szegedy, 2015).

3.6 Transposed Convolutions

The last concept discussed in this section is tightly entangled to the field of image segmentation and is a special case of the convolutional operation: the *transposed convolution* (A. Zhang et al., 2021). It is used to upsample feature maps back from feature space to the original or close to the original image resolution to get the final segmented output. Many papers refer to transposed convolutions as *up-convolutions*. After describing the convolutional process in detail in section 3.3, Figure 3.14 should be sufficient to understand the process of transposed convolution.

Like in the normal convolution operation, it is also possible to set the parameters stride and padding, which will determine the output size of the transposed convolution. Figure 3.15 shows an example of a 2×2 input upsampled with stride 2 and

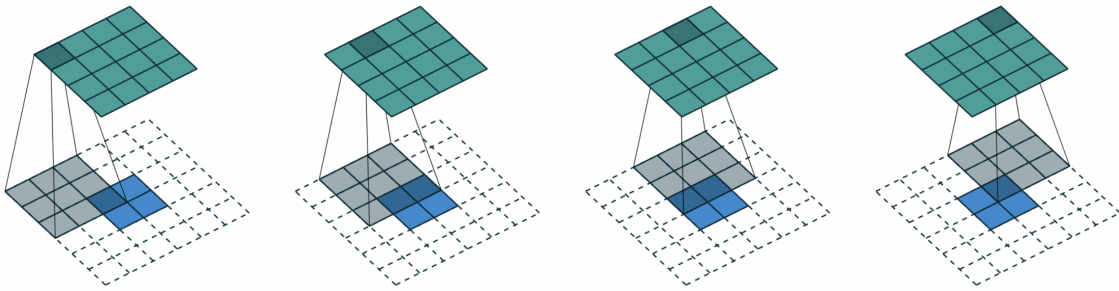


Figure 3.14 **Transposed convolution**

A transposed convolution with padding 2 and stride 1 is performed on a 2×2 input (blue) with a 3×3 filter kernel (grey), resulting in a 4×4 output (green). As shown in (Dumoulin & Visin, 2018).

padding 2.

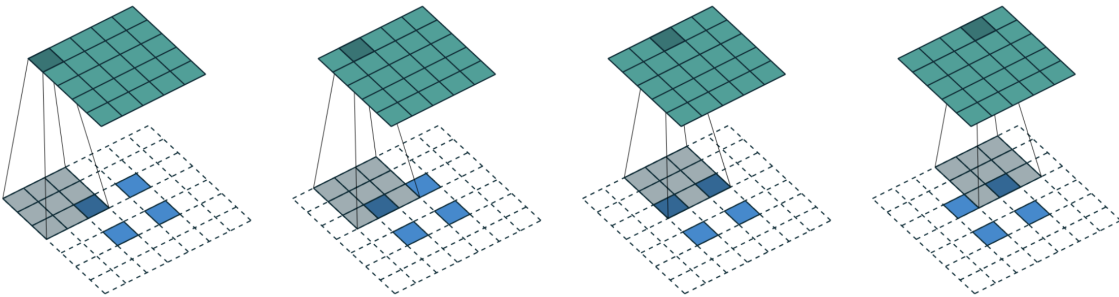


Figure 3.15 **Stride and padding in transposed convolutions**

A transposed convolution with padding 2 and stride 2 is performed on a 2×2 input (blue) with a 3×3 filter kernel (grey), resulting in a 5×5 output (green). Note that the stride is conceptually not applied to the filter but to the input creating space between all input pixels. As shown in (Dumoulin & Visin, 2018).

4 Uncertainty in Deep Learning

Due to the fast advancements and vast successes of deep learning algorithms in various fields of application, they are increasingly used in real-life systems. Common examples in the literature are autonomous vehicles (Kuutti, Bowden, Jin, Barber, & Fallah, 2021), the detection of cancerous cells (Shen et al., 2019), and high-frequency trading (Arévalo, Niño, Hernández, & Sandoval, 2016). A challenging problem connected to this development is the lack of said algorithms to capture model uncertainty, providing overconfident predictions for unknown data domains. This is especially crucial in safety-critical applications (Gal & Ghahramani, 2016). Figure 4.1 provides a toy example of this problem.

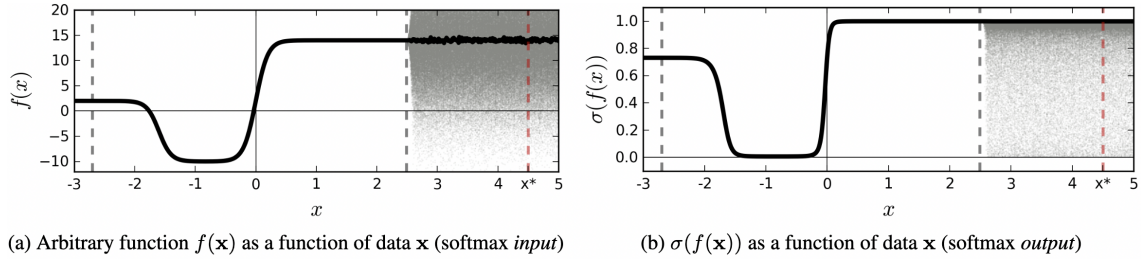


Figure 4.1 **Model uncertainties with softmax activation**

A model is trained on an arbitrary function $f(x)$ to discriminate between two classes (0 and 1) with training data provided between the grey dashed lines. Function uncertainty is given by the shaded area. x^* denotes a single point outside of the training data. The softmax output $\sigma(f(x))$ predicts x^* to be of class 1 with probability 1 ignoring the function uncertainty. The model therefore extrapolates with unreasonable high confidence outside the scope of its training data. Passing the distribution, respectively, function uncertainty through the softmax function σ would yield better estimates of the real model uncertainty (Gal & Ghahramani, 2016).

Often in deep learning, a distinction between *aleatoric* (statistical) and *epistemic* (systematic) uncertainty is introduced:

- **Aleatoric Uncertainty**

Aleatoric uncertainty corresponds to the naturally existing randomness or

noise arising from the fundamental variability of all experiments, measurements, etc. (Hüllermeier & Waegeman, 2021). This means that the provision of more data does not decrease the aleatoric uncertainty (Gal, 2016). In the case of image segmentation tasks, this kind of uncertainty is explicitly made available. By means of a Softmax function, the models output a categorical distribution over all classes. In cases where the input is inherently ambiguous, the Softmax outputs are more evenly distributed over several classes. This represents low model confidence in its prediction and high aleatoric uncertainty (Gustafsson, Danelljan, & Schon, 2020).

- **Epistemic Uncertainty**

Epistemic uncertainty stems from the lack of knowledge about a system (Bjarnadottir, Li, & Stewart, 2019), which in the case of an artificial neural network would be its parameters θ (Gustafsson et al., 2020). By increasing the available training data and, therefore, the amount of information about an underlying function, the epistemic uncertainty can be reduced (Hüllermeier & Waegeman, 2021). Each model parameter realization, respectively, function stems from a distribution of possible parameter realizations, respectively, functions and has a certain probability. Therefore, it is possible to estimate the epistemic uncertainty by extracting the variance between multiple function realizations (Gal, 2016).

This chapter covers two approaches to extract epistemic model uncertainty in deep learning: *Bayesian Deep Learning* and *Deep Ensembles*. Both approaches have already been used and tested extensively, with *Bayesian Deep Learning* being the more popular one in the current scientific literature. The chapter ends with the introduction of *active learning*, which takes advantage of available model uncertainties to minimize future efforts when creating additional training datasets. This can be advantageous when increasing the scope of the pipeline detection model to new geographical regions.

4.1 Bayesian Deep Learning

Instead of regarding the parameters of a NN to be single-point estimates, *Bayesian deep learning* considers the weights of a NN to be distributions of possible parameter realizations, which is used to express its epistemic uncertainty. According to *Bayes'*

theorem, the posterior distribution is given by

$$p(\theta|D) = \frac{p(D|\theta)p(\theta)}{p(D)} \quad , \quad (4.1)$$

where D is the available data and θ the model parameters. The distribution is characterized by the data likelihood $p(D|\theta)$ and the chosen prior $p(\theta)$ and updated when new data is available: This is called *Bayesian inference*. The uncertainty of the model parameters is then marginalized out, which results in the predictive posterior distribution:

$$p(y^*|x^*, D) = \int p(y^*|x^*, \theta)p(\theta|D)d\theta \quad , \quad (4.2)$$

where y^* and x^* represent a single output and input sample. The integral is intractable, and sampling from the true posterior distribution is practically not possible. Therefore, different methods to approximate the true posterior distribution have been developed (Gustafsson et al., 2020). A very efficient one was introduced by Gal et al. in (Gal & Ghahramani, 2016). Their work has proven that the use of *Monte Carlo dropout* is a valid approximation of the true posterior distribution. Monte Carlo dropout describes the method of setting each single weight in each layer to 0 in each forward pass through a NN with probability p at evaluation time. The output of a NN using Monte Carlo dropout is obtained by passing a single sample multiple times through the network and calculating the average over all forward passes. The uncertainty estimates can, for example, be derived from the standard deviation σ of the outputs of all forward passes like in (Zech & Ranalli, 2020), or their variance σ^2 like in (Hartmann et al., 2021). Monte Carlo dropout can also be understood as the combination and averaging over different NNs who share their parameters (Lakshminarayanan, Pritzel, & Blundell, 2016). This leads to an alternative approach to Monte Carlo dropout: *Deep ensembles*.

4.2 Deep Ensembles

The idea of using deep ensembles to quantify epistemic model uncertainty was introduced in (Lakshminarayanan et al., 2016). Before, deep ensembles had only been used to boost the predictive performance of NN architectures. Their motivation was to develop a method to estimate predictive model uncertainty in a simple and scalable way. By running multiple realizations of the same point estimate NN architecture in parallel, they extracted state-of-the-art epistemic uncertainty estimates, which can be – given enough memory – scaled easily. Multiple different realizations of the same point estimate NN architecture are created by starting the training of

every single network with a random initialization of its weights and the shuffling of the training dataset. This way, the networks – also called *ensemble members* – are converging onto different local minima of the underlying function. The final prediction of an ensemble is given by its mean prediction

$$p(y|\vec{x}) = M^{-1} \sum_{m=1}^M p_{\theta_m}(y|\vec{x}, \theta_m) \quad , \quad (4.3)$$

where $p(y|\vec{x})$ is the probability of y given the input \vec{x} , M the number of ensemble members, and θ_m the parameters of each specific model m . The epistemic uncertainty of the prediction is then approximated by the variance σ^2 between the different model outputs $p_{\theta_m}(y|\vec{x}, \theta_m)$ (Lakshminarayanan et al., 2016).

In their work, the authors explicitly state that their approach is a non-Bayesian solution to the problem of uncertainty quantification in deep learning. (Gustafsson et al., 2020) compared deep ensembles to Bayesian approaches and argued that the different realizations of NNs used in ensembles could be interpreted as samples from the true posterior distribution. This would make deep ensembles also a Bayesian approach. Their experiments between deep ensembles and Bayesian dropout in terms of the quality of uncertainty quantifications and ease of implementation suggest that the use of deep ensembles is superior to Monte Carlo dropout. Therefore, deep ensembles will be used to estimate model uncertainties in this work.

4.3 Active Learning

The concept of *active learning* in deep learning is closely tied to the possibilities that arise out of the extraction of epistemic model uncertainty estimates with the above-described techniques. Because of the comparably high costs of labeling satellite imagery for image segmentation tasks, the idea of active learning can potentially decrease the efforts that have to be put into manually labeling new datasets for new regions.

Active learning – sometimes also referred to as *optimal experimental design* – describes the process of letting a model choose its training samples in order to optimize the training procedure and its outcomes. This is contrasted by the *default* approach in which a large number of samples in no particular order or composition is presented to the network. This is advantageous in cases where the amount of available, unlabeled data is high and the labeling itself is very costly – be it in time or money (Settles, 2009).

Thus, active learning enables the minimization of manual dataset labeling since only

samples of high *curiosity* to the model have to be labeled by the practitioner. Thus, the reduction of systematic model uncertainty is more sample efficient. A sensible measure to discriminate between samples that should be labeled and vice versa is the model uncertainty in regard to each sample. Like already described, deep ensembles will be used to estimate uncertainties in this work. Figure 4.2 demonstrates the performance enhancements due to an active learning approach compared to a randomly generated training dataset.

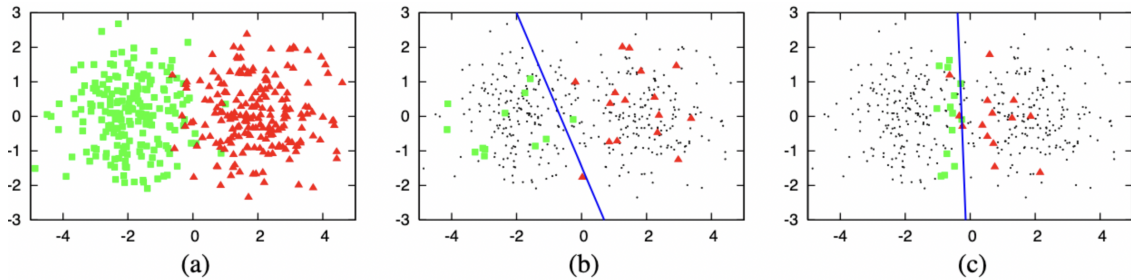


Figure 4.2 **Active learning: sample efficiency**

(a) A toy dataset with 400 samples from 2 classes sampled from gaussian distributions. (b) The suboptimal decision boundary (blue) gained with logistic regression on 30 randomly drawn labeled samples. (c) The decision boundary of a logistic regression model fitted on 30 actively chosen training samples using uncertainty sampling showing a high sample efficiency (Settles, 2009).

5 Model and data prerequisites

The preceding section 4.3 concludes the theoretical foundations of this work. In the following, the necessary model and data prerequisites in order to pursue the goals of this thesis are discussed. As a quick reminder: This work aims to reexamine the generalizability of a model trained on data from Great Britain and Northern Germany to other more heterogeneous European regions. Additionally, the potential of an active learning approach shall be studied to enable a more efficient model transfer to other regions. Questions that are related to these topics, like finding a sufficient number of deep ensemble members for high-quality uncertainty estimates and identifying geographical properties that deteriorate the model performance, are also investigated.

First, a suitable model architecture that serves as the basis for the deep ensembles needs to be selected. Therefore, two different model architectures – the *U-Net* and *LinkNet* architectures – are described and tested with a variety of so-called *backbone* architectures. All technical terms, architectures, and implementational details are explained in their respective section. The different entities are compared in several relevant quantities. Next, the revision of the data generated in the work of (Dasenbrock et al., 2021), which contains satellite images of pipeline pathways in Great Britain and Northern Germany, is presented. This is followed by the extension of the dataset with satellite imagery from Spain in order to set up the experiments on an active learning approach.

5.1 Choosing a Model Architecture and Encoder-Backbone

The chapter at hand focuses on the evaluation of different model architectures for deep learning ensembles. Each candidate model consists of a base architecture and a backbone. The base architecture has to account for a variety of challenges related to the task of detecting pipeline pathways. These include small training datasets, high input noise because of low-resolution satellite imagery, and intensive computational and memory requirements due to the usage of deep ensembles. The following two base architectures, which have low training sample requirements – also on noisy input data – and feature manageable hardware requirements, have been selected:

- **U-Net**

The U-Net architecture is a well-proven CNN architecture originally engineered for the segmentation of medical imagery (Ronneberger, Fischer, & Brox, 2015). It has already been used in the master thesis of Jan Dasenbrock, which was the basis for the paper (Dasenbrock et al., 2021). Other examples that are related to the topic of the thesis at hand, are the works of Matthias Zech and Joseph Ranalli on the detection of photovoltaic areas in satellite data (Zech & Ranalli, 2020) and a paper from (Hartmann et al., 2021) with the title *Bayesian U-Net for Segmenting Glaciers in SAR Imagery*. All use cases featured small, sometimes noisy training datasets. This makes U-Net a valid candidate solution for the base architecture.

- **LinkNet**

Tests on different neural network architectures in my internship preceding this master thesis have shown that LinkNet architectures provide segmentation accuracies comparable to U-Net architectures while being slightly faster in training and evaluation time. This will also be shown in 5.1.4. Considering that the use of deep learning ensembles is intensive in memory consumption and computations, this property makes LinkNet a possible candidate for the use as base architecture.

Both architectures – U-Net and LinkNet – can be combined with different *backbone architectures* to improve the model performance. A more detailed explanation of both base architectures and the backbone architectures will be the topic of the following sections 5.1.1, 5.1.2, and 5.1.3. This chapter will close with the comparison of all possible architecture-backbone combinations.

