Cartography M.Sc.

Master thesis

Area Feature Reconstruction From Historical Topographic Maps Using Different Deep Learning Architectures

Sasanka Madawalagama

Technical University of Munich

TECHNISCHE UNIVERSITÄT WIEN
Vienna University of Technology

TECHNISCHE UNIVERSITÄT DRESDEN

UNIVERSITY OF TWENTE.

2022

TASK DESCRIPTION

Master Thesis Topic 2021

Area Feature Reconstruction from Historical Topographic Maps Using Different Deep Learning Architectures

Partner University: TU Dresden

Objective

The performance of Deep Learning architectures along with their specific requirements in a feature extraction process from scanned historic topographic maps shall be tested and be evaluated.

Key Words

Feature Recognition, Historic Maps, Digital Archives, Scans, Map Rendering, Deep Learning, DeepLearning Architecture.

Description

The thesis shall contribute to research in historic map semantics performed in a working group at Dresden University of Technology and Dresden University of Applied Science.

Presently in focus are scanned map series resulting from state-wide surveys around 1800. An overall target is a program development enabling a step-by step automated transformation of scans into collections of structured, tagged geo-data. The master student shall research feature recognition capabilities and performance in a comparison of different deep learning architectures that are freely available and adaptable to the given task with a moderate effort. Selected high-quality scans of various historic map series make up the input source. With the focus been set on the isolation of the best-performing architecture, the recognition task can be limited to an exemplary class of features: area features, as for instance buildings, quarries, ponds/lakes, etc. A clear look on the best architecture may not exclusively consider the highest detection quality but also the associated training requirements (volume, time) or the computation time, in brief the costs.

An existing in-house workflow based on a UNet architecture, may serve as an initial (reference) solution. Moreover, a further interesting aspect of research will be, how well our in-house software performs in the generation of training sets on the basis of machine generated synthetic map graphics, whereas the idea is to semi-automatically simulate a wide range of graphic fabrics that exist as a combination of the target feature and its topographic context in a series of maps. All tasks can be performed in close communication within the working group. Necessity and types of post-processing may additionally be discussed and be prepared for future work.

Application Scope

A start is given with selected map sheets of a Saxonian Topographic Survey (1780 – 1825). In case of successful reconstruction of features, tests might be extended to acquired scans of topographic surveys of a comparable age and scale class.

References

BUDIK,B., VAN DIJK,T.C.,WOLF,A.(2016). Matching Labels and Markers in Historical Maps. AnAlgorithm with Interactive Postprocessing. -Transactions on Spatial Algorithms and Systems(TSAS)2, 13:1-13:24.

LEYK, S., & BOESCH, R. (2010). Colors of the past: Color image segmentation in historical topographic maps based on homogeneity. GeoInformatica 14(1), pp. 1-21.

MENDT,J. (2014). Virtuelles Kartenforum 2.0 -Geodateninfrastruktur für die Raum-Zeit-Forschung-mit historischen Karten. BIS – Das Magazin der Bibliotheken in Sachsen 2014, Nr. 3, pp. 140-142.

Yao-Yi Chiang, S. L. (2014). A survey of digital map processing techniques. ACM Comput. Surv. 47,Article 1, 1.

ZHANG, Y., MA, Q., CHIANG, Y.-Y., KNOBLOCK, C., ZHANG, X., YANG, P. & HU, X. (2019). Extracting geographicfeatures from the Internet: A geographic information mining framework. Knowledge-Based Systems,174(15): 57–72. doi: 10.1016/j.knosys.2019.02.031

Supervision

Dr. Nikolas Prechtel (TUD)

Prof. Markus Wacker (HTW Dresden)

STATEMENT OF AUTHORSHIP

Herewith I declare that I am the sole author of the thesis named

„Area Feature Reconstruction from Historical Topographic Maps Using Different Deep Learning Architectures“

which has been submitted to the thesis assessment board today.

I have fully referenced the ideas and work of others, whether published or unpublished. Literal or analogous citations are clearly marked as such.

Dresden, 18/09/2022 Sasanka Madawalagama

Acknowledgement

First, I would like to express a sincere appreciation to the people who contributed to building this research. To Dr. Nikolas Prechtel for his continues encouragements and immense support, which motivated me to go one step further; to Prof. Dr. Markus Wacker for always guiding me towards the right way of doing a master thesis; and to Jens Friedrich for sharing his knowledge and always being there for help. A Special thank goes to Prof. Andrew Ng for creating an amazing online course on deep learning, which gave me the advanced knowledge required to complete this thesis.