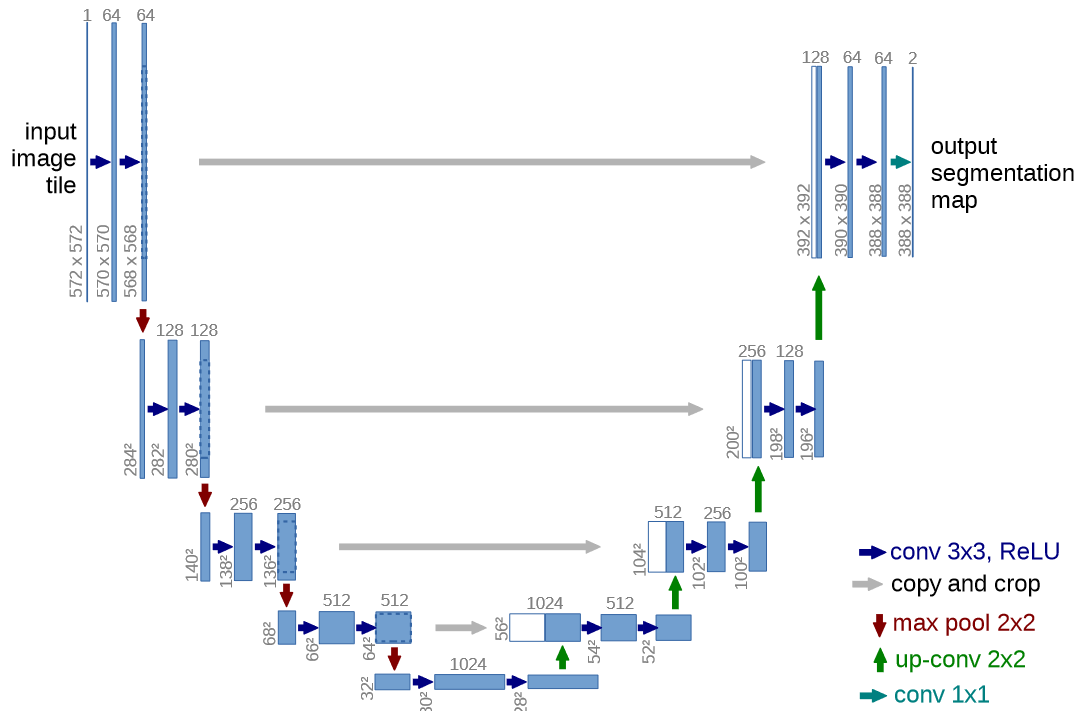
5.1.1 U-Net

The U-Net architecture has its origins in the field of biomedical image segmentation and was proposed by Olaf Ronneberger, Philipp Fischer, and Thomas Brox from the University of Freiburg in 2015 (Ronneberger et al., 2015). It belongs in the category of encoder-decoder-based models and has become a widely spread architecture also outside of medical and biomedical image segmentation (Minaee et al., 2020). The aim of the 2015 published paper *U-Net: Convolutional Networks for Biomedical Image Segmentation* was to create an architecture that can be trained very sample efficient by means of strong data augmentation. The model has won two challenges at the International Symposium on Biomedical Imaging 2015 (Olaf Ronneberger, 2015).

U-Net is a fully convolutional neural network which means that the network does not rely on any fully connected layers. It consists of two paths: The contracting path – also-called *encoder* – on the left side of the network compresses the input image via a series of convolutions and max-pooling operations to transform the high-resolution euclidean space into a high dimensional feature space. The opposite path – called *decoder* or *expansive path* – transforms the high dimensional feature space back to a high-resolution euclidean space to extract an output segmentation map. This back-transformation is done by upsampling the feature maps from the previous layer. The received upsampled feature maps are then concatenated with resolution-wise matching feature maps of the encoder path to localize the upsampled feature maps. Subsequently, a convolution layer will halve the number of feature maps and be trained to create a more precise output based on these concatenated inputs (Olaf Ronneberger, 2015). This architecture consisting of the encoder and decoder can be depicted as a roughly symmetrical network forming a U-shape, as shown in Figure 5.1.

5.1.2 LinkNet

The LinkNet architecture was proposed by Abhishek Chaurasia and Eugenio Culurciello (2017) in their publication *LinkNet: Exploiting Encoder Representations for Efficient Semantic Segmentation*. Their goal was to increase the parameter efficiency of neural networks, meaning they wanted to lower the computational complexity and increase the training and evaluation speed while maintaining state-of-the-art accuracies on typical benchmark datasets. Like U-Net, LinkNet is tailored toward use cases in the field of image segmentation and uses an encoder-decoder architecture


 Figure 5.1 **U-Net architecture**

A 572x572 greyscale image is – by means of several 3x3 convolutions and max-poolings – encoded into 1024 28x28 feature maps. Subsequently, the decoder path maps these feature maps – concatenated with the matching cropped feature maps of the encoder path – through 2x2 up-convolutions and 3x3 convolutions to the initial spatial resolution. Due to the multiple convolutional operations, there is a loss of information at the border of the image. The output has the dimensions of 2x388x388 because the architecture at hand segments into two categories (Ronneberger et al., 2015).

(Chaurasia & Culurciello, 2017).

The novelty of LinkNet lies in the way the neural network connects the encoder to the decoder path. Instead of e.g., saving the max-pooling indices in the down-sampling process or concatenating the output of encoder blocks with the input of a decoder block (like in U-Net), LinkNet bypasses the output of an encoder block directly to the corresponding decoder block (see Figure 5.2). The information is then added onto the output of the previous decoder block. This way, the decoder can benefit from the learned representations in the encoder without a need for additional trainable parameters. This makes the operation computationally efficient while maintaining spatial information (Chaurasia & Culurciello, 2017).

Different benchmark tests in the original paper from 2017 have shown that the proposed concept works as intended. It has been shown that LinkNet reaches equal

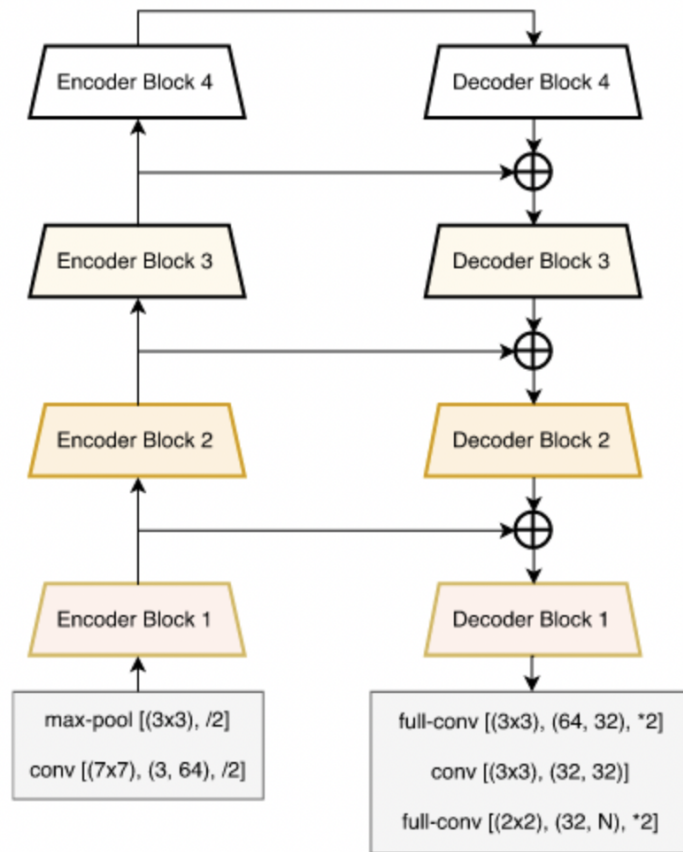


Figure 5.2 **LinkNet architecture**

Preceding the encoder, the input image is downsampled twice by a factor of 2 (denoted by $/2$) through a 7×7 convolution layer and a 3×3 max-pooling operation. This is followed up by four encoder blocks from which the output of the first three encoder blocks is passed directly to the decoder and added to the output of the corresponding preceding decoder block. After the decoder, the outputs are getting upsampled twice by a factor of 2 (denoted by $*2$) (Chaurasia & Culurciello, 2017).

or even higher benchmark scores compared to baseline models with a 10-fold higher number of parameters. These properties make the architecture a promising candidate for this master thesis with respect to the approach of deep learning ensembles.

5.1.3 Backbone Architectures

Encoder-decoder-based neural networks for semantic image segmentation can be combined with different so-called *backbone architectures* on the encoder path. Compared to implementing an encoder path from scratch, the use of a backbone has the advantage that we can use a proven pre-trained network architecture, like VGG or ResNet models, as a feature-extractor to increase accuracy and reduce training time. The chosen backbone architecture in the encoder path has just to be matched with a fitting decoder path to re-transform the input image. There are different python libraries that fulfill this purpose. One of them is the *Segmentation Models* library, which is available as a TENSORFLOW implementation (Yakubovskiy, 2019) and a PYTORCH implementation (Yakubovskiy, 2020). Due to the steady increase in popularity since its release in 2016, which is depicted in Figure 5.3, and a more extensive selection of backbone architectures in the Segmentation Models library, the PYTORCH implementation is used in this work.



Figure 5.3 **Google Trends comparison for search entrys *tensorflow* (red) and *pytorch* (blue)**

The graph shows the relative search interest in a weekly resolution from 17.07.2016 – 12.06.2021. The highest value of 100 is the peak in search frequency for either of the compared terms. All other frequencies are depicted relative to this value, meaning that a value of 50 represents half of the search requests compared to the peak in a given week (Trends, 2021). The interest in PYTORCH is following a steadily increasing trend while it is apparently gaining market-share.

Out of the variety of available backbone architectures in the Segmentation Models library, seven are picked for later comparisons. An exhaustive list of all backbone architectures can be found on Segmentation Models' GitHub repository in (Yakubovskiy, 2020). Each backbone architecture is available in different versions, meaning they can differ in network depth and width but make use of the same core elements and same principles. In two of five cases, only the backbone version with the lowest amount of parameters was chosen because of the benefits in regard to training time and memory usage. Following the deep learning ensembles approach,

this will minimize the overall time used for training and evaluation. The following list gives an overview of the picked backbones and a brief explanation of the key properties and motivations issued by the authors of the papers where each architecture was originally introduced.

- **ResNet-18 and ResNet-34**

In the 2015 published paper *Deep Residual Learning for Image Recognition*, the authors tackled the problem that by adding more layers to a deep neural network its accuracy eventually starts to saturate until it drops rapidly. This effect is called *degradation* and is not linked to overfitting (He, Zhang, Ren, & Sun, 2015) but to the vanishing gradient problem (Kramer, 2020), which was already discussed in subsection 3.2.1. Intuitively, added layers to a network should not decrease the accuracy of an output but rather learn an identity mapping if the added layers do not add any benefits. ResNet modules – which are most often a stack of layers – add their input back to the output, which is called an identity shortcut (see Figure 5.4).

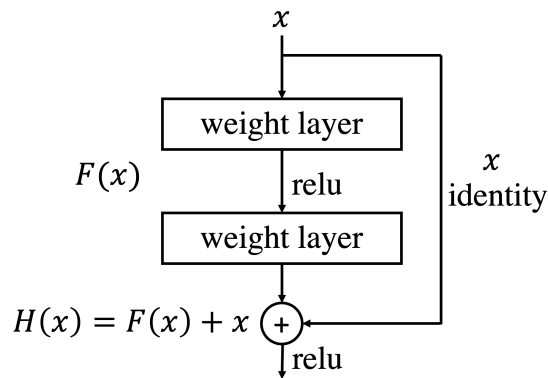


Figure 5.4 **ResNet building block**

The identity x , which is the input to two subsequent weight layers, is also bypassed to be later added to the output of said weight layers. Inspired by (He et al., 2015)

This makes the problem of learning the underlying mapping $H(x)$ a problem of learning the residual mapping $F(x) := H(x) - x$, which means that the original function becomes $H(x) := F(x) + x$. The authors have shown that this is an effective strategy counteracting the problem of degradation. It allows for deeper architectures with increasing accuracy scores compared to their plain counterparts without identity shortcuts (He et al., 2015). The number attached to ResNet (e.g., 18 in 'ResNet-18') refers to the number of layers in the model architecture.

- **MobileNetV2**

The motivation behind MobileNet was to create a small and fast model architecture that is tailored towards the usage in mobile and embedded systems for visual tasks. This was achieved by using depthwise separable convolutions, meaning that a conventional convolution is factorized into two operations: A depthwise convolution and a 1×1 pointwise convolution. This leads to a massive decrease in parameters and number of computations while causing – keeping the intended use case in mind – a reasonable loss in accuracy (Howard et al., 2017). MobileNetV2 added further features resulting in a better performance of the MobileNet neural network family (Sandler, Howard, Zhu, Zhmoginov, & Chen, 2019).

- **VGG11 and VGG16**

The main focus of the 2015 published paper *Very Deep Convolutional Networks for Large-Scale Image Recognition* was to analyze the dependence between the depth and the accuracy of deep neural networks. The authors used up to 19 layers of 3×3 convolutions and have shown that an increase in network depth is beneficial for the network performance in terms of prediction accuracy. The number attached to VGG (e.g., 16 in 'VGG16') represents the number of layers used in the specific model. In 2014 it was one of the winners at the ImageNet Large Scale Visual Recognition Challenge (Simonyan & Zisserman, 2015) and is still widely available in standard deep learning libraries today.

- **DenseNet121**

The DenseNet architecture can be seen as an extension of the idea behind ResNet. According to the authors, "DenseNets have several compelling advantages: they alleviate the vanishing-gradient problem, strengthen feature propagation, encourage feature reuse, and substantially reduce the number of parameters" (Huang, Liu, van der Maaten, & Weinberger, 2018). Instead of providing identity shortcuts just from the input of a block to its output (like in ResNet), DenseNets provide the output of each layer to all its subsequent layers inside of a dense block – so-called *dense connections*. Furthermore, the bypassed output is not simply added to the input of another layer but concatenated. DenseNets have shown state-of-the-art results while needing a substantially lower parameter-count and being less computationally taxing.

- **EfficientNet-B0**

The purpose of the paper *EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks* was to study how models can be scaled in depth, width,

and resolution to obtain an optimized model performance at a given amount of resource budget. The authors developed a whole family of networks called EfficientNets, which reached state-of-the-art performances. EfficientNet-B0 was developed as a baseline network which was then scaled in depth, width, and resolution to create its descendants EfficientNet-B1, EfficientNet-B2, \dots , EfficientNet-B7 (Tan & Le, 2020).

5.1.4 Comparisons

In the following, the different architecture-backbone combinations are tested. For the approach of deep learning ensembles, it is important to find a solution that provides a good balance between prediction accuracy, the number of parameters needed, and the training time. A focus on just one of these aspects would make the model most likely unfeasible for later deployment.

In order to make the comparisons as meaningful as possible while keeping the evaluation and training times at a reasonable level, all architecture-backbone combinations have been trained the same way. Additionally, each test run inherited 50 epochs to account for the variance in runtime between single epochs and was only performed once, meaning that no further statistics like a mean runtime or variance have been captured. Hence, the following numbers have to be regarded only as indicators for the model performances. In regard to finding a fitting architecture-backbone combination this should serve the purpose well.

Adam was used as the optimizer with the learning rate set to 1E-3 and a batch size of 1. The loss function was given by

$$\text{loss}(x, \text{class}) = -x[\text{class}] + \log\left(\sum_j \exp(x[j])\right) \quad , \quad (5.1)$$

the standard cross entropy loss provided by PYTORCH (Torch Contributors, 2019). The dataset included all samples created in the work of (Dasenbrock et al., 2021) augmented with horizontal and vertical flipping applied with a probability of $p = 0.5$. This resulted in 430 training samples and 108 validation samples. Each sample has a dimension of $64 \times 64 \times 6$ (width \times height \times channels), showing that all available bands with a resolution of 30^2 m px^{-1} from Landsat 5 have been used. All test runs were like already mentioned run for 50 epochs. To evaluate the model performances, the *Intersection over Union* (IoU) score – also referred to as *Jaccard index* – was used, which is given by

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|} \quad (5.2)$$

(A. Zhang et al., 2021). A can, for example, refer to the pixels labeled as *pipeline* in the ground truth, and B represents the pixels classified as *pipeline* in the prediction. The IoU can have values between 0, meaning there is no similarity between the ground truth and prediction, and 1, representing a perfect prediction. Figure 5.5 provides a graphical representation of the Jaccard index.


$$\text{IoU} = \frac{\text{Area of Overlap}}{\text{Area of Union}}$$


Figure 5.5 **Intersection over union score**
As shown in (Rosebrock, 2016).

Table 5.1 sums up all test results. Due to the different design choices and goals of U-Net and LinkNet, it is expected that all backbone-LinkNet combinations should use fewer parameters and shorter training time than their U-Net counterpart. This expectation was met for all architecture-backbone combinations. The shortest training time was reached by the combination LinkNet+ResNet-18 backbone. This combination was also the setup used in the initial paper in 2017 with which the LinkNet architecture scored state-of-the-art results in well-known benchmark datasets (Chaurasia & Culurciello, 2017). With a validation-IoU-score of 0.60, LinkNet+ResNet-18 is the third most accurate model tested and is only surpassed by VGG16 combinations, which needed more than double the amount of training time. The U-Net counterpart for the ResNet-18 backbone reached a slightly worse validation-IoU-score while taking 28.2 % longer to train.

Most architecture-backbone combinations with a smaller amount of trainable parameters showed significantly lower validation-IoU scores making them less viable for a reliable detection of pipelines. The EfficientNet-0 backbone tests resulted in very long training times despite their comparatively low parameter counts. This makes them unfeasible for the deep learning ensembles strategy followed in this work. The same applies to the implementations of DenseNet121 and ResNet-34. The U-Net+MobileNetV2 combination showed the highest training-IoU-score. This does not have a direct influence on the selection of the right architecture-backbone combination because after 50 epochs of training, almost all combinations showed such high numbers. This means the models are strongly overfitting to the training

Table 5.1 **Architecture-backbone comparisons**

Architecture / Backbone	Parameters	Training time (*)	Train-IoU (**)	Val.-IoU
U-Net / ResNet-18	14.3M	44m 22s	0.67	0.59
LinkNet / ResNet-18	11.7M	34m 36s	0.72	0.60
U-Net / ResNet-34	24.4M	80m 0s	0.69	0.57
LinkNet / ResNet-34	21.8M	65m 54s	0.65	0.55
U-Net / MobileNetV2	6.6M	51m 16s	0.93	0.56
LinkNet / MobileNetV2	4.3M	47m 5s	0.68	0.51
U-Net / VGG11	18.3M	63m 9s	0.72	0.59
LinkNet / VGG11	10.5M	44m 18s	0.71	0.60
U-Net / VGG16	23.8M	95m 6s	0.70	0.63
LinkNet / VGG16	16.0M	85m 35s	0.69	0.62
U-Net / DenseNet121	13.6M	115m 55s	0.72	0.56
LinkNet / DenseNet121	10.4M	129m 25s	0.70	0.58
U-net / EfficientNet-B0	6.3M	149m 10s	0.81	0.58
LinkNet / EfficientNet-B0	4.2M	134m 16s	0.69	0.53

* Training time has been determined on a 2020 Apple M1 Chip. 8 GB of RAM, PYTHON 3.9.4, PYTORCH 1.8.0, and SEGMENTATION-MODELS-PYTORCH 0.1.3. Each training run was performed on 50 epochs with 430 training samples, 108 validation samples, Adam optimizer with a learning rate of 1E-3, and a batchsize of 1.

** Training-IoU depicts the Training-IoU reached in the epoch with highest Validation-IoU

data and are becoming worse at generalization, meaning the validation-IoU-scores are decreasing. On the other hand, this is a good indicator for the capability of the tested models to depict the detection of pipeline construction sites on satellite

imagery.

In conclusion, the choice of a fitting architecture-backbone combination is a trade-off between the number of parameters, which taxes the computational memory, the training time, which is a good indicator for the computational complexity, and the model accuracy, making a statement about its usefulness and reliability when deployed. Additionally, a lower amount of parameters reduces the parameter space and consequently decreases the overall model uncertainty, given that the model is able to capture the underlying function. All things considered, the LinkNet+ResNet-18 model shows a fast training time, uses a manageable number of parameters, and provides good accuracy scores. It is therefore selected as the basis for the deep learning ensembles.

5.2 Revision: The Great Britain and Northern Germany Dataset

The *Great Britain and Northern Germany dataset* stems from the proof-of-concept study conducted by (Dasenbrock et al., 2021) and includes different pipeline segments from the gas transmission network of Great Britain (GB) and the NEL pipeline in Northern Germany (NEL). In the following, its denomination is abbreviated as GBNEL. It will serve as the data basis for an initially trained deep ensemble. The respective deep ensemble will be used to test its ability to generalize to other regions in Europe.

To create the original GBNEL dataset, (Dasenbrock et al., 2021) firstly filtered OPENSTREETMAP (OSM) data with the ESY-OSMFILTER (Pluta & Lünsdorf, 2020) for pipelines in GB and Northern Germany. The esy-osmfilter is an open-source Python library that reads and filters OSM data to then output Python dictionaries or JSON files. OSM itself is a project that collects and provides free geographic data of the world. This includes a variety of objects like streets, hiking and bicycle trails but also the energy and gas infrastructure (OpenStreetMap, 2020). The filtered OSM data was then loaded into GOOGLE EARTH ENGINE (GEE) (GoogleEarthEngine, 2021), which is an extensive catalog for satellite imagery and geospatial data with analytical capabilities on a planetary scale. Furthermore, tools, like the GEE code editor that allows interaction with the catalog, are provided. The authors used the GEE code editor to load Landsat 5 Tier 1 Surface Reflectance imagery that covers the area of interest given by the OSM pipeline data. Because satellite data often spatially exceeds the area of interest, it had to be cropped accordingly. The cropped data was then downloaded and afterward sliced into $64 \text{ px} \times 64 \text{ px}$ tiles. The ground truth masks were created by projecting the pipeline course onto all-black (value: 0) $64 \text{ px} \times 64 \text{ px}$ TIFF files that are matched with the corresponding $64 \text{ px} \times 64 \text{ px}$ Landsat 5 image tiles. Pixels on the mask files that lie within the range of 30 m of the pipeline course were set to white (value: 1). This provides a classification of *background* and *pipeline*. Figure 5.6 depicts this process. Only image-mask pairs that contain more than 50 pixels of pipeline were kept and considered for the training of the model.

Initially, the data was divided into the GB and the NEL data set, and a model was trained exclusively on the GB data and validated on the German NEL data. This approach was chosen to test if the model is able to generalize to other regions and was confirmed by high validation scores on the NEL pipeline imagery (Dasenbrock et al., 2021).

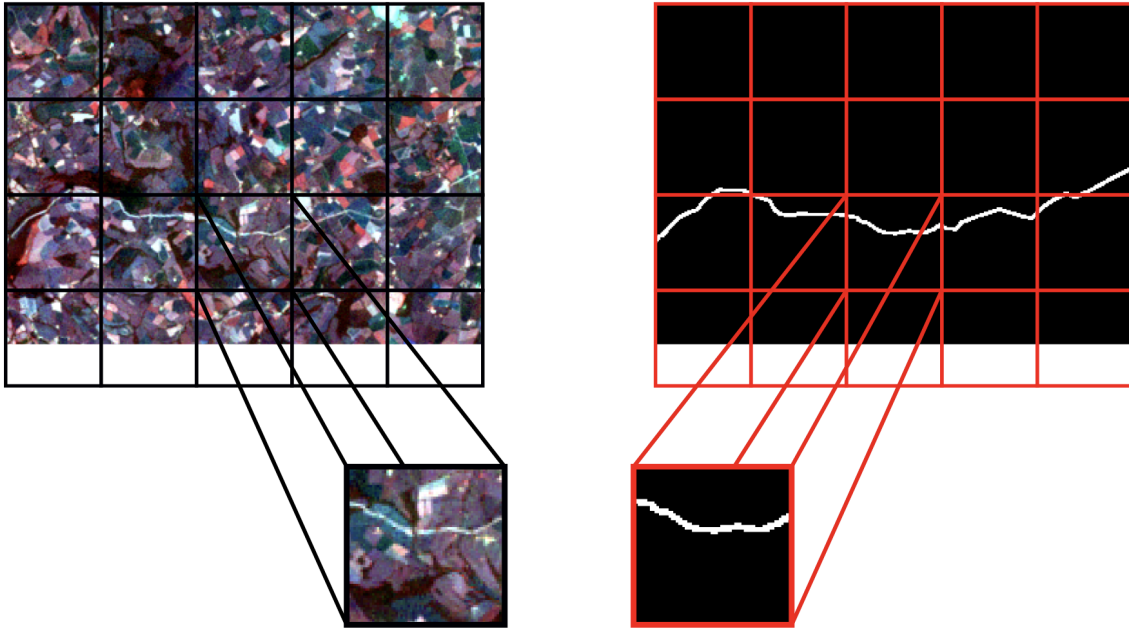


Figure 5.6 **Creation of image-mask pairs**

Left side shows the downloaded Landsat 5 Tier 1 Surface Reflectance imagery divided into $64 \text{ px} \times 64 \text{ px}$ tiles, and the right side shows the corresponding masks (Dasenbrock et al., 2021).

The thesis at hand takes advantage of the preprocessed satellite imagery from the work of (Dasenbrock et al., 2021) but does not make use of the created masks nor differs between GB and NEL data. The matching of satellite imagery with OSM data to create the ground truth masks requires very high accuracy of the OSM data. This is not always the case for pipeline data, like seen in Figure 5.7, resulting in data noise that possibly deteriorates the model performance. Additionally, the process of labeling all pixels that are within a range of 30 m to the OSM pipelines as pipeline can result in thicker pathways than necessary. A strategy that counteracts this problem is a manual data labeling process. This means that for each image, the mask is created by a human using an input device like a computer mouse to label each pixel individually. This work makes use of the online data labeling platform APEER provided by Carl Zeiss AG (Carl Zeiss Microscopy GmbH, 2021). All $64 \text{ px} \times 64 \text{ px}$ image tiles have been relabeled in a manual fashion.

In total, the revised GBNEL dataset consists of 357 samples, of which 34 are from the NEL pipeline and 323 from the GB gas transmission network. Each sample has the dimensions $64 \times 64 \times 6$ (*width* \times *height* \times *c*), where *c* is the number of channels. The channels include the Landsat 5 bands TM1, TM2, TM3, TM4, TM5, and TM7. The thermal TM6 band with a resolution of 120^2 m px^{-1} is neglected to ensure data consistency. In order to use the dataset to train deep ensembles, it was split into

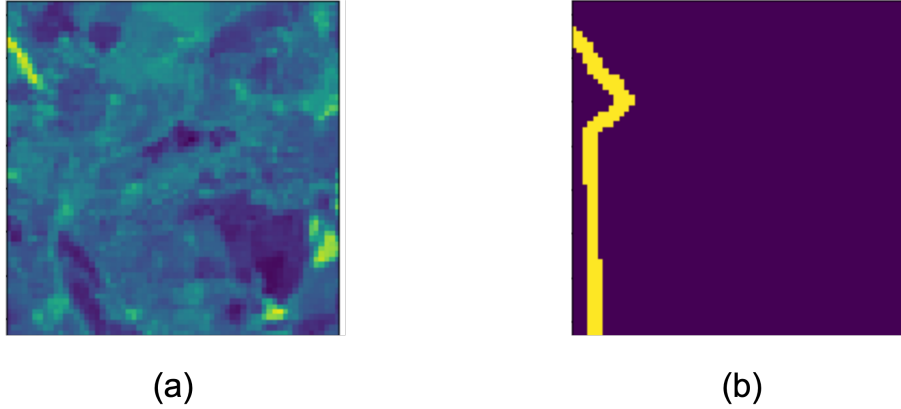


Figure 5.7 **Automatically created image-mask pair**

An image-mask pair created by applying OSM data (b) to a satellite image (a). In this case, the mismatch arises most likely out of the fact that the right-of-way (bright yellow/green) has not been created at the time the satellite image has been taken.

a training and a test dataset with a ratio of 80 to 20. The training data was then augmented with horizontal and verticals flips with a probability of $p = .5$ to increase the number of samples available for training. This resulted in 409 training images and 73 test images. The final training and test datasets were saved to be later reloaded, ensuring a consistent database.

5.3 Dataset extension: The Spanish Dataset

To test the active learning and the generalization capabilities of the ensemble, respectively, its members, a second dataset was created: The Spain dataset (ES). It can serve as a stand-alone dataset or extend the GBNEL data. Spain was the region of choice because of its substantial difference in geographical and vegetational features compared to the GBNEL data, like seen in Figure 5.8.



Figure 5.8 **Overview map of Europe**
Picture downloaded from (geojson.io, 2021).

Additionally, the OSM data availability proves to be very reliable (Figure 5.9), and the Spanish gas transmission system operator *Enagás* provides helpful pipeline data, like the year of construction and the connected locations (Enagás, 2021). This benefits the process of creating new training and validation data greatly.

The process of generating individual samples of satellite imagery for Spain follows a different approach in comparison to the data generation process of the UKNEL samples. Using GEE, a location consisting of a latitude and longitude is generated on the map. The generation of a location can, for example, be done with a mouse click or in other ways, like the provision of multiple locations in a table that is loaded into GEE. A JavaScript script loads Landsat 5 Tier 1 Surface Reflectance images – a so-called *collection* – at the location in a given space of time. Ideally, the time



Figure 5.9 **Available OSM pipeline data in Spain**

Pipelines are shown in black. Data was extracted with *esy-osmfilter* (Pluta & Lünsdorf, 2020) and visualized with (geojson.io, 2021).

window is chosen close after the construction date of the pipeline to detect. The downloaded collection is then sorted by the parameter *CLOUD_COVER*, and only the image with the lowest amount of clouds is considered for the subsequent steps (Figure 5.10a). A rectangle is then drawn around the chosen location that covers a region of roughly 50 px in each direction, depicting an area of approximately 9 km² (Figure 5.10b). The satellite data covered by the rectangle is then clipped and downloaded (Figure 5.10c). In a post-processing step, the downloaded image data is trimmed to the final dimensions of $64 \times 64 \times 6$ (width \times height $\times c$) using a simple python script (Figure 5.10d).

This method can be used to automate the download of relevant satellite imagery of very specific sites from which the detection of pipeline pathways can be started. Model runtimes, download times, and memory requirements are minimized. The SciGRID_{gas} data set provides a variety of sites that would serve as sensible starting points – like geolocated compressor stations – for an automated pipeline detection tool.

The labeling of the data was done manually with APEER. In total, 247 samples from 9 pipeline segments have been generated. The samples are randomly distributed over the pipeline segments shown in Figure 5.11. All samples contain the Landsat 5 TM1, TM2, TM3, TM4, TM5 and TM7, bands. The samples were then randomly split

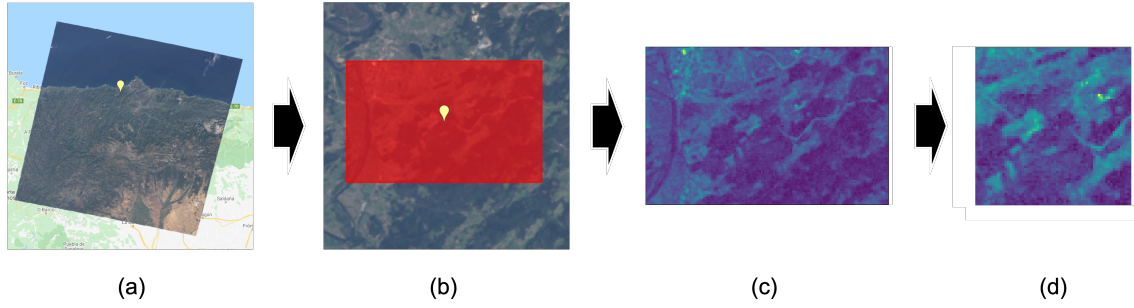


Figure 5.10 **Landsat 5 imagery download and preparation**

(a) A single Landsat 5 image from Northern Spain with the lowest *CLOUD_COVER* available between 01.03.1998 and 31.10.1998. The chosen location is indicated in yellow. (b) A rectangle is drawn around the location. The width and length of the rectangle depend on the longitude and latitude of the location of the rectangle. Because of the way the shape of a rectangle is estimated in the script, rectangles closer towards the equator become wider, while rectangles closer to the poles become longer. (c) The downloaded image ($156 \times 101 \times 6$ channels) clipped from the rectangle created in (b). (d) Final image trimmed to final dimensions of ($64 \times 64 \times 6$ channels).

into a training and test dataset with a ratio of 80 to 20. Additionally, random data augmentations – in particular horizontal and vertical flips of the input images – were applied to the training data with a probability of $p = 0.5$. This resulted in the final ES dataset consisting of 291 training samples and 43 test samples. The dataset is saved to be available for later experiments. This provides a consistent data basis.



Figure 5.11 **Pipeline segments in final dataset**

The pipeline segments used as the source for the training and test dataset are indicated in red. Map created with (geojson.io, 2021) and pipeline data extracted from (OpenStreetMap, 2020) with esy-osmfilter (Pluta & Lünsdorf, 2020).

6 Hypotheses, Evaluation and Results

After obtaining the prerequisites – a sensible deep ensemble model-backbone architecture and the respective datasets – the three hypotheses of this work, their evaluation methods, and the results are presented. The approach on how each hypothesis is evaluated differs greatly. Additionally, the works and results on a hypothesis always impact its subsequent hypothesis: The deep ensemble trained on the GBNEL data used to test the model generalizability to other regions serves as the basis for determining a sufficient amount of ensemble members for high-quality uncertainty estimates in the next hypothesis. The results will then be used to potentially decrease the number of ensemble members needed for the active learning experiments. The outputs of a resulting deep ensemble will be analyzed in regard to influencing factors that systematically deteriorate model accuracies. Hence, this chapter is structured along the hypotheses. Each section consists of a derived hypothesis, the methods and experimental setup to evaluate it, and the concluding results. Hypothesis-specific statistical tests and metrics are introduced in their respective section for better comprehensibility.

6.1 Hypothesis 1: Model Generalizability

The work of (Dasenbrock et al., 2021) has shown that a model trained on data from Great Britain is able to generalize to other regions like Northern Germany (NEL pipeline) with high evaluation IoU scores. This also suggests that the data from GB and NEL is very similar regarding its properties and originates from closely related basic distributions. Looking at Figure 6.1, it becomes apparent that both regions located within the two white dashed lines have rather similar vegetation and, in some areas, a comparable topography.

The properties of Spain between the dashed orange lines are substantially different and likely make up a different basic distribution in regard to the task of predicting pipeline pathways. For example, the contrast between the right-of-way and its sur-

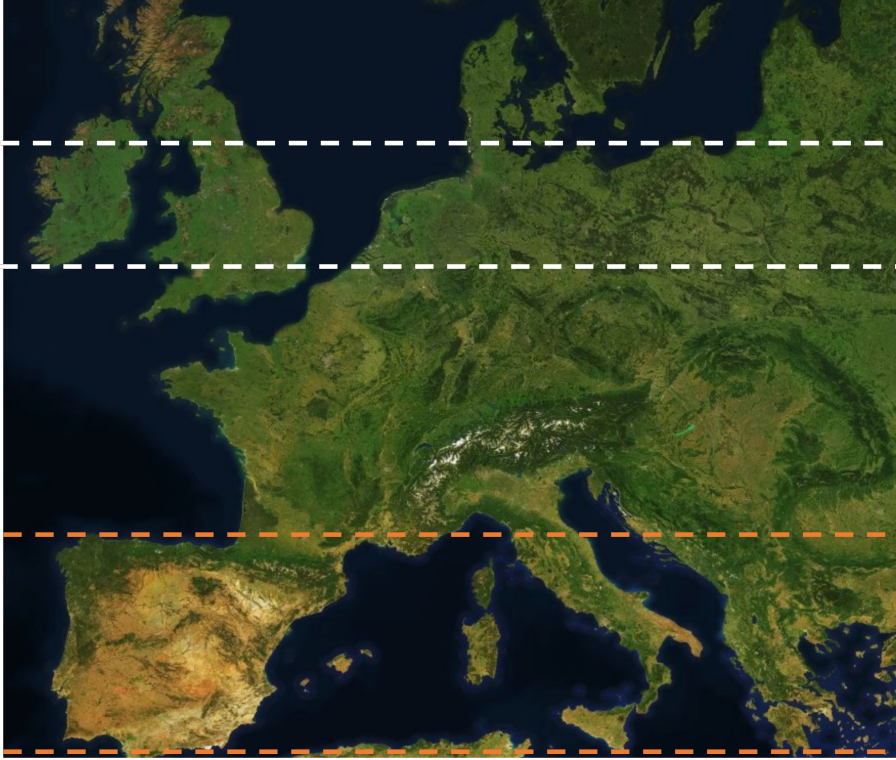


Figure 6.1 **Exemplary comparison of different regions**

The white dashed lines indicate most of the area in which the GBNEL data is located. The orange dashed lines encapsulate Spain. The two Regions substantially differ in their properties regarding the vegetation and topography. Picture extracted from (geojson.io, 2021).

roundings is higher in areas with high vegetation (GB and Northern Germany) in comparison to sandy soils with low vegetation (Spain). This is depicted in Figure 6.2. Additionally, other image features that can potentially affect model predictions, like the presence of roads and mountain ridges, can differ in their frequencies and properties.