I am incredibly thankful to my family, for always being there for me even through difficult times, and to my loving (soon-to-be) wife for her moral support and for not letting me procrastinate.

I am very grateful to be a part of this wonderful master's program that gave me so many opportunities and life memories. I am thankful to all my Carto friends who made these two years special and wonderful.

ABSTRACT

Historical maps contain a vast amount of valuable geographic information that can be used to analyse the past. However, automatically extracting information from historical maps is challenging due to several factors, such as inferior graphical quality and inconsistency of cartographic symbols. Deep learning technology has great potential to deal with such complex tasks because deep learning has already established its superiority in computer vision. In this study, the performance of different deep learning architectures was compared to extract area features from the historical topographic map series of Germany named "Milage Sheets of Saxony". Deep learning based semantic segmentation and object detection strategies were used to extract buildings, rivers, lakes and forest areas from map scans, and the performance was evaluated. In semantic segmentation, ResNet architecture showed the best performance in terms of accuracy and efficiency, achieving an IoU of 0.88 and a F1 score of 0.79. In object detection, YOLO v5 X architecture outperformed the rest by achieving a mAP of 0.92 and a F1 score of 0.89. A deep learning pipeline was developed to train, evaluate, and visualise the performance of the models. The major limitation of this study was the lack of training data. A simple yet effective software was developed to extract training data from map scans. This study demonstrates the unmatching capability of deep learning technology in performing complex feature extraction tasks in digital map processing.

CONTENTS

Table of Contents

FIGURES

TABLES

FORMULAS

LIST OF ABBREVIATIONS

Adam	Adaptive Moment Estimation
AI	Artificial Intelligence
CNN	Convolution Neural Networks
DBSCAN	Density-Based Spatial Clustering of Applications with Noise
DL	Deep Learning
GIS	Geographic Information Systems
GPU	Graphics Processing Unit
GUI	Graphical User Interface
IoU	Intersection over union
mAP	mean average precision
OBIA	Object-Based Image Analysis
SGD	Stochastic Gradient Descent

1 INTRODUCTION

1.1 Background

Historical maps store valuable information documenting human activities and natural features on earth over long periods. They are a significant source of information providing historical geography, topographic changes, settlements, political claims, and the evolution of cartographic methods. Historical topographic maps are often the most prominent information source about the earth as surveyed using past geodetic techniques. Such information is essential for studies that require detailed, long-term historical geographic data. Considering the importance of historical maps, many digital map archives have been established worldwide to preserve these valuable documents. The United States Geological Survey historical topographic map archive (Allord et al., 2014) and the Map Forum of the Saxon State Library (Kartenforum der Sächsischen Landesbibliothek) (Mendt, 2014) are great examples of such archives. The Map Forum of the Saxon State Library provides over 20,000 historical maps, including large-scale topographic maps of Germany, accessible to the public through an online portal (Mendt, 2014).

Moving beyond the preservation aspects of these precious historical maps, they provide information to analyse the historical geographic data to obtain insights into the history. In addition, the geographic information obtained by processing the historical maps can be used in spatiotemporal analysis in the fields of spatial, social, environmental, and health sciences.

Modern GIS systems facilitate the analysis of historical and geographical data by providing necessary analysing and visualising tools. However, the *challenge lies in extracting the data from the scanned map documents.* Digital conversion of map archives has widely progressed, but it got stuck in offering scans instead of structured geo-information. For quantitative analysis, it is essential to extract the information on the map scans in a structured format using digital map processing. The most straightforward way is manually digitising the map scans, which is highly time-consuming, inefficient and impractical to perform for an archive of historical maps due to the vast amount of available map sheets. A higher degree of automation for map processing is required, considering the millions of scanned maps stored in digital archives (Chiang et al., 2014).

Understanding maps and extracting the features are highly complex tasks. For us as humans, understanding maps is a cognitively demanding task that needs to be learned and practised to master. A similar learning process needs to be carried out for a machine or a computer to understand a map and extract its features. In general, it is called machine learning. In the pre deep learning era, researchers and scientists have used traditional machine learning methods to attempt cognitively demanding tasks with little success. For a machine to successfully perform such a complex task, it must learn. This is where deep learning becomes significant.