It is therefore hypothesized that a model respectively deep ensemble trained on GBNEL data is not able to generalize well to other, more heterogeneous regions like Spain. This should show in low IoU scores and high model uncertainties on ES test images from Spanish right-of-ways.

6.1.1 Model Generalizability: Training of a GBNEL Deep Ensemble

To evaluate the model generalizability to the ES data in terms of IoU scores and model uncertainties, a deep ensemble is trained on the GBNEL data.

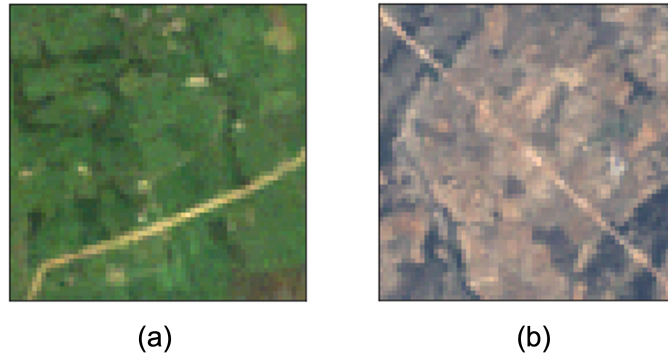


Figure 6.2 **Right-of-way visibility in different regions**

(a) A right-of-way in UK going from the bottom-left to the right. (b) A right-of-way in Spain going from the top-left to the bottom-right.

As elaborated in subsection 5.1.4, the tests on different architectures and backbones suggest that the combination of a LinkNet architecture with a ResNet-18 backbone is a sensible choice for the deep ensemble. Like in subsection 5.1.4, the GBNEL models were implemented and trained with PYTORCH and the Segmentation Models library. The previously created 409 training samples and 73 test samples from the GBNEL dataset were loaded.

In total, 32 models were trained, from which only 16 were considered for the ensemble. The comparisons between deep ensembles and Bayesian dropout in (Gustafsson et al., 2020) suggest that the added benefits in terms of uncertainty quantification and IoU scores stagnate with more than 16 ensemble members. The metric used to discriminate between models chosen for the final ensemble is the IoU score reached on the validation data. All 32 models were initialized with random weights but shared the same hyperparameters:

- Optimizer: Adam
- Learning rate: 1E-3
- Batch size: 32
- Loss function: Cross entropy loss

The training dataset was shuffled after each epoch to satisfy the requirements for training a deep ensemble, as stated in (Lakshminarayanan et al., 2016). The training for each model was stopped after 50 epochs or earlier if the validation loss did not decrease anymore for ten consecutive epochs. No model has reached the maximum possible number of epochs. The model and optimizer states – which represent the parameter realizations at a given training step – with the best validation loss were

then saved to be available for future usage. The training and validation IoU scores for the 16 best-performing models are shown in Table 6.1.

Table 6.1 **GBNEL ensemble member performances**

Model	Training-IoU (*)	Validation-IoU
1	0.77	0.57
2	0.73	0.59
3	0.71	0.59
4	0.77	0.59
5	0.69	0.58
6	0.72	0.60
7	0.70	0.57
8	0.72	0.57
9	0.70	0.58
10	0.65	0.57
11	0.72	0.56
12	0.68	0.59
13	0.80	0.57
14	0.77	0.57
15	0.76	0.60
16	0.70	0.58
* Training data IoU at training step with lowest validation loss		

The best validation IoU for a single ensemble member is 0.60, and the average model validation IoU is 0.58. The average model validation IoU is not to be mistaken for the ensemble prediction that averages the predictions of all ensemble members for each sample and classifies based on this mean prediction. The validation IoU of the whole deep ensemble is 0.63, which confirms that ensembles can outperform their individual members, as stated in (Lakshminarayanan et al., 2016).

6.1.2 Model Generalizability: Results

To evaluate if the deep ensemble trained on the UKNEL data is able to generalize to Spanish regions, the deep ensemble is applied to the ES test data. This resulted in an IoU score of 0.12 (Figure 6.3). Because the ES training data has also not been presented to the ensemble, it can also provide valid assumptions about the generalizability of the GBNEL trained ensemble. The validation IoU on the ES training data is also 0.12. These scores show that the deep ensemble is not able to reliably detect pipeline pathways in Spain. It is therefore not able to generalize to other, more heterogeneous regions in a satisfying manner.

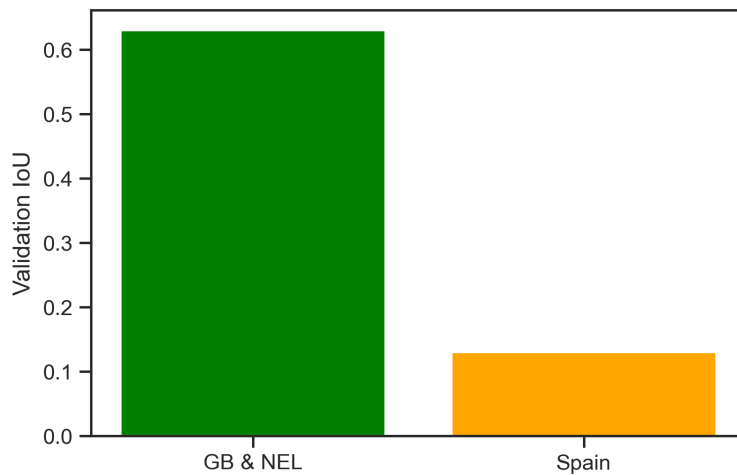


Figure 6.3 **Validation IoU for GBNEL and Spanish test datasets**

Validation IoUs were calculated using the complete deep ensemble with its 16 members trained on the GBNEL data.

Further evaluations on the estimated uncertainties, represented by the variance between the ensemble members, show an average uncertainty for GBNEL samples of 8.79 and an average uncertainty for ES samples of 20.22. The average uncertainty was calculated by summing up all pixel variances for all samples of each region and dividing them by the number of samples of each region. Figure 6.4 also shows that the spread of sample variances in Spain is a lot higher than for GBNEL data. This is expected after observing low validation IoUs because epistemic uncertainty estimates provide an approach to discriminate between in and out of distribution samples. Naturally, an ensemble should perform worse on out-of-distribution samples because they stem from a different basic distribution that was not part of the training data. As described in chapter 4, the epistemic uncertainty can be decreased by presenting more samples to a model. The deep ensemble was only trained with a variety of GBNEL data samples, which resulted in lower uncertainty estimates for the respective validation dataset compared to the Spanish data. The high spread of sample variances in Figure 6.4 for the Spanish validation data also suggests that some image samples are very closely related to the GBNEL data while others are significantly dissimilar. Both datasets, therefore, originate from different basis distributions.

To get a better understanding of the differences between satellite imagery of pipeline pathways in GBNEL and Spain, Figure 6.5 and Figure 6.6 are provided.

Figure 6.5a depicts a typical well-segmented satellite image. The prediction (a_3) is almost identical to the ground truth shown in (a_2). Aside from some pixels

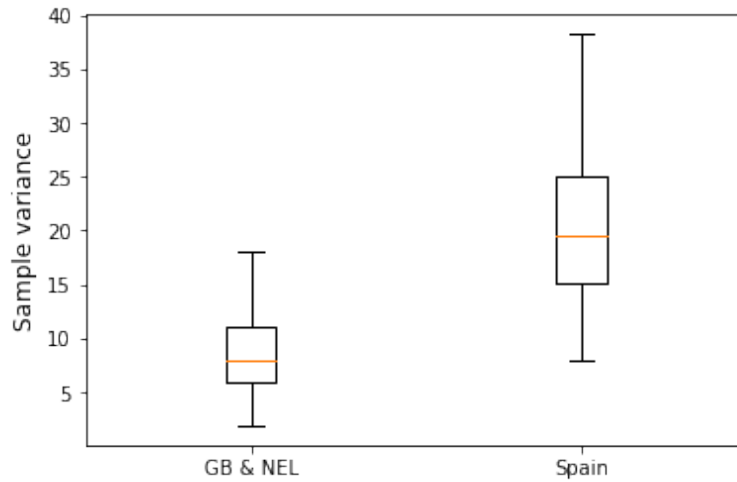


Figure 6.4 **Boxplots of GBNEL and ES sample variance**

The sample variance is given by the summation of all pixel variances for an input image derived from the complete deep ensemble with its 16 members.

at the top of the image, the deep ensemble only expresses relevant uncertainty at the border pixels of the pipeline pathway. This is owed to the circumstance that pixel values only represent the average values of their underlying ground truth. At low-resolutions like 30^2 m px^{-1} , the actual pipeline pathways can not be depicted with sufficient sharpness of separation. This specific uncertainty source is mostly aleatoric because it depends on the noise of the measuring method, respectively, its subsequent data processing. Additional training data will not decrease this source of uncertainty.

Figure 6.5b is a similar example but with an increase in uncertainty estimates close to a field with apparently low vegetation. Figure 6.5c, d, and e exemplarily depict the main source of uncertainty in the GBNEL data. Structures that have a certain similarity to pipeline construction sides, like dirt roads, field borders, and river banks (c_1), deteriorate the model performance and show high uncertainty estimates.

Figure 6.5f is an example where a low image contrast due to cloud cover hinders the ensemble from detecting any pipeline pixels. The high uncertainty estimates (f_4) show that the deep ensemble is in principle able to recognize that the area of the pipeline pathway is of special interest.

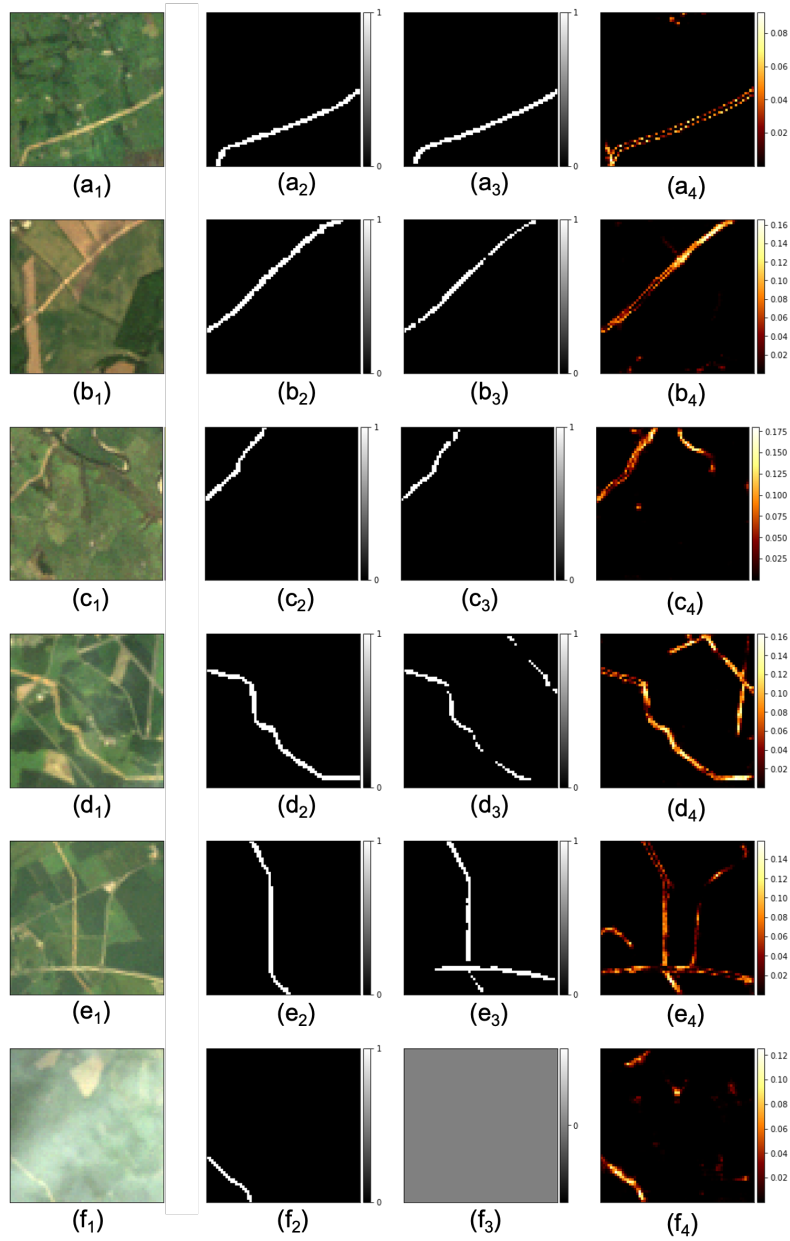


Figure 6.5 **Selected model outputs for GBNEL pipeline pathways**

Left column shows the inputs in true color. Second and third column depict the ground truth, respectively, the ensemble prediction ($M=16$) with black (0) referring to the class *background* and white (1) being *pipeline*. Last column shows the model uncertainty approximated by the ensemble variance.

As seen in Figure 6.6a₃–f₃, it becomes apparent that the ensemble performance is deteriorated for Spanish imagery in comparison to the GBNEL data. The model mainly fails to capture pipeline pathways instead of outputting false positive predictions. To the human eye, Spanish images provide less contrast between the right-of-ways and the background, which can have an impact on the model performance itself. It is also observed that the background serves as a source of high model uncertainty estimates – especially on sandy soils or rocky grounds – like seen in Figure 6.6a₄ and Figure 6.6e₄. The GBNEL model outputs had minimal high background uncertainty estimates except for pipeline pathway-like structures. This fortifies the statement that the two datasets originate from different basic distributions.

Concluding this section, the ensemble trained on GBNEL data is not able to satisfyingly generalize to Spanish satellite imagery. Nevertheless, the estimated model uncertainties give confidence that with additional training data from Spain, the model performance will increase drastically. Figure 6.6b₄–f₄ show high ensemble member variances at the actual pipeline pathways. This epistemic uncertainty will most likely be reduced with an increase in available information, respectively, Spanish training data, which will benefit the model performance.

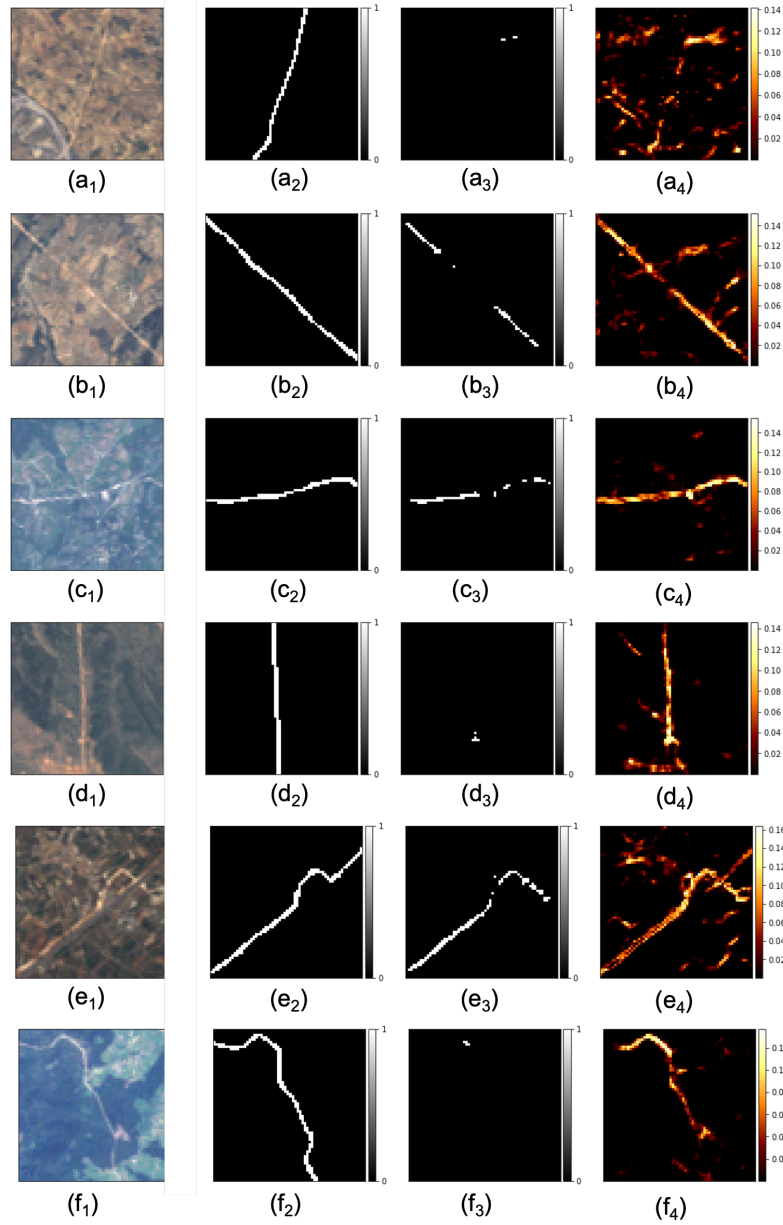


Figure 6.6 **Selected model outputs for Spanish pipeline pathways**

The left column shows the inputs in true color. Second and third column depict the ground truth and the ensemble prediction ($M=16$) with black (0) referring to the class *background* and white (1) being *pipeline*. The last column shows the model uncertainty approximated by the ensemble variance.

6.2 Hypothesis 2: Efficiently Estimating Model Uncertainties

As mentioned before, in their comparisons between deep ensembles and Monte Carlo dropout, (Gustafsson et al., 2020) showed that 16 deep ensemble members proved to be a good number of models to estimate model uncertainties. With more than 16 members, the quality gains in uncertainty estimates stagnated. Of course, the results are highly dependent on the data the uncertainty is estimated on. In this case, the Cityscapes dataset (Cordts et al., 2016), in which urban street scenes, mostly from Germany, are segmented and annotated, was used (Figure 6.7). The authors trained an ensemble to label 19 different classes (road, sidewalk, car, etc.). The Cityscapes scenes presented to the ensemble and the task of labeling 19 classes is assumedly more complex than the segmentation of pipeline pathways. It is therefore hypothesized that a smaller number of ensemble members is sufficient to obtain reasonable uncertainty estimates. A confirmation of this hypothesis and the associated possible decrease in ensemble members would benefit the future training and evaluation efforts, computation times, and memory requirements.



Figure 6.7 **Cityscape street scenes**

A typical street scene segmented into *road*, *sidewalk*, *pedestrian*, etc. Image from (Cityscapes Dataset, 2021)

6.2.1 Efficiently Estimating Model Uncertainties: The Wasserstein Distance and the KS-Test

The test on whether it is possible to use a lower number of ensemble members (<16) while maintaining high-quality uncertainty estimates is carried out in a two-step approach. First, the sample uncertainty distributions for each number of possible GB-NEL ensemble members (2-15) are compared to the sample uncertainty distribution obtained with all 16 ensemble members. The *Wasserstein distance* is introduced and

used for this. A sensible solution would be one where the addition of more ensemble members would not yield a significant increase in similarity towards the uncertainty estimates of the full GBNEL ensemble. A Kolmogorov-Smirnoff-test (KS-test) will then be used to calculate a p-value making a statement about whether the uncertainty estimates produced by an ensemble with a lower number of members and the uncertainty estimates of the full ensemble are drawn from the same distribution. In order to promote a good understanding of the used approach, the Wasserstein distance and the KS-test are introduced in more detail:

- **Wasserstein distance**

The Wasserstein distance calculates the minimum amount of work required to transform a distribution u into a target distribution v . The two distributions are often depicted as two piles of earth where one pile is transformed into the second one. The amount of soil and the distance it has to be moved represents the Wasserstein distance. It is therefore also-called the *Earth Mover's Distance* (EMD) (Panaretos & Zemel, 2019). Its formula for two 1-D distributions is given by

$$W_p(u,v) = \inf_{\pi \in \Gamma(u,v)} \int_{\mathbb{R} \times \mathbb{R}} |x - y| d\pi(x,y) \quad , \quad (6.1)$$

where $\Gamma(u,v)$ is the set of all probability distributions on $\mathbb{R} \times \mathbb{R}$ whose marginals are u and v . Speaking in the earth moving analogy, $\Gamma(u,v)$ represents all possible ways on how to move soil from u to transform it into v . $|x - y|$ is the distance between two points of u and v that are transformed into each other. $d\pi(x,y)$ is the respective mass, respectively, joint probability that is associated with these points. The infimum of all possible $\Gamma(u,v)$ integrals over all points x and y is the Wasserstein distance $W_p(u,v)$ (Ramdas, Trillos, & Cuturi, 2017). See Figure 6.8 for an illustrative example of the Wasserstein distance in 2-D.

- **KS-test**

The KS-test tests if a sample distribution is sufficiently adjusted to a given basic distribution. Its null hypothesis H_0 is that the two distributions tested are equal, and consequently, the alternative hypothesis H_A states that they are different. In this experiment, the sample distribution is generated by the ensembles with only a portion of ensemble members. The basic distribution is provided by the full ensemble sample uncertainty estimates. The KS-test works independently from the assumed underlying distribution and returns valid results also for small sample sizes. It can be applied to continuous and discrete distributions. Its statistic is calculated by determining the absolute frequencies E presuming H_0 . This allows for the calculation of the cumulative

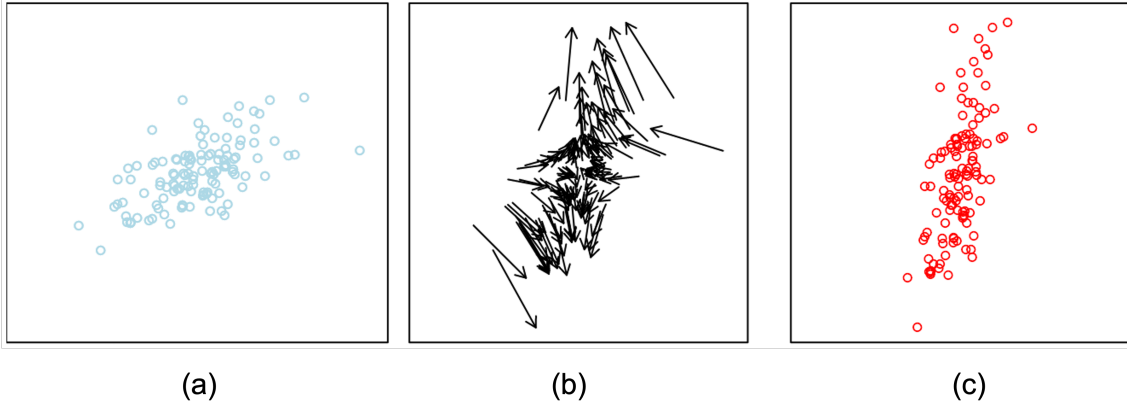


Figure 6.8 **Wasserstein distance in 2-D**

The distribution (a) is transformed into the distribution (c). (b) shows the optimal solution on how to move each point from (a) that minimizes the amount of work needed to transform it into (c). The Wasserstein distance is the sum over all lengths of the depicted vectors in (b) times their associated weight. In this example, all weights are equal. Illustration from (Panaretos & Zemel, 2019).

frequencies F_E . The same is done for the sample distribution, which results in F_B . The absolute biggest difference of F_B and F_E divided by the number of samples represents the test statistic D :

$$D = \frac{\max |F_B - F_E|}{n} \quad (6.2)$$

A p-value can be obtained with the help of a lookup table for different sample sizes and significance levels (Sachs, 2004).

6.2.2 Efficiently Estimating Model Uncertainties: Results

The Wasserstein distances between the sample uncertainty estimates of the full ensemble trained on the GBNEL data and the ensembles with only a portion of ensemble members are depicted in Figure 6.9. They were calculated using the implementation from the Python library `SciPy` (The SciPy community, 2021b).

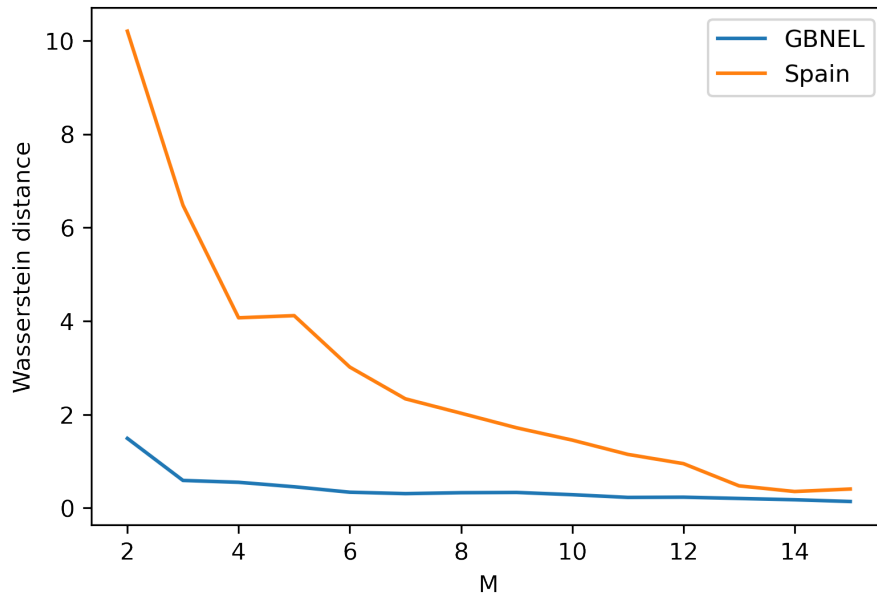


Figure 6.9 **Wasserstein distances of sample uncertainty estimates**

The blue graph depicts the Wasserstein distances between the sample uncertainty estimates of deep ensembles with only M members and the full deep ensemble with 16 members trained on the GBNEL data on the GBNEL test data. The orange graph depicts the distances on the ES test data.

The graph for the GBNEL test data shows that the sample uncertainty distribution obtained with only six ensemble members has already converged towards the estimates of the full ensemble. The addition of more members does not significantly increase the quality of uncertainty estimates. For the Spanish test data, an ensemble member count of 13 represents the uncertainty estimates of the full ensemble reasonably well. Figure 6.10 compares each individual sample uncertainty estimate of the Spanish test data obtained with $M = 5$ (blue) and $M = 13$ (orange) ensemble members to the full ensemble estimate. If the Wasserstein distance were 0, all samples would be located on the red line, meaning that the distributions are the same. It is observed that the sample uncertainty estimates of the ensemble with five members deviate significantly from the red line. It underestimates uncertainties of low magnitude and overestimates sample uncertainties of higher magnitude. An ensemble with 13 members diverges only slightly from an optimal fit with seemingly no systematic tendencies of under- or overestimating sample uncertainties in comparison to the full ensemble.

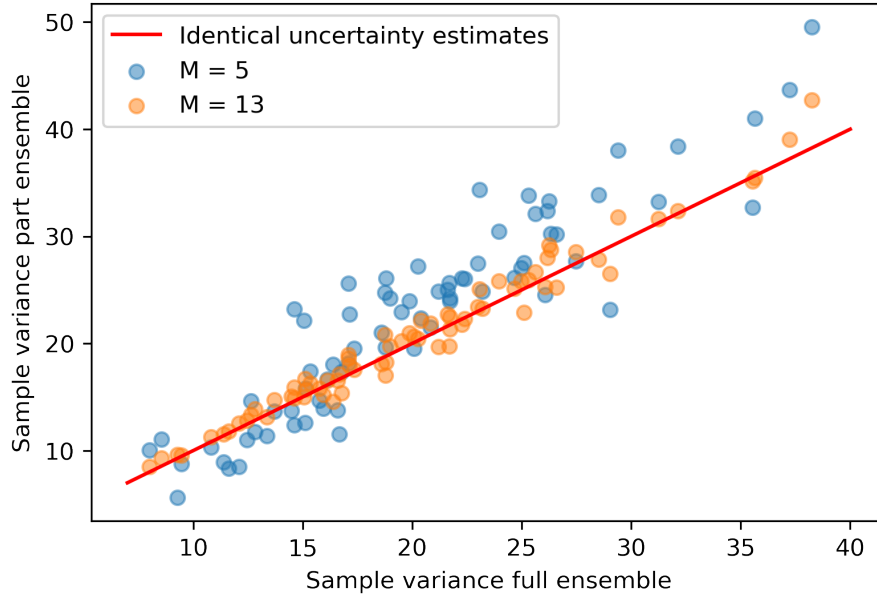


Figure 6.10 **Comparison of individual sample uncertainty estimates**

Sample variances obtained with $M = 5$ (blue) and $M = 13$ (orange) ensemble members over the sample variances obtained with the full ensemble. The red line depicts the location of sample variances if the distributions are identical.

Because the Wasserstein metric is not bound to a specific interval, the level of Wasserstein distances due to the different properties of the GBNEL and ES data in Figure 6.9 is substantially lower for the GBNEL test data than for the ES test data. This is because the uncertainty estimates for the GBNEL test data are low in comparison to the ES test data (see Figure 6.4), keeping the distances between the sample variances low in the first place. Figure 6.9 also gives reason to believe that the addition of more ensemble members is more beneficial in cases where there is high epistemic uncertainty. Increasing the information presented to the model decreases the epistemic uncertainty and possibly decreases the number of ensemble members needed in a progressing active learning approach. Examining this potential connection would be an interesting working hypothesis of another work.

As mentioned, a KS-test is used to determine whether the uncertainty sample distribution obtained with 13 ensemble members is sufficiently adjusted to the uncertainty sample distribution obtained with all 16 ensemble members. The calculation of the KS statistics was implemented with the Python library `SciPy` (The SciPy community, 2021a). For the GBNEL test data, the KS statistic was 0.0485 with a p-value of 0.9998 and for the Spanish test data 0.0548 with a p-value of 0.9999. The null hypothesis is therefore not rejected, meaning that both 13 member ensemble

uncertainty estimates are sufficiently adjusted to their respective full ensemble uncertainty estimates.

It is argued that the uncertainty estimates in Figure 6.4 on the GBNEL data can serve as an approximation of the lower uncertainty bound of the mixed GBNEL and ES data for the active learning approach in the following section. The uncertainty estimates on the ES data can be interpreted as an upper uncertainty bound of the mixed data. These assumptions are made because a robust ensemble trained on the GBNEL data will likely not become remarkably more certain about its predictions on GBNEL test data without the provision of more GBNEL training samples. The experiments in section 6.1 have also shown that the Spanish data stems from a different basic distribution. This suggests that the presentation of Spanish training data will likely have a rather low impact on the ensemble uncertainty estimates on GBNEL test data. Retraining an ensemble with the GBNEL and ES training data will, on the other hand, most likely decrease the epistemic model uncertainty on the ES test data.

This line of argument is not entirely sound. The blending of GBNEL and ES data will result in a different data distribution which leads to a different model, respectively, deep ensemble. It will output different uncertainty estimates. Nevertheless, the uncertainty estimates of the GBNEL trained ensemble should still serve as an adequate approximation of the maximum and minimum uncertainties observed on a mixed dataset. Therefore, it is sensible to only consider 13 ensemble members for the subsequent tasks without deteriorating the uncertainty estimates significantly. This will also have the advantage of lower training and evaluation times. A test on evaluation speed of the whole ensemble on the Spanish test data resulted in the following figures:

- 13 ensemble members: 14.8 s \pm 538 ms per loop (mean \pm std. dev. of 7 runs, 20 loops each)
- 16 ensemble members: 20.2 s \pm 647 ms per loop (mean \pm std. dev. of 7 runs, 20 loops each)

As in previous tests, a 2020 Apple M1 Macbook with 8GB of RAM was used. An adaption of the ensemble size from 16 to 13 decreases the evaluation time by 26.7% while retaining a high quality of uncertainty estimates. Therefore, only 13 deep ensemble members are considered for the following experiments on the active learning approach.

6.3 Hypothesis 3: Active Learning

The third hypothesis tries to answer the question of whether the training process and the ensemble performance itself can be improved by following an active learning approach. As suggested in section 4.3, it is hypothesized that the selection of samples with high model uncertainty for the next training step allows for higher sample efficiency. This relationship should manifest in higher prediction accuracies stemming from a model trained with curated samples in comparison to a randomly trained model with an equal number of samples.

In hypothesis 1, it was shown that the overall uncertainty estimates are lower for data that stems from the basic distribution the model was trained on than for other data (Figure 6.4). This also suggests that the model uncertainty estimates on the validation data will decrease with each iteration of the active learning procedure. According to (Settles, 2009) an active learning approach is also advantageous when searching for the optimal decision boundary between classes. Hence, the model performance in terms of validation IoU scores should be superior compared to an ensemble trained with random sample selection.

6.3.1 Active Learning: Procedure

In order to test the above-stated hypotheses, two different deep ensembles are trained. The first deep ensemble is trained following an active learning approach meaning that the samples presented to the models are actively chosen. The estimated sample uncertainty, represented by the variance between the different ensemble member outputs, is used as a metric to discriminate between samples that are selected respectively not selected for the training. The training of the second deep ensemble follows the traditional approach of using random samples.

The results in section 6.2 have stated that the use of only 13 ensemble members is sufficient in order to obtain high-quality uncertainty estimates. Both deep ensembles used in this section, therefore, consist of 13 ensemble members. Each ensemble member is initialized with random weights, and the sample order is altered for each epoch. The Adam optimizer with a learning rate of 1E-3 is used. The batch size is set to 16, and the cross entropy loss serves as the loss function. The experiment was carried out in the following way:

First, the previously created GBNEL and ES datasets were loaded, merged, and randomly shuffled. This resulted in 700 training images and 116 test samples. A *seed dataset* is then derived, which only inherits 25% of the samples from the original

dataset. The seed dataset has the purpose of providing a data basis for the training of an initial ensemble. The size of the seed dataset was chosen to be 25% of the original dataset because 175 samples should provide an adequate amount of information for a LinkNet architecture to provide sensible outputs for the task of detecting pipeline pathways. Additionally, this leaves a profound data basis of 525 samples left to be used for an uncertainty-driven training setup.

The initial ensemble, respectively, *seed ensemble* is used to estimate the sample uncertainties of the remaining 75% of data in order to initiate the active learning process. The trained seed ensemble is saved and copied. This way, both deep ensembles – the one trained with an active learning approach (AL) and the one trained with random samples (RAND) – make use of the same underlying models in the beginning and therefore share a starting point. This ensures the comparability of the two ensembles. The continued training of the deep ensembles differs in the way new samples are acquired for the next training step. Algorithm 1 shows the pseudocode for the training of the ensembles on a high level.