Deep learning is a subset of machine learning inspired by a human brain structure to develop systems that learn similarly to how humans learn (Glassner, 2021). Deep learning algorithms perform a task repeatedly, making adjustments each time to improve the results, much like how humans learn from experience. Because neural networks have several “deep” layers that facilitate learning, this process is referred to as "deep learning". Deep Learning has significantly impacted many scientific fields in recent years (Goodfellow et al., 2016). For example, it can train computer programs that outperform human players in games like Go, automatically drive a car on a busy road or win a fine art competition by artificially creating a painting. Before the invention of deep learning, many of these problems were regarded as being beyond the capabilities of computers, even in science fiction literature.

1.2 Motivation and Problem Statement

Digital map processing can be defined as computational methods to extract and recognise geographic features from scanned images of maps. Researchers from various disciplines, such as computer science, cartography, and geography, have been working on inventing and improving digital map processing techniques since the early 1980s (Chiang et al., 2014). It is an emerging research field that grew out of image processing, computer vision, document analysis, and cartography. Multiple studies have adopted different digital image processing techniques to automate map processing. The major limitation of the methods is scalability (Chiang & Knoblock, 2013; Drolias & Tziokas, 2020). *These systems do not scale well for processing large numbers and varieties of historical maps.* An overview and limitations of digital map processing techniques are provided in chapter2.

Printed maps are considered complex documents visualising geospatial data that refer to the location and attribute of the objects on earth or a phenomenon. Maps utilise different visual variables to convey the information to their reader. In a broader sense, topographic maps use colour to convey thematic information, such as land cover types and symbols used to describe different geographic features. Furthermore, maps utilise many visual variables such as colour, texture, size and orientation for composing different symbols. Due to this complexity, it is more difficult to machine extract information from a map than from other written or printed documents (Groom et al., 2020). In addition, maps have overlapped symbology, increasing the degree of complexity in map processing. A challenging aspect of digital map processing is to spatially isolate these overlapping layers of geographic information from each other and to filter out those map elements that obstruct the extraction of the geographical layer of interest.

Historical maps raise unique challenges in digital map processing. General digital image processing techniques cannot be directly applied to historical maps due to the complexity induced by multiple factors. The graphical quality of the scanned historical maps is often adversely affected by ageing, which results in bleaching effects (Chiang et al., 2014). As these maps are produced manually, there are significant variations of cartographic symbols in colour, size, shape, orientation and placement (Groom et al., 2020).

Considering the complexity of digital map processing for historical maps, a different approach is required to unlock the embedded geospatial data efficiently. Recent studies prove that deep learning provides a way forward in digital map processing due to its capability to handle large diversified input data (Chiang et al., 2020b; Ekim et al., 2021; Uhl et al., 2020). *However, deep learning in digital map processing is still a young research field with challenges that must be addressed.* This research is aimed to narrow the gap between the fields of deep learning and digital map processing.