Algorithm 1 Progressive training

Require: seed data, seed ensemble, test data, remaining training data, batch size, epochs
training_data \leftarrow seed data
data \leftarrow remaining training data
test_data \leftarrow test data
n \leftarrow batch size
while $\text{len}(\text{data}) > 0$ **do**
 ensemble \leftarrow load(*best_models*_{*t*-1}) \triangleright Seed ensemble in first iteration $t = 0$
 sample_uncertainties \leftarrow *ensemble*.estimate_uncertainties(*data*) \triangleright Only AL
 data.sort(key = *sample_uncertainties*, reverse = **True**) \triangleright Only AL
 training_data.extend(*data*[:*n*]) \triangleright Extend training data with new samples
 data \leftarrow *data*[*n*:] \triangleright remove new samples from remaining data
 for *model* **in** *ensemble* **do** \triangleright Train all ensemble members
 for *i* **in** range(epochs) **do**
 model.train(*training_data*)
 if Loss(*model*_{*i*}.eval(*test_data*)) < Loss(*model*_{*i*-1}.eval(*test_data*)) **then**
 *model*_{*i*}.save(*best_model*) \triangleright Save best model for future use
 else
 Continue

The batch size, which represents the number of new samples added to the training dataset after each training iteration, was chosen to be 70. An exception is the last iteration, where 105 samples (15% of the total data) were added. Thus, the two deep ensembles were progressively trained on 25%, 35%, 45%, 55%, 65%, 75%, 85%, and 100% of the available GBNEL & ES data. The batch size of 70 was chosen because it

provides a variety of training checkpoints to compare both ensembles while keeping the invested time and effort put into the training at a manageable amount.

6.3.2 Active Learning: Results

The main results of the previously described method are depicted in Figure 6.11. It is observed that both ensembles increase their performance in regard to the validation IoU at an almost equal rate for the first three training iterations. The validation IoUs for the seed ensembles are obviously identical because they share the same ensemble members. The performances of the AL and RAND approaches start to diverge between the training iterations, where 45% and 55% of all available samples are presented to the NNs. It is therefore conjectured that the minimum amount of training data required for the ensembles to sufficiently grasp the task of detecting pipeline pathways for the GBNEL and ES data is somewhere between 45% and 55% of all available samples. It is also observed that the differences between individual ensemble member IoU scores decrease between these two iterations. This is indicated by the shaded areas around the solid lines where the lower bound represents the worst-performing ensemble member and the upper bound the best-performing member. The IoU scores with 55% of available training data are 0.487 for the AL ensemble and 0.477 for the RAND ensemble. In the following two training iterations (65% and 75% of data), the performance of the RAND ensemble does not increase. The actively trained ensemble shows slight improvements in prediction accuracies, increasing its validation IoU to 0.500. The ensemble already converged with only 75% of the available samples. The randomly trained ensemble still improves when adding another 70 samples converging at a validation IoU score of 0.496. The validation scores for both ensembles trained on 100% of the data are 0.505 (AL) and 0.498 (RAND). Although the differences in ensemble performances are small, Figure 6.11 also shows that past the 45% mark, the worst- and best-performing AL members outperform their RAND counterparts in almost all iterations. Therefore, it is confirmed that in this case the active learning approach is more sample efficient and outperforms the randomly trained ensemble.

It was also hypothesized that the model uncertainty estimates on the validation data would decrease with each iteration of the active learning procedure. This will be answered using the results shown in Figure 6.12. Indeed, except for the iteration from 45% to 55% of available data, the median uncertainty estimates are continuously decreasing with each iteration of the training procedure. This not only holds true for the actively trained ensemble but also for the randomly trained ensemble. Because the epistemic uncertainty is likely to decrease with more information

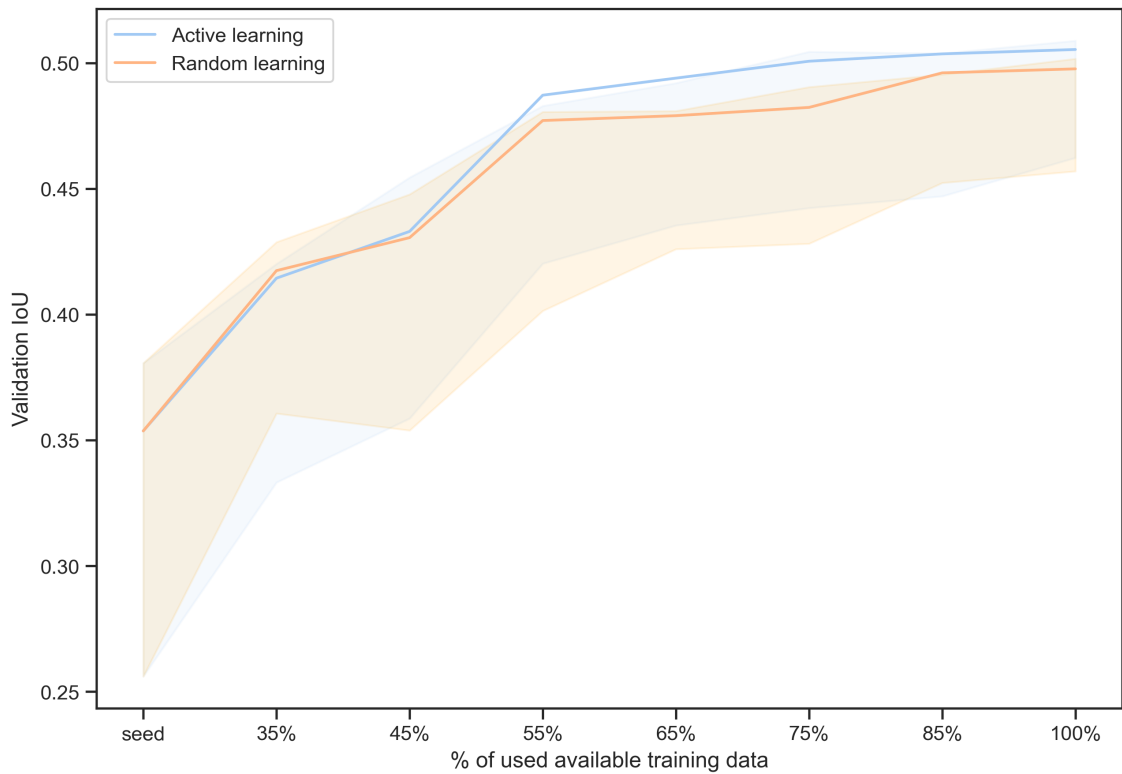


Figure 6.11 **Comparison of ensemble validation IoUs**

Solid lines show the validation IoUs of the respective ensemble predictions for each training iteration. The shaded areas are bound by the worst and best-performing ensemble members in respect to the IoU score.

presented to the model, this was expected. The fact that the level of uncertainty estimates of the RAND ensemble is consistently slightly lower than that of the AL ensemble was, on the other hand, not expected. Even though the interquartile range (IQR) is smaller for the AL ensemble after converging at 75% of available training data – meaning that the uncertainty estimates are gathering closer around the median – its median sample uncertainty is still higher. A closer examination of the circumstances did not provide sufficient insight to extract an educated explanation for the observations. Further experiments are required to gather more information regarding the subject. This will not be covered in this thesis but can serve as an interesting working hypothesis of another work.

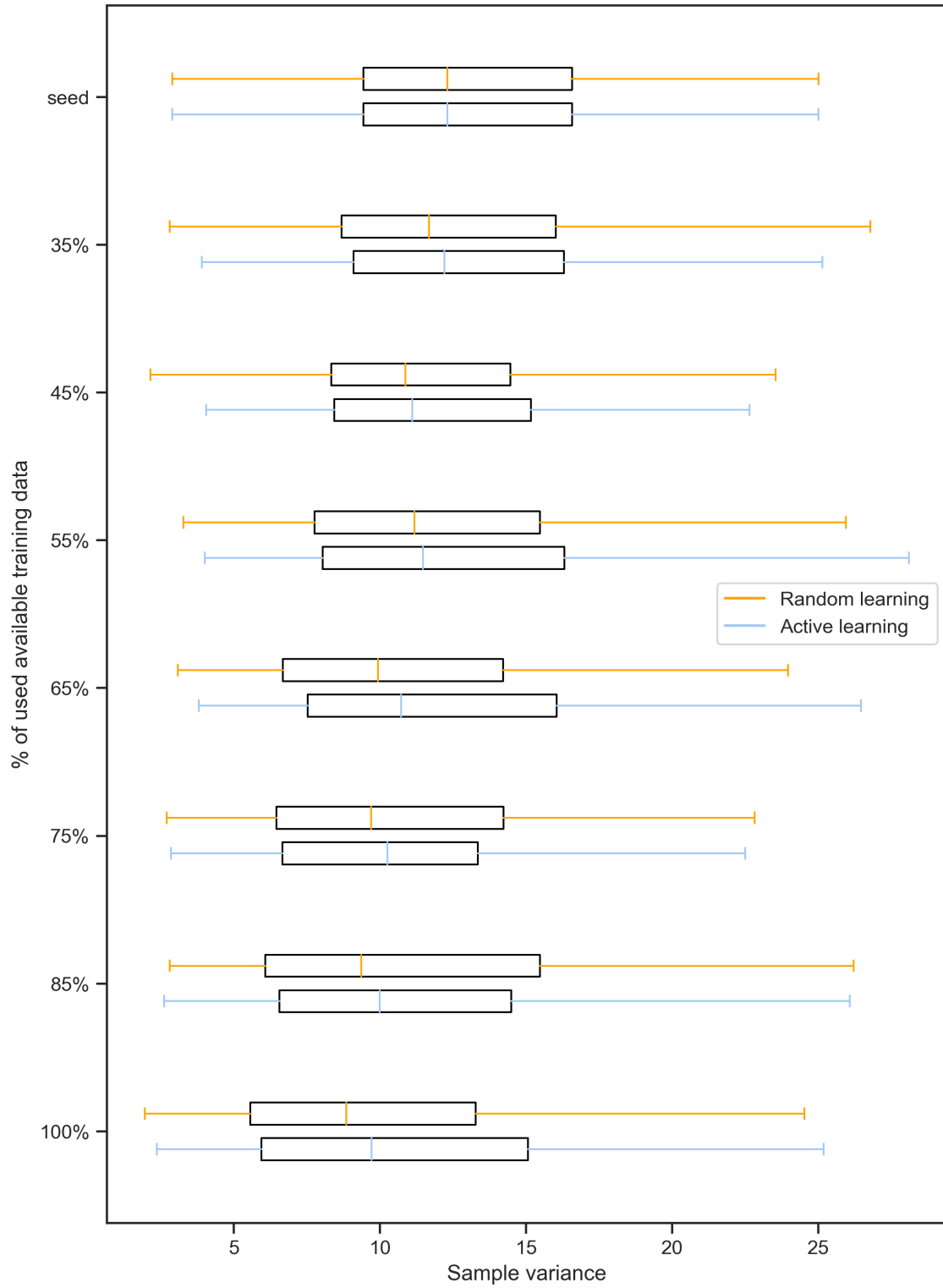


Figure 6.12 **Comparison of sample uncertainty estimates**

Besides the hypotheses in question, a lot of curiosity stems from the fact that the highest IoU score of the active learning ensemble trained on the merged GBNEL & ES data performs worse than the GBNEL ensemble from section 6.1 on their re-

spective validation dataset. Therefore, Figure 6.13 compares the AL ensemble IoU scores on the separated GBNEL and ES datasets. It is observed that the AL model

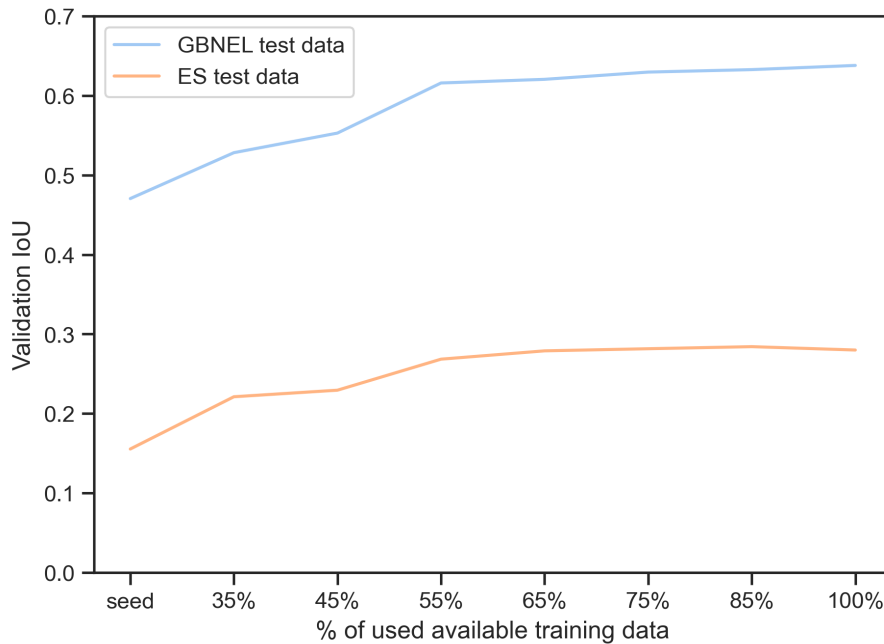


Figure 6.13 **Comparisons of IoU GBNEL and ES**

The highest IoU score of the AL ensemble on the GBNEL validation data is 0.638 and 0.284 on the ES data.

performs even better than the GBNEL ensemble on the GBNEL validation data (0.64 vs. 0.63). The ostensible deterioration of the model performance originates exclusively from the ES data, where only an IoU score of 0.28 is reached. A qualitative analysis of the uncertainty estimates for each training iteration provides further insights:

- The bulk of the 70 samples with the highest uncertainty estimates provided from the seed ensemble are from Spain. With each training iteration, the ratio between ES and GBNEL data selected for the next batch shifts towards more GBNEL samples. The last batch of 105 samples consists of mostly GBNEL data. This is a clear indicator for the circumstance that the model is significantly more uncertain when predicting Spanish pipeline pathways.
- The model performance seems to suffer from low contrasts between the background and the right-of-way. These low contrasts are mainly present on Spanish satellite images (see Figure 6.6).

Additionally, a third test was conducted to rule out the possibility that a model trained on merged GBNEL and ES data performs significantly worse on ES test

data than a model trained exclusively on ES data. A single LinkNet+ResNet-18 model was trained only considering the ES training data. The batch size was set to 16, and all other parameters were kept the same as in the previous training setups in this work. The best validation IoU score was 0.30, meaning that the model is also not able to satisfyingly detect pipeline pathways in Spain when exclusively trained on ES data.

The previously made statement in section 6.1 that the model performance will likely drastically increase for detecting Spanish pipeline pathways when provided with more ES data has to be revised. Even though there is a significant improvement from a validation IoU of 0.12 (GBNEL trained ensemble) to 0.28, the ensembles are still not performing satisfyingly well on ES data. Further qualitative analysis show that the main sources of error in this regard are false negative model predictions. Problems often arise out of low contrasts between the right-of-way and its surroundings. In the GBNEL regions, low contrasts are especially abundant when the pipeline pathway is close to arable land or other areal structures like towns and industrial sites Figure 6.14a. In Spain, on the other hand, low contrasts between the pipeline pathway and its background are often omnipresent because of sandy and rocky soils Figure 6.14b. Uncertainty estimate series produced by the active learning ensemble over all training iterations provide further insights into the problem of low contrasts in the input images. As shown in Figure 6.15a₁, the seed ensemble initially produces some significant uncertainties around pipeline-like structures for a typical GBNEL sample. Additional background uncertainty noise is present. With each training iteration, the model uncertainty and especially the background uncertainties are decreasing. The ensemble ends up with only small uncertainty estimates around the actual pipeline pathway and at right-of-way-like structures (Figure 6.15a₈). The final prediction of the ensemble (Figure 6.15a_p) is very good.

The bottom example, Figure 6.15b, provides an uncertainty estimate series for an input image that suffers from a low contrast between the right-of-way and its background. The seed ensemble expresses high uncertainty estimates around a wide corridor spanning from the mid-left to the mid-right of the image (b₁). With each training iteration, this corridor narrows down (b₂ – b₈). The bulk of shown uncertainties are not related to the pipeline pathway itself. The ensemble is consequently not able to detect pipeline pixels (b_p) but suffers from background noise that has similar spectral properties as pipeline construction sites. This problem is frequently observed for satellite imagery of pipeline pathways in Spain, which ultimately deteriorates the overall model performance on ES data significantly.

This concludes the chapter on the hypotheses and results of this work. A variety of

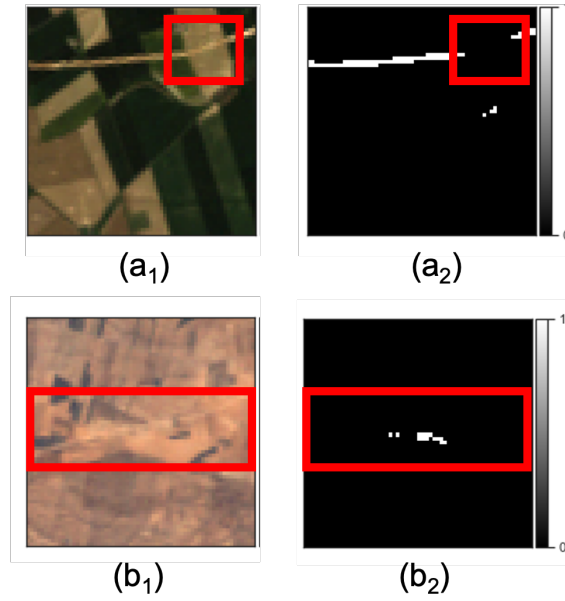


Figure 6.14 **Low pipeline-background contrasts in GBNEL and ES data**

The red squares indicate the region of interest for each pipeline path. The model input (a_1) shows a typical error source generating false negative predictions for GBNEL data due to low image contrasts. The model output is depicted in (a_2). The area of low contrast is bound to a small subarea within the satellite image. (b_1) represents a characteristic classification error on the Spanish test dataset. As seen in b_2 , the ensemble is not able to detect any true pipeline pixels at all. The area of low contrast extends to the whole input image because of the sandy/rocky soils with low vegetation that are present in many parts of Spain.

valuable insights and implications regarding the initially stated research questions have been provided. These will be discussed in the following chapter 7.

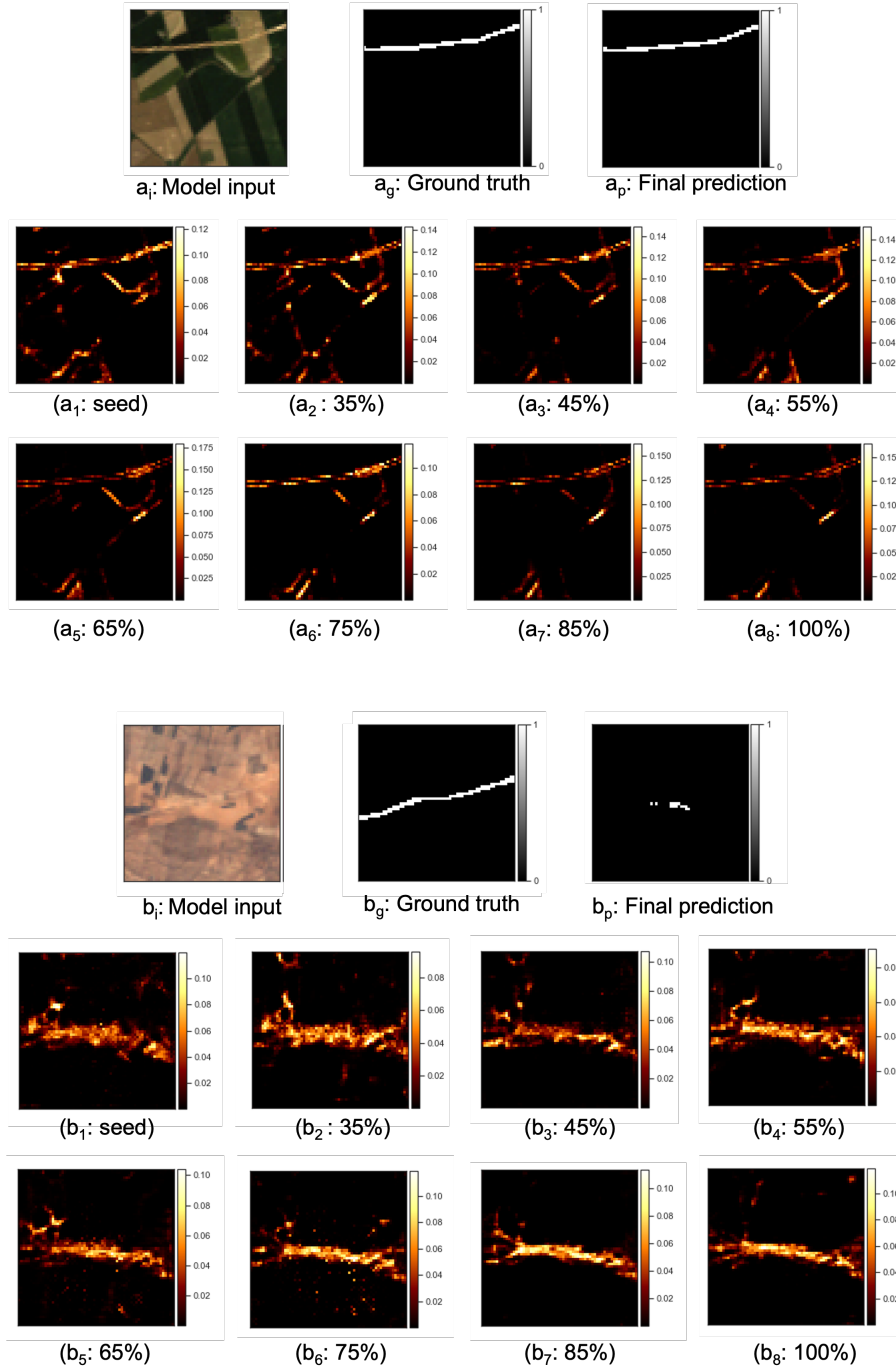


Figure 6.15 **Two uncertainty estimate series for typical GBNEL and problematic ES data**

The top example represents a typical GBNEL sample, while the bottom one suffers from low right-of-way-background contrasts. The model inputs, ground truths, and final predictions of the ensemble actively trained with 100% of available GBNEL & ES training data are depicted in the respective top row. Subplots a_1 – a_8 and b_1 – b_8 show the estimated model uncertainties for each training iteration, where the percentage represents the proportion of total training data used.

7 Discussion

The first insights of this work have been provided by the comparisons of the different model architecture and encoder-backbone combinations. Various state-of-the-art backbones and two model architectures – U-Net and LinkNet – have been considered, described, and tested. In hindsight, the decision to carry out the comparisons proved to be of great influence for all subsequent tasks. The usage of the default architecture-backbone combination in the Segmentation Models library – namely an U-Net architecture with a VGG16 backbone – would have resulted in an almost threefold increase in training time compared to the used LinkNet-ResNet-18 combination (Table 5.1). Additionally, the number of parameters would have been doubled, resulting in higher memory requirements. Especially when using deep ensembles where multiple realizations of the same model architecture are used, this would have been critical. The prediction accuracy of the subsequently used LinkNet-ResNet-18 model, represented by the IoU on the validation data, only slightly deteriorated in comparison to the U-Net-VGG16 model that performed best in this regard. This has been proven to be neglectable. If a model to detect pipeline pathways is deployed in future projects, the use of a LinkNet-ResNet-18 architecture is recommended. Additional work should be put into the tuning of the model hyperparameters like the chosen optimizer, the learning rate, and the training batch size. This was outside the scope of this thesis but could improve the model performance further.

The pipeline detection model developed in (Dasenbrock et al., 2021) has proven to be capable of generalizing well from training data originating from Great Britain to Northern Germany. The thesis at hand used a similar model but applied it to a more differing and heterogeneous region compared to the training data in order to test its generalizability: Spain. Insufficient IoU scores showed that the model is not able to satisfyingly detect pipeline pathways in Spain. It will be of great importance when applying the model to new regions that it is also specifically trained for the new regions. While the model is permanently applied to new regions and consequently more training data is added, the need for new training data will diminish

with time. This is because the knowledge of the model will become broader, and the differences between new regions and the regions already shown to the model will likely decrease. To speed up this process and to train more sample efficient, the potential of an active learning approach was investigated.

The experiments related to the active learning approach have shown that it is advantageous to actively choose training samples. This not only allows for an efficient adaption of the trained models to other regions because only small efforts have to be put into the labeling of new training data. It also increases the model performances overall. In order to keep the training times within a manageable magnitude, the number of samples added for the next training iteration was set to 70. This batch size proved to be a reasonable amount of samples for the evaluation of whether an active learning approach is feasible at all. When deploying the active learning approach in the context of a project creating dependable European gas pipeline maps – like SciGRID_gas – the batch size should be adjusted. The reasoning for this is threefold:

- It is supposed that more training iterations with smaller added training sample sizes allow for a better and faster adjustment of the decision boundary between different classes (*pipeline* and *background*). This is because large batch sizes could overestimate the importance of the expressed model uncertainties of a single training iteration in regard to the overall training process. This can be interpreted as temporal overfitting of the model towards its current uncertainty estimates. Thus, the training process becomes less efficient, and the model possibly does not capitalize on its full potential.
- Closely related to the reasoning of the previous bullet point is the fact that exposing a model to new samples alters its epistemic uncertainty in regard to the remaining samples. The sample uncertainty rankings vary, influencing the order in which samples are presented to the model and if they are presented at all.
- The use of high batch sizes increases the effort that needs to be put into manual dataset labeling for each training iteration drastically. This is due to the reasons stated above but also because a convergence of the model can only be observed time- respectively sample-delayed.

Due to the low-tier hardware used for this work, the possible additional needs in training time due to smaller added batch sizes can easily be coped with by adding more computational resources.

Another influencing factor that is related to the deployment of an active learning approach considers the development of an active learning workflow. It should answer the questions on how and at which point of the training process unlabeled samples are generated and possibly labeled. The method used for the download of Spanish satellite imagery described in section 5.3 can serve as a valid starting point: A location consisting of a latitude and longitude and a period of time is chosen by the practitioner. The respective satellite imagery is downloaded and preprocessed to obtain the final image dimensions. Afterward, the specific sample uncertainty can be estimated by a deep ensemble determining whether the sample needs manual labeling. When reaching a certain threshold of new labeled samples, the ensemble will be retrained. This process is continuous.

In subsection 6.2.2, it was observed that the uncertainty estimates, represented by the variance of the ensemble member outputs, converge earlier with fewer members when the epistemic uncertainty is low. This observation needs further verification but hints and the possibility of removing single ensemble members with a progressing active learning approach. This would benefit computation and memory requirements and therefore make the deployment of the model more efficient. The use of the Wasserstein distance and a KS-test proved to be valuable in order to sensibly decrease the number of ensemble members.

Another key insight of this work is provided by the deteriorated model performance on satellite imagery from Spanish pipeline pathways. This holds true for all deep ensembles and models trained regardless of their training dataset composition. It is estimated that there is no necessity to train multiple different models for different regions but sufficient to develop one general deep ensemble that can be applied to the whole of Europe. Implied that the detection of pipeline right-of-ways in one specific region is closely related to the detection of pipeline pathways in other regions, knowledge gained through samples from other regions should improve the model performance for that respective region. This statement is an example of the well-established *transfer learning* approach where the knowledge of a model trained in one setting is exploited to improve the prediction accuracies in another setting (Goodfellow et al., 2016).

Analyses of the unsatisfactory model performance on the ES data suggest that low right-of-way to background contrasts are the main source of error. This is backed by the model outputs and the uncertainty estimates provided by the ensembles. The Spain dataset has proven to be more heterogeneous than the GBNEL dataset.

The ES data includes a variety of low contrast images. For example, some show a sandy, respectively, rocky soil, and others depict a comparably higher amount of vegetation. If we consider these two examples as different subclasses of ES images, each subclass only possesses a very small sample size. The individual sample sizes are too low to train a model in a reliable manner. The amount of samples has to be increased substantially to cover each subclass sufficiently. An active learning technique should prove to be very useful here. Another approach could be additional preprocessing steps for images stemming from regions with said low contrast characteristics. Contrast and color adjustments can, for example, benefit the image quality (Aimi Salihah, Mashor, Harun, Abdullah, & Rosline, 2010) and maybe improve the visibility of pipeline right-of-ways to the model. Because the adjustments would impact the data distribution itself, the model needs to be retrained with the adjusted images. Images that would be provided to the model to predict pipeline pathways have to be adjusted in the same way as well.

A different take on the low ensemble performance on Spanish satellite imagery would be to neglect Spain as a whole. As seen in Figure 5.9, the OSM data density proves to be very high, probably depicting all Spanish gas transmission pipelines. Additionally, the Spanish TSO Enagás provides the build years for all pipelines on its website. Therefore, the detection of pipeline pathways in Spain to create an accurate data model of the region might not be required at all. A focus on other regions can be much more fruitful. The SciGRID_{gas} project has shown that the OSM data availability is substantially lower for Eastern and South-Eastern Europe (Diettrich et al., 2021). The majority of regions in these parts of Europe demonstrate higher vegetation than Spain, as seen in Figure 5.8. The comparisons between GBNEL and ES data has shown that this increases the contrast between a newly established right-of-way and its surroundings. It is therefore claimed that the trained models will likely perform well in eastern and south-eastern Europe. This has to be further elaborated when deploying a model to detect all pipeline pathways in Europe. Minor deteriorations in model performances for these regions can be counteracted quickly by the use of the earlier discussed active learning approach. The downside of simply dodging the problem of low contrast images, like they are occurring in Spain, is that this also applies to other regions where the OSM data is sparse. Examples are some parts of Italy and Greece. The elegant solution would be to increase the amount of low ROW-background contrast training data to decrease the epistemic model uncertainties and increase the model performance in the regions of question.

Another noticeable source of error that resulted in false positive pipeline predictions were pipeline pathway-like structures. A visual inspection of the model outputs and uncertainty estimates reveals that especially dirt roads and field boundaries seem to be problematic. This implies that further preprocessing steps have to be established to improve the model performance for future usage. The mentioned pipeline pathway-like structures can be filtered out before an image is presented to the model. Another option would be that the location of these structures could be fed into the ensemble as an input. Either way, OpenStreetMap provides a profound database that is recommended to be used for the preceding filtering of roads and potentially other available structures.

At last, the satellite data source is addressed. The Landsat 5 mission ended in 2013. Obviously, there is no Landsat 5 data available for pipelines that were built after the ending of the mission. An alternative could have been the Landsat 7 mission. It started in 1999 and provides, among others, the same relevant bands and resolutions as the Landsat 5 mission. The problem is that the Scan Line Corrector of Landsat 7 failed in 2003, resulting in significant data gaps for all satellite images. This makes them most likely impractical to the trained ensembles. Further information can be found in (USGS, 2021b). Another option – the Landsat 8 satellite, which is on duty since 2013 – could prove to be a valid alternative to the Landsat 5 data. The difficulty here lies in its sensor *Operational Land Imager (OLI)* that deviates in the bandwidths compared to the Landsat 5 Thermal Mapper but shares the same resolutions (USGS, 2021c). The trained ensembles need to be reevaluated on the Landsat 8 data in order to make valid statements about its usefulness. If the two different data sources prove to be incompatible, the ensemble trained on the Landsat 5 data likely provides an advantageous starting point to retrain it to be applied to Landsat 8 data. This is in line with the previously mentioned transfer learning approach.

The next chapter will cover the conclusion of this work and provide a brief outlook on the necessary steps to potentially apply the model to the entire European gas transmission network.

8 Conclusion & Outlook

The main motivation behind this work was it to further develop the approach from (Dasenbrock et al., 2021) of combining remote sensing and modern machine learning algorithms to detect gas transmission pipelines in Europe. The general concept of the method has been proven in the said study and confirmed in this master thesis. However, in order to put the method to use, for example, to create an open-source data model of the European gas transmission grid, like attempted in the SciGRID_gas project, further studies on the subject had to be conducted. This included the testing of the initial model, trained on satellite imagery from Great Britain and Northern Germany, in other, more heterogeneous European regions. The thesis at hand used the Spanish gas transmission grid for this purpose. It has been shown that it is necessary to specifically adapt the model to currently unknown regions to obtain reliable pipeline detections. Therefore, additional training data needs to be generated in regions expressing poor model performances. Because the generation of new training samples used to train the model turned out to be a laborious task, this work introduced the active learning approach. By means of deep ensembles, the model uncertainties – represented by the variance between the member model outputs – have been estimated. These epistemic uncertainty estimates were used to not randomly pick samples for the training of the model but to actively choose particular samples that are of high curiosity to the model, respectively, deep ensemble. The active learning approach allowed the deep ensembles to more efficiently adapt to a new region by using fewer samples while showing better prediction performance. Therefore, it is highly recommended to follow this approach when deploying the model in the future to detect pipeline pathways in Europe. Additionally, a sensible number of deep ensemble members has been derived by the comparison of different sample uncertainty estimate distributions. This decreased the memory and computational requirements when using deep ensembles.

This work not only reveals valuable insights into ways on how to more efficiently train and improve the model but also tested a variety of architecture-backbone combinations to provide further efficiency gains: The use of a LinkNet+ResNet-18 model proved to cut the training time of each model member by more than 63% and halved

the number of parameters needed in comparison to the model used in (Dasenbrock et al., 2021). This is especially vital when using deep ensembles for uncertainty estimates.

The uncertainty estimates also gave a good intuition for the main sources of misclassification. Particularly low contrasts between the right-of-way and its surroundings, which mainly occurred on Spanish satellite data, resulted in unsatisfying model performance. Subsequent projects can take advantage of the generated Great Britain, Northern Germany, and Spain datasets and further train the model to also be able to detect pipelines in low right-of-way-background contrast settings. Contrast and color adjustments could prove to be useful.

According to the authors' assessment, the next steps in regard to the further development of the method concern the following three major working packages:

- **Establishing an active learning workflow**

To put the developed method to use in an efficient manner, it is recommended to develop a mostly automated active learning workflow. It should be capable of downloading satellite imagery at a beforehand specified location and period of time, preprocessing it, evaluating its uncertainties, and providing it to the practitioner if manual labeling is required. After requiring a certain threshold of new samples, the model is retrained. This way, the method can be applied to the European gas transmission grid as a whole in a progressing and efficient manner.

- **Including additional satellite data sources**

The generation of new Landsat 5 data ended with the malfunction of its Thematic Mapper in 2011. Therefore, a new source of satellite imagery needs to be included for the detection of more recent pipeline pathways. The Landsat 8 mission appears to be a valid replacement solution. The model needs to be adapted accordingly, and additional tests have to be carried out to evaluate whether a single model is sufficient to capture pipeline right-of-ways on Landsat 5 and Landsat 8 data.