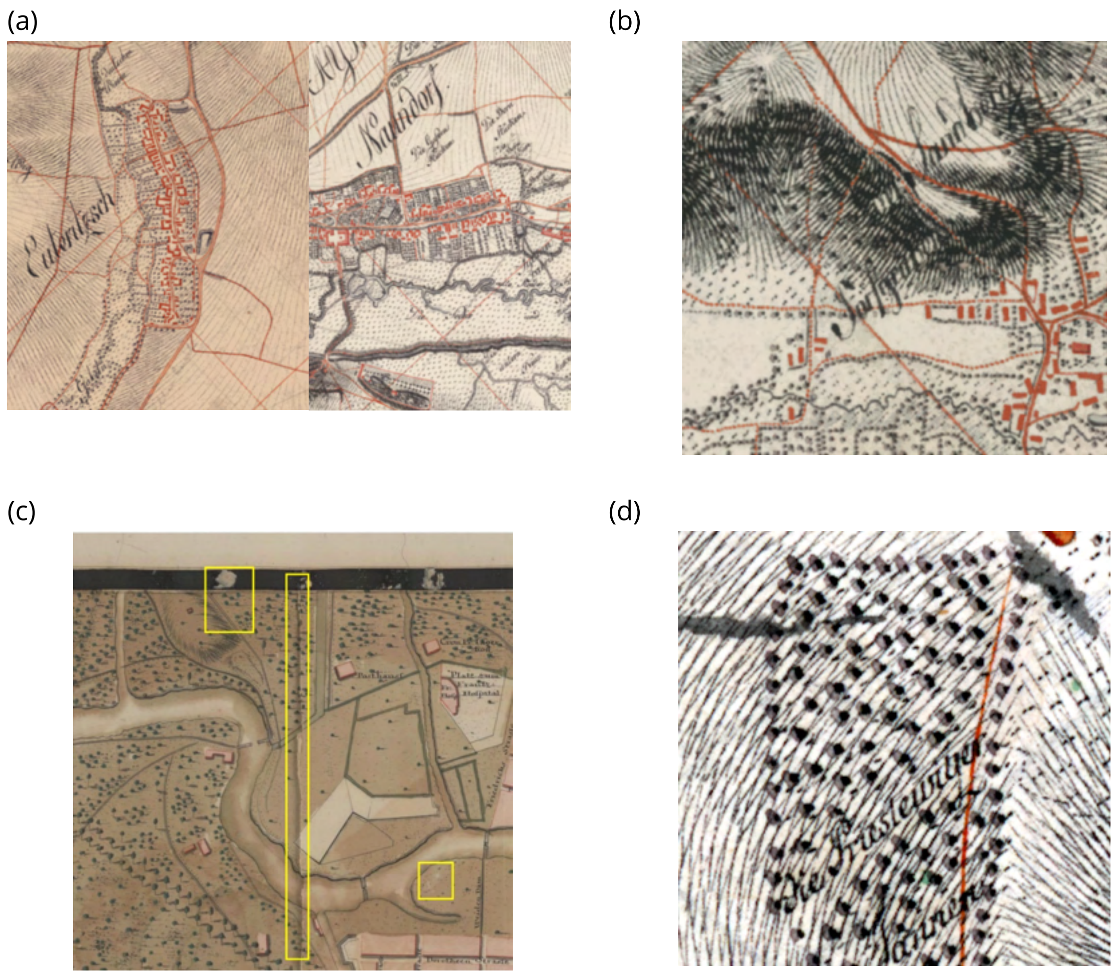


Figure 2-1 : Challenges in digital map processing on historical maps (a) Different graphical qualities (b) Overlapping symbols (c) Effects of ageing (d) Variation of cartographic symbols (Image Source: Map Forum of the Saxon State Library).

1.3 Knowledge Gap and Contribution

Compared to other machine learning methods, deep learning has the unique ability to perform computer vision tasks such as object recognition and classification with minimum user interaction by looking at the data and learning complex algorithms that can never be explicitly programmed by a human. Specific to feature extraction in digital map processing, deep learning methods can capture the underlying semantics of map features that are difficult to be captured by conventional feature extractors such as SIFT (Scale Invariant Feature Transformation) or SURF (Speeded Up Robust Features)(Kang, 2020). Also, it is proven that deep learning models can handle the complexity of maps in digital map processing. Despite all the advantages provided by deep learning and the availability of computational power, deep learning is not widely used to process archived maps on a large scale.

Deep learning is still a new research field in cartographic domains, such as digital map processing. The full potential of deep learning in digital map processing has not been realised since the users of scanned historical maps and computer scientists who invent deep learning algorithms come from different backgrounds and frequently operate in silos (Chiang et al., 2020b). Moreover, deep learning and cartography are separate paradigms. There is no established framework to merge the two sciences to work in harmony in the domains such as digital map processing.

Deep neural networks are composed of a vast number of parameters, and they need massive training datasets to achieve an optimal performance in deep learning. Therefore, the availability of sufficient training data is key to the successful application of deep learning methods. A significant reason for the current advancements in deep learning for computer vision tasks such as object detection is the availability of publicly available training datasets. For example, MS-COCO, a rich object detection and segmentation dataset, contains 330,000 images with 80 object classes (Lin et al., 2014). Such datasets facilitate the training of deep learning models by providing training data with good quality and sufficient quantity. Developers of cutting-edge deep learning applications take advantage of such datasets to build and fine-tune their models. However, there is no publicly available machine learning dataset in the context of digital map processing. Compared to the maps, which are documents with cartographic features denoted by symbols, all the publicly available machine learning datasets contain daily images. It is a significant setback to develop and train deep learning models to extract features from historical maps. In order to develop deep learning models for digital map processing tasks, developing a training dataset from ground zero is also required, which is a tedious and costly task. For the field of map processing, it is frequently impractical to obtain the resources needed to make a vast amount of training data for training deep learning models that work seamlessly with maps in different cartographic styles and scales (Chiang et al., 2020a).

1.4 Research Objectives

The primary objective of this study is to **evaluate different deep learning architectures for digital map processing focusing on areal feature reconstruction from historical topographic maps**.