- **Providing auxiliary preprocessing steps to the satellite imagery**

It has been shown that the main factors deteriorating the model performance are twofold. Pipeline pathway-like structures, like roads and field borders, sometimes lead to false positive predictions. This can be counteracted by using, e.g., OSM data of said structures as another input to the model and should be implemented in following the projects. False negative predictions are particularly bound to low contrasts between the right-of-way and its sur-

roundings. Tests on whether this can be solved by altering the contrast and colors of the input images need to be carried out and eventually be applied by default. The addition of further measurements, like other bands from new satellite missions, can also prove to be helpful.

Following the above-stated working steps will likely result in a tool of great value to the creation of high-quality data models of the European gas transmission grid. These models can support profound political and societal decisions and eventually help overcome the challenges that arise out of the transition of the current European energy system toward a CO₂-neutral energy system.

References

- Aimi Salihah, A., Mashor, M., Harun, N. H., Abdullah, A. A., & Rosline, H. (2010). Improving colour image segmentation on acute myelogenous leukaemia images using contrast enhancement techniques. In *2010 ieee embs conference on biomedical engineering and sciences (iecbes)* (p. 246-251). doi: 10.1109/IECBES.2010.5742237
- Alkazraji, D. (2008). *A quick guide to pipeline engineering*.
- Arévalo, A., Niño, J., Hernández, G., & Sandoval, J. (2016). High-frequency trading strategy based on deep neural networks. In *International conference on intelligent computing* (pp. 424-436).
- Bjarnadottir, S., Li, Y., & Stewart, M. G. (2019). Chapter nine - climate adaptation for housing in hurricane regions. In E. Bastidas-Arteaga & M. G. Stewart (Eds.), *Climate adaptation engineering* (p. 271-299). Butterworth-Heinemann. Retrieved from <https://www.sciencedirect.com/science/article/pii/B9780128167823000097> doi: <https://doi.org/10.1016/B978-0-12-816782-3.00009-7>
- Carl Zeiss Microscopy GmbH. (2021). *Apeer anotate*. Retrieved 16.07.2021, from <https://www.appeer.com/home>
- Carvalho, R., Buzna, L., Bono, F., Gutiérrez, E., Just, W., & Arrowsmith, D. (2009, 08). Robustness of trans-european gas networks. *Physical review. E, Statistical, nonlinear, and soft matter physics*, 80, 016106. doi: 10.1103/PhysRevE.80.016106
- Cerniauskas, S., Jose Chavez Junco, A., Grube, T., Robinius, M., & Stolten, D. (2020). Options of natural gas pipeline reassignment for hydrogen: Cost assessment for a germany case study. *International Journal of Hydrogen Energy*, 45(21), 12095-12107. Retrieved from <https://www.sciencedirect.com/science/article/pii/S0360319920307023> doi: <https://doi.org/10.1016/j.ijhydene.2020.02.121>
- Chaurasia, A., & Culurciello, E. (2017, Dec). Linknet: Exploiting encoder representations for efficient semantic segmentation. *2017 IEEE Visual Communications and Image Processing (VCIP)*. Retrieved from <http://dx.doi.org/10.1109/VCIP.2017.8305148> doi: 10.1109/vcip.2017.8305148
- Cityscapes Dataset. (2021). *Dataset overview - type of annotations*. Retrieved 06.12.2021, from <https://www.cityscapes-dataset.com/dataset-overview/>
- Cordts, M., Omran, M., Ramos, S., Rehfeld, T., Enzweiler, M., Benenson, R., ... Schiele, B. (2016). The cityscapes dataset for semantic urban scene un-

- derstanding. In *Proc. of the ieee conference on computer vision and pattern recognition (cvpr)*.
- Dasenbrock, J., Pluta, A., Zech, M., & Medjroubi, W. (2021). *Detecting pipeline pathways in landsat 5 satellite images with deep learning*.
- Diettrich, J., Pluta, A., & Medjroubi, W. (2021, May). *Scigrid_gas iggielgn*. Zenodo. doi: 10.5281/zenodo.4767098
- Dumoulin, V., & Visin, F. (2018). *A guide to convolution arithmetic for deep learning*.
- Enagás. (2021). *Transmission network - gas pipelines*. Retrieved 27.10.2021, from https://www.enagas.es/enagas/en/Transporte_de_gas/Red_de_transporte/Gasoductos
- EUGAL. (2021). *Von der deutschen ostsee bis nach tschechien*. Retrieved 03.12.2021, from <https://www.eugal.de/eugal-pipeline/trassenverlauf>
- European Parliament. (2017). *Gasversorgungssicherheit in europa*. Retrieved 08.12.2021, from <https://www.europarl.europa.eu/news/de/headlines/economy/20170911ST083502/infografik-gasversorgungssicherheit-in-europa>
- Federal Ministry for Economic Affairs and Energy. (2019). *Erdgasversorgung in deutschland*. Retrieved 08.12.2021, from <https://www.bmwi.de/Redaktion/DE/Artikel/Energie/gas-erdgasversorgung-in-deutschland.html>
- Gal, Y. (2016). Uncertainty in deep learning.
- Gal, Y., & Ghahramani, Z. (2016). Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning* (pp. 1050–1059).
- geojson.io. (2021). *Worldmap*. Retrieved 25.10.2021, from <https://geojson.io/#map=5/49.067/-0.483>
- Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning*. MIT Press. (<http://www.deeplearningbook.org>)
- GoogleEarthEngine. (2021). *A planetary-scale platform for earth science data and analysis*. Retrieved 25.10.2021, from <https://earthengine.google.com/>
- Gorelick, N., Hancher, M., Dixon, M., Ilyushchenko, S., Thau, D., & Moore, R. (2017). Google earth engine: Planetary-scale geospatial analysis for everyone. *Remote Sensing of Environment*. Retrieved from <https://doi.org/10.1016/j.rse.2017.06.031> doi: 10.1016/j.rse.2017.06.031
- Gustafsson, F. K., Danelljan, M., & Schon, T. B. (2020). Evaluating scalable bayesian deep learning methods for robust computer vision. In *Proceedings of the ieee/cvf conference on computer vision and pattern recognition workshops*

(pp. 318–319).

- Géron, A. (2019). The perceptron. In A. Géron (Ed.), *Hands-on machine learning with scikit-learn, keras, and tensorflow* (2nd Edition ed., p. 282–282). O’Reilly Media, Inc.
- Hartmann, A., Davari, A., Seehaus, T., Braun, M., Maier, A., & Christlein, V. (2021). *Bayesian u-net for segmenting glaciers in sar imagery*.
- He, K., Zhang, X., Ren, S., & Sun, J. (2015). *Deep residual learning for image recognition*.
- Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., ... Adam, H. (2017). *Mobilenets: Efficient convolutional neural networks for mobile vision applications*.
- Huang, G., Liu, Z., van der Maaten, L., & Weinberger, K. Q. (2018). *Densely connected convolutional networks*.
- Hüllermeier, E., & Waegeman, W. (2021). Aleatoric and epistemic uncertainty in machine learning: An introduction to concepts and methods. *Machine Learning*, 110(3), 457–506.
- IEA. (2019). *The future of hydrogen*. Retrieved from <https://www.iea.org/reports/the-future-of-hydrogen>
- Ioffe, S., & Szegedy, C. (2015). *Batch normalization: Accelerating deep network training by reducing internal covariate shift*.
- Isaac, T. (2019, 05). HyDeploy: The UK’s First Hydrogen Blending Deployment Project. *Clean Energy*, 3(2), 114–125. Retrieved from <https://doi.org/10.1093/ce/zkz006> doi: 10.1093/ce/zkz006
- Johnson, N., Gagnolet, T., Ralls, R., & Stevens, J. (2011). Natural gas pipelines. *Nature. org*, 1–9.
- Kingma, D. P., & Ba, J. (2017). *Adam: A method for stochastic optimization*.
- Kirillov, A., He, K., Girshick, R., Rother, C., & Dollár, P. (2019). *Panoptic segmentation*.
- Kramer, O. (2020). *Lecture Notes: Deep Learning, Carl von Ossietzky Universität Oldenburg*.
- Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25, 1097–1105.
- Kuutti, S., Bowden, R., Jin, Y., Barber, P., & Fallah, S. (2021). A survey of deep learning applications to autonomous vehicle control. *IEEE Transactions on Intelligent Transportation Systems*, 22(2), 712–733. doi: 10.1109/TITS.2019.2962338

- Lakshminarayanan, B., Pritzel, A., & Blundell, C. (2016). Simple and scalable predictive uncertainty estimation using deep ensembles. *arXiv preprint arXiv:1612.01474*.
- LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., & Jackel, L. D. (1989). Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4), 541–551.
- Liew, S. C. (2001). *Principles of remote sensing - satellite orbits*. Retrieved 03.12.2021, from <https://crisp.nus.edu.sg/~research/tutorial/spacebrn.htm>
- Liu, X., Deng, Z., & Yang, Y. (2018, Jun). Recent progress in semantic image segmentation. *Artificial Intelligence Review*, 52(2), 1089–1106. Retrieved from <http://dx.doi.org/10.1007/s10462-018-9641-3> doi: 10.1007/s10462-018-9641-3
- Medjroubi, W., Diettrich, J., Pluta, A., & Dasenbrock, J. (2021). *Gerneral information: Welcome to the scigrid_gas project webpage*. Retrieved 30.04.2021, from <https://gas.scigrid.de>
- Menon, E. S. (1978). *Pipeline planning and construction field manual*. Gulf Professional Publishing.
- Minaee, S., Boykov, Y., Porikli, F., Plaza, A., Kehtarnavaz, N., & Terzopoulos, D. (2020). *Image segmentation using deep learning: A survey*.
- NEL. (2013). *Letztes nel-teilstück kurz vor der fertigstellung*. Retrieved 03.12.2021, from [https://www.nel-gastransport.de/index.php?id=826&L=54&tx_news_pi1\[news\]=107&tx_news_pi1\[controller\]=News&tx_news_pi1\[action\]=detail&cHash=00e416f04a21720e4db4e89e0744e9a8](https://www.nel-gastransport.de/index.php?id=826&L=54&tx_news_pi1[news]=107&tx_news_pi1[controller]=News&tx_news_pi1[action]=detail&cHash=00e416f04a21720e4db4e89e0744e9a8)
- Nwankpa, C., Ijomah, W., Gachagan, A., & Marshall, S. (2018). *Activation functions: Comparison of trends in practice and research for deep learning*.
- Olaf Ronneberger. (2015). *Our u-net wins two challenges at isbi 2015*. Retrieved 08.07.2021, from <https://lmb.informatik.uni-freiburg.de/people/ronneber/isbi2015/>
- OpenStreetMap. (2020). *Main page — openstreetmap wiki*. Retrieved from https://wiki.openstreetmap.org/w/index.php?title=Main_Page&oldid=2013332 ([Online; accessed 25-Oktober-2021])
- Orhan Degermenci. (2019). *Design of pipeline systems - right of way*. Retrieved from <https://www.piping-world.com/design-of-pipeline-systems-right-of-way>
- Otsu, N. (1979). A threshold selection method from gray-level histograms. *IEEE Transactions on Systems, Man, and Cybernetics*, 9(1), 62-66. doi: 10.1109/

TSMC.1979.4310076

- Panaretos, V. M., & Zemel, Y. (2019). Statistical aspects of wasserstein distances. *Annual Review of Statistics and Its Application*, 6(1), 405-431. Retrieved from <https://doi.org/10.1146/annurev-statistics-030718-104938> doi: 10.1146/annurev-statistics-030718-104938
- Pluta, A., & Lünsdorf, O. (2020). Pb esy-osmfilter – a python library to efficiently extract openstreetmap data. *Journal of Open Research Software*, 8. doi: 10.5334/jors.317
- Ramdas, A., Trillos, N. G., & Cuturi, M. (2017). On wasserstein two-sample testing and related families of nonparametric tests. *Entropy*, 19(2). Retrieved from <https://www.mdpi.com/1099-4300/19/2/47> doi: 10.3390/e19020047
- Ronneberger, O., Fischer, P., & Brox, T. (2015). *U-net: Convolutional networks for biomedical image segmentation*.
- Rosebrock, A. (2016). *Intersection over union (iou) for object detection*. Retrieved 01.11.2021, from <https://www.pyimagesearch.com/2016/11/07/intersection-over-union-iou-for-object-detection/>
- Rosenblatt, F. (1958). The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6), 386.
- Sachs, L. (2004). *Angewandte statistik : Anwendung statistischer methoden* (Elfte, überarbeitete und aktualisierte Auflage ed.).
- Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., & Chen, L.-C. (2019). *Mobilenetv2: Inverted residuals and linear bottlenecks*.
- Settles, B. (2009). Active learning literature survey.
- Shen, L., Margolies, L. R., Rothstein, J. H., Fluder, E., McBride, R., & Sieh, W. (2019). Deep learning to improve breast cancer detection on screening mammography. *Scientific reports*, 9(1), 1–12.
- Simonyan, K., & Zisserman, A. (2015). *Very deep convolutional networks for large-scale image recognition*.
- Spektrum. (2017). *Lexikon der mathematik - perceptron*. Retrieved from <https://www.spektrum.de/lexikon/mathematik/perceptron/7763> ([Online; Stand 30. August 2021])
- Stanford CS231. (2021). *Pooling layer*. Retrieved 06.08.2021, from <https://cs231n.github.io/convolutional-networks/#pool>
- Sultana, F., Sufian, A., & Dutta, P. (2020, Aug). Evolution of image segmentation using deep convolutional neural network: A survey. *Knowledge-Based Systems*, 201-202, 106062. Retrieved from <http://dx.doi.org/10.1016/j.knosys.2020.106062> doi: 10.1016/j.knosys.2020.106062

- Tan, M., & Le, Q. V. (2020). *Efficientnet: Rethinking model scaling for convolutional neural networks*.
- The SciPy community. (2021a). *scipy.stats.kstest*. Retrieved 19.11.2021, from <https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.kstest.html>
- The SciPy community. (2021b). *Scipy stats wasserstein distance*. Retrieved 23.11.2021, from https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.wasserstein_distance.html
- Torch Contributors. (2019). *Crossentropyloss*. Retrieved 12.07.2021, from <https://pytorch.org/docs/stable/generated/torch.nn.CrossEntropyLoss.html>
- Trends, G. (2021). *Interesse im zeitlichen verlauf*. Retrieved 12.06.2021, from <https://trends.google.de/trends/explore?date=today%205-y&q=pytorch,tensorflow>
- USGS. (2021a). *Earthexplorer*. Retrieved 03.12.2021, from <https://earthexplorer.usgs.gov/>
- USGS. (2021b). *Landsat 7*. Retrieved 04.12.2021, from https://www.usgs.gov/core-science-systems/nli/landsat/landsat-7?qt-science_support_page_related_con=0#qt-science_support_page_related_con
- USGS. (2021c). *Landsat 8: Landsat 8 instruments*. Retrieved 03.12.2021, from https://www.usgs.gov/core-science-systems/nli/landsat/landsat-8?qt-science_support_page_related_con=0#qt-science_support_page_related_con
- USGS. (2021d). *Landsat collection 1*. Retrieved 04.12.2021, from https://www.usgs.gov/core-science-systems/nli/landsat/landsat-collection-1?qt-science_support_page_related_con=1#qt-science_support_page_related_con
- USGS. (2021e). *Pb landsat 5: Landsat 5 instruments*. Retrieved 03.12.2021, from https://www.usgs.gov/core-science-systems/nli/landsat/landsat-5?qt-science_support_page_related_con=0#qt-science_support_page_related_con
- Weier, J., & Herring, D. (2000). *Measuring vegetation (ndvi and evi)*. Retrieved 03.12.2021, from <https://earthobservatory.nasa.gov/features/MeasuringVegetation>
- Wikipedia. (2020). *Ki-winter — wikipedia, die freie enzyklopädie*. Retrieved from <https://de.wikipedia.org/w/index.php?title=KI-Winter&oldid=202118393> ([Online; Stand 27. Juli 2021])

- Yakubovskiy, P. (2019). *Segmentation models*. https://github.com/qubvel/segmentation_models. GitHub.
- Yakubovskiy, P. (2020). *Segmentation models pytorch*. https://github.com/qubvel/segmentation_models.pytorch. GitHub.
- Yamashita, R., Nishio, M., Do, R., & Togashi, K. (2018, 06). Convolutional neural networks: an overview and application in radiology. *Insights into Imaging*, 9. doi: 10.1007/s13244-018-0639-9
- Zech, M., & Ranalli, J. (2020, 06). Predicting pv areas on aerial images with deep learning.. doi: 10.1109/PVSC45281.2020.9300636
- Zhang, A., Lipton, Z. C., Li, M., & Smola, A. J. (2021). Dive into deep learning. *arXiv preprint arXiv:2106.11342*.
- Zhang, Z., Fidler, S., & Urtasun, R. (2016). *Instance-level segmentation for autonomous driving with deep densely connected mrfs*.


Danksagung

Ein besonderer Dank gilt meinen Betreuern Adam und Matthias, die mir stets mit Rat und Tag zu Seite standen. Sowohl die fachlichen als auch persönlichen Gespräche waren sehr wertvoll in den letzten Monaten. Darüber hinaus möchte ich auch den anderen Kolleginnen und Kollegen am DLR für die interessante und sehr kollegiale Zusammenarbeit danken.

Die wichtigste Person während des Studiums und vor allem in dieser Schlussphase der Masterarbeit ist Julia. Sie hat mir jederzeit den Rücken freigehalten und stand mir immer unterstützend und ermutigend zu Seite. Dafür danke ich ihr von ganzem Herzen.

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 Oldenburg festgelegt sind, befolgt habe.

A handwritten signature in black ink, appearing to read 'H.-P. Tetens' with a stylized flourish at the end.

Hendrik-Pieter Tetens

Matrikelnummer 5906917

Oldenburg, den 10. Dezember 2021