The focus of the study has been set on the isolation of the best-performing architecture. However, the scope of the research will be limited to an exemplary class of cartographic features: area features, i.e., buildings, water features and forests.

The following sub-objectives further structure the primary objective of the research.

1) Identify strengths, weaknesses and challenges of different deep learning architectures for feature recognition from historical map scans.
2) Performance evaluation of selected deep learning architectures and their specific requirements in areal feature extraction from scanned historical topographic maps.

1.5 Research Questions

1.5.1 [RQ1] What deep learning architectures are to be used in evaluation?

In computer science, a large variety of deep learning models exist, each addressing a specific set of problems. A convolution Neural Network or CNN is a type of deep learning based artificial neural network developed explicitly for image processing and computer vision tasks (Goodfellow et al., 2016). CNNs have proven so effective for image analysis tasks that they are the go-to method for prediction problems involving image data as an input. The class of neural networks used in this study would be CNNs. However, there are many cutting-edge CNN architectures such as UNet, LeNet, AlexNet, Inception, Xception and many more.

It is impossible to implement and compare all the CNN architectures within this research scope. The deep learning architecture to be evaluated will be selected based on an extensive literature review and study of the CNN models in similar applications.

1.5.2 [RQ2] How to create training datasets?

In order to extract features from scanned maps using deep learning, the deep learning models are required to be trained with a labelled training dataset. However, most labelled training data from public datasets are daily images containing real-world objects, while maps are document images with geographic features represented by cartographic symbols. Due to this reason, it is not possible to use publicly available machine learning datasets for training the models. Therefore, a training dataset is created specifically for this research based on 1:20000 scale maps of the Saxonian Topographic Survey from 1780 to 1825. In addition, necessary tools are developed to make the task easy and efficient.

1.5.3 [RQ3] How to adapt existing deep learning architectures for feature extraction in digital map processing?

The scope of this study will be limited to the extraction of areal features. Therefore, a deep learning pipeline is developed for area feature extraction from historical maps incorporating training and evaluation of the model and visualising the results. In addition, selected deep learning architectures are included in the training phase of the pipeline so the user can select the architecture based on the requirement.

1.5.4 [RQ4] How can the models be evaluated?

Evaluation matrices of the selected deep learning models are selected to measure the segmentation/detection quality and computational time

2 THEORETICAL BACKGROUND

2.1 AI, Machine Learning and Deep Learning

Artificial Intelligence, Machine Learning, and Deep Learning have become hugely influential technologies in computer science and current industries, as these technologies push the boundaries of computational problem solving (Goodfellow et al., 2016). The applications of these technologies can range from playing a chess game to state-of-the-art self-driving cars or a computer program which can turn a few sentences into a beautiful painting. Even though these technologies have nuance, these terms are often used interchangeably, which can lead to confusion. An easy way to understand the difference between Artificial Intelligence, Machine Learning and Deep Learning is with a Venn diagram (figure 2-1). Deep Learning is a subset of Machine Learning, a subset of Artificial Intelligence. Machine learning and deep learning are two specific subfields of Artificial intelligence.

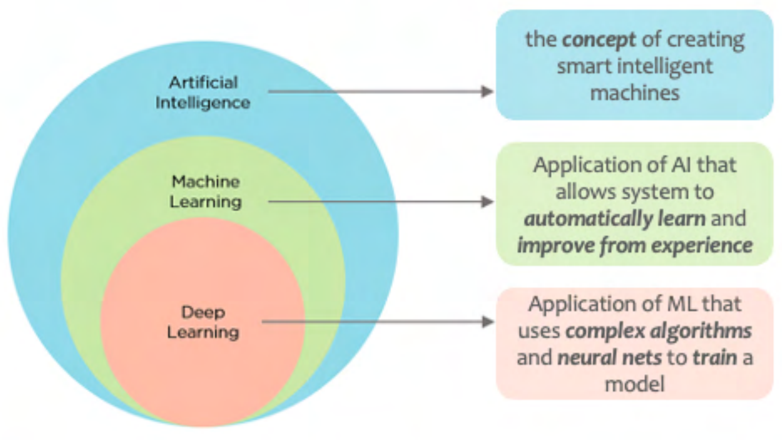

Artificial Intelligence
Machine Learning
Deep Learning
the **concept** of creating smart intelligent machines
Application of AI that allows system to ***automatically learn*** and ***improve from experience***
Application of ML that uses ***complex algorithms*** and ***neural nets*** to ***train*** a model

Figure 2-1 : Venn diagram explaining artificial intelligence, machine learning, and deep learning.

In general, Artificial Intelligence (AI) refers to a computation that mimics the cognitive capabilities of a human brain. Even though current AI models are far from the cognitive ability of the human brain (Glassner, 2021), AI conceptualises the creation of smart, intelligent machines. The primary objective of Artificial Intelligence is to create self-sufficient computers capable of thinking and acting like humans. These machines can learn and solve problems to replicate human behaviour and complete tasks. More fundamentally, an AI model can simply be a programmed rule (set of If-else statements) that instructs the machine to respond in a particular manner under specific scenarios.

Machine learning is a discipline in computer science that falls under artificial intelligence that use computer algorithms and analytics to create prediction models. The Machine Learning book authored by Mitchell, 1997 defines machine learning algorithms as "A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E." Machine learning can utilise a large amount of both structured and unstructured data and learn from data using various algorithms and techniques to make predictions. Machine learning algorithms can be classified into three main categories (Alzubaidi et al., 2021). Supervised learning algorithms use labelled data which means the outcome or the target variable is known, so the system predicts the future based on the past data. Linear regression, support vector machines and decision tree-based classification are some examples of supervised learning. Classical supervised machine learning requires more human assistance. On the other hand, Unsupervised learning uses unlabelled data to find patterns from data. Finally, in Reinforcement learning, an agent is trained to perform a task in an uncertain environment where the agent is rewarded by the environment based on how successful the agent is in completing the task.

2.2 Deep Learning

Deep learning is a subset of machine learning representing a network of compute units with three or more layers (Glassner, 2021). These compute units are often called neurons; hence the network of compute units is called an artificial neural network.

In the machine learning field, deep learning has recently been recognised as the gold standard (Chiang et al., 2020a). Additionally, it has steadily grown to be the most popular computational strategy in machine learning, producing exceptional outcomes on several challenging cognitive tasks that are on par with or above human performance. The capacity to learn from enormous volumes of data and adapt to complex scenarios are the main advantages of deep learning. Deep learning is also referred to as scalable machine learning (Goodfellow et al., 2016) because of the ability to use massive datasets and improve as the dataset grows. Furthermore, deep learning accomplishes "end-to-end learning", in which a network is given raw data and a job to fulfil, such as classification, and automatically learns how to do so.

Deep learning differs from non-deep or classical machine learning by the type of input data it works with and the learning methods (Sejnowski, 2020). Also, deep learning reduces most of the data preprocessing required by classical machine learning. These algorithms can ingest and analyse unstructured data such as text and pictures and automate feature extraction, reducing reliance on human inputs. For example, in the context of digital map processing, If it is required to categorise buildings, trees and lakes from a scanned image of a map, a deep learning algorithm can decide which characteristics (colour, shape) are most essential in distinguishing one symbol from another. In contrast, a human expert in classical machine learning manually establishes this feature hierarchy. A limitation of deep learning is that it requires a massive amount of processing power (Goodfellow et al., 2016). High-performance graphics processing units (GPUs) are suitable since they can manage a massive volume of operations over numerous cores with enough memory. But acquiring and managing such a demanding hardware environment is highly expensive.

A deep neural network is constructed with an input layer that accepts data inputs, multiple hidden layers and an output layer that provides expected outputs. In a fully connected deep neural network, such as the figure 2-2, a single neuron in a layer takes the previous layer's outputs as inputs (except for the first input layer).

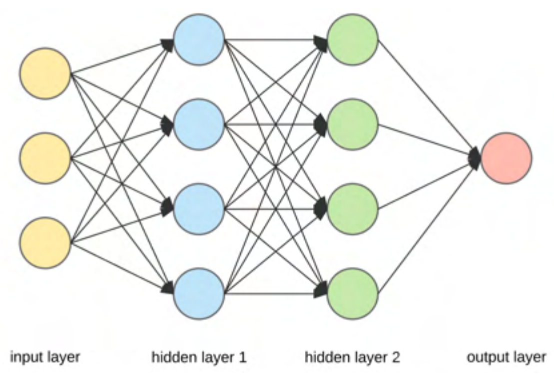


Figure 2-2 : An Illustration of a deep neural network (Dertat, 2017).

An intuitive way to understand the underlying principles of deep learning is by understanding a single neuron of a neural network. The computation of a single neuron is followed by these steps.

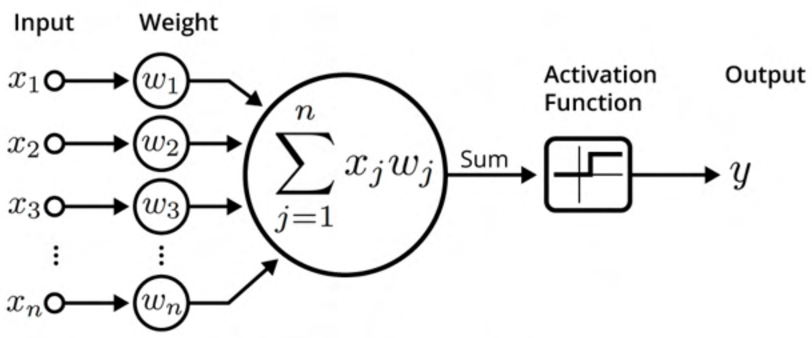


Figure 2-3 : An illustration of a neuron in a Neural Network (Dertat, 2017).

1. The input of the neuron is taken as the vector $X = (x_1, x_2, x_3, \dots \dots, x_n)$ and the corresponding weights to each input are taken as $w = (w_1, w_2, w_3, \dots \dots, w_n)$. At the initial stage, the weights are assigned randomly.
2. The weighted sum of the inputs is calculated $Z = \sum_{j=1}^{n} x_j w_j$
3. The calculated weighted sum is passed through an activation function A. The activation function decides whether the neuron should be fired or not. The activation function takes the weighted sum Z and a bias x_0 as the inputs and gives the output Y. In a multi-layer, multi-neuron neural network, this output Y denotes the significance of the particular neuron to the final output. In a single neuron neural network, output Y represents the predicted output.

   The ability of an activation function to bring non-linearity to a neural network is its most essential attribute in training a neural network. Due to the non-linearity introduced by the activation function, neural networks can learn complex non-linear patterns from the data (Glassner, 2021). Therefore, a neural network is just a linear regression model without an activation function.

   The following are commonly used activation functions in deep learning and artificial neural networks with reference to the study done by Nwankpa et al., 2018 comparing differing activation functions.

Sigmoid Function

Sigmoid activation function, also called logistic function, is generally used in binary classification. Sigmoid transforms the input value providing outputs in the range of 0 and 1. However, the sigmoid function is only used in shallow neural networks because it hinders performance when training a deep neural network. This is due to the vanishing gradient problem and not being zero-centred caused by the nature of the sigmoid function.

$$\sigma(x) = \frac{1}{1+e^{-x}}$$

Equation 1: Sigmoid function

Hyperbolic Tangent Function (Tanh)

The hyperbolic Tangent Function or the Tanh function is zero centred function which has a range of -1 and 1. Compared to the sigmoid function, the Tanh function gives better training performance because it is zero-centred but does not solve the vanishing gradient problem. Therefore, it is possible to use the Tanh function in multi-layer neural networks and is mainly used in recurrent neural networks for natural language processing and speech recognition tasks.

$$f(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$$

Equation 2: Hyperbolic Tangent Function

Rectified Linear Unit Function (ReLU)

Rectified Linear Unit or ReLU activation function is another non-linear, computationally simple activation function that has gained prominence in deep learning. The key benefit of the ReLU function over sigmoid and Tanh activation functions is that it does not activate all neurons simultaneously and yields faster performance. Since it does not compute exponents or divisions, the computation of the neural network is highly enhanced. ReLU was introduced in 2010 and has been the most widely used activation function in deep learning due to its efficiency and effectiveness (Krizhevsky et al., 2017). ReLU and its variants are used in many deep learning architectures, especially in convoluted neural networks.

$$f(x) = \max(0, x) = \begin{cases} x, & if\ x \geq 0 \\ 0, & if\ x < 0 \end{cases}$$

Equation 3: Rectified Linear Unit Function

Sigmoid, Tanh and ReLU are the activation functions used in this study for the selected predefined deep learning architectures. However, there are many more activation functions being used in the domain of deep learning.

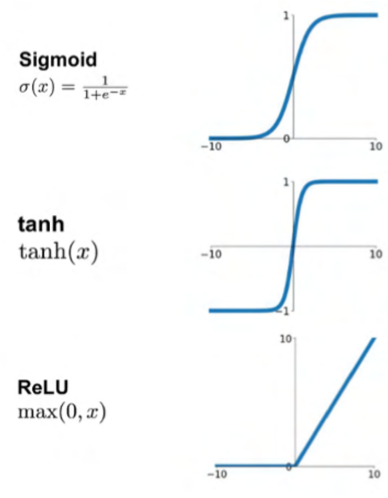


Figure 2-4 : Graphs of different activation functions.

4. The predicted output Y is compared with the actual output Y'(ground truth) using a loss function J, and the loss is calculated $Loss = J(Y, Y')$. The loss function measures how well the neural network performed in relation to a given training sample and predicted output. Total loss is a measure of the total error.
5. Training of the neural network is performed to find the optimal weights that minimise the loss. Then, the weights are updated using a method called backpropagation by minimising the loss. The significance of backpropagation in neural networks was first recognised after the study by Rumelhart et al., 1986. The research discusses various neural networks in which backpropagation works far quicker than prior ways of learning, allowing neural nets to handle previously intractable tasks. Today, the backpropagation algorithm is the backbone of neural network learning. The partial derivative $\frac{\partial J}{\partial w}$ of the loss function J with respect to any weight w_j in the network is at the base of backpropagation. The partial derivative describes how rapidly the loss changes when the weights are changed. With the help of $\frac{\partial J}{\partial w}$, it is possible to adjust the weights of the neural network and learn the best possible weights that minimise the loss.
   Training is an iterative process, so multiple iterations over the same input dataset will be used to find the optimal weight values. Training is stopped when the loss reaches or gets significantly closer to its minimum.

2.3 Convolution Neural Networks (CNNs)

Convolution Neural Network or CNN is a well-known architecture of deep learning techniques commonly employed in image processing applications. The visual system of the brain served as inspiration for CNN's architecture.

Similar to a conventional neural network, a CNN has an input layer, one or more hidden layers, and an output layer. At the start of the model training phase, each neuron is given a weight adjusted to minimise the loss function with backpropagation. However, there are several architectural differences in CNNs compared to conventional neural networks. CNN comprises three unique types of layers (Albawi et al., 2017): convolutional, pooling, and fully connected. Due to the availability of these special layers, CNNs scale well to analyse images where regular neural networks do not. For example, if we take a RGB image of 16px by 16px, the input dimension of a traditional neural network would be 16x16x3 = 768 input units which is acceptable. But when the image size increases to 512px by 512px, the input size of the network would increase to 786,432, so the number of parameters of the network and the complexity increases exponentially. With convolution neural networks, it is possible to scale for image data without exponentially increasing the complexity of the neural network.

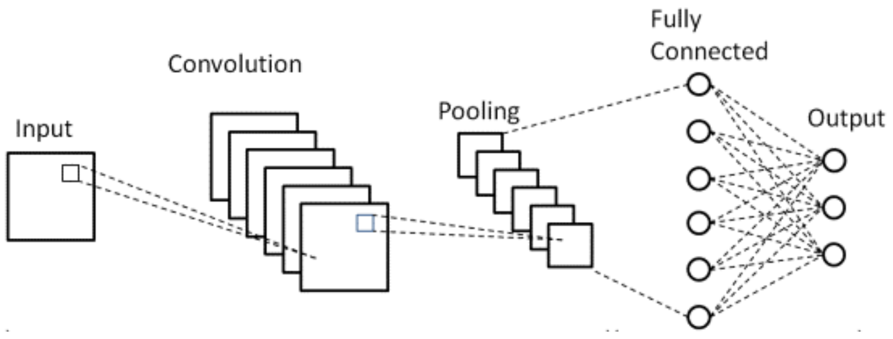


Figure 2-5 : Illustration of a CNN architecture (O'Shea & Nash, 2015).

Convolutional layers serve as CNN's input layers, while fully connected layers serve as output layers. A series of convolutional and pooling layers make up the hidden layers of the neural network. CNN uses the first hidden layers to find basic visual patterns like edges, colours, etc. However, when it goes deeper into the network, the complexity of the extracted features rises, and filters are used to isolate more prominent elements (O'Shea & Nash, 2015), such as facial features or symbols, depending on the application domain. Finally, features are recognised, and class predictions are given using the combined output of all the layers.

2.3.1 Convolution Layer

Convolutions are often used in digital image processing for various purposes, including sharpening, enhancing, and intensifying. Convolutions are used in CNN to extract the key features from the image, making convolution the CNN's building block (Alzubaidi et al., 2021). Each convolution layer comprises learnable filters known as kernels which are basically two-dimensional arrays. These kernels are moved over the input images, generating output feature maps. This step is known as convolution. Since the weights of the kernels are learnable through the training phase of the neural network, the kernels are automatically adjusted to extract the required features.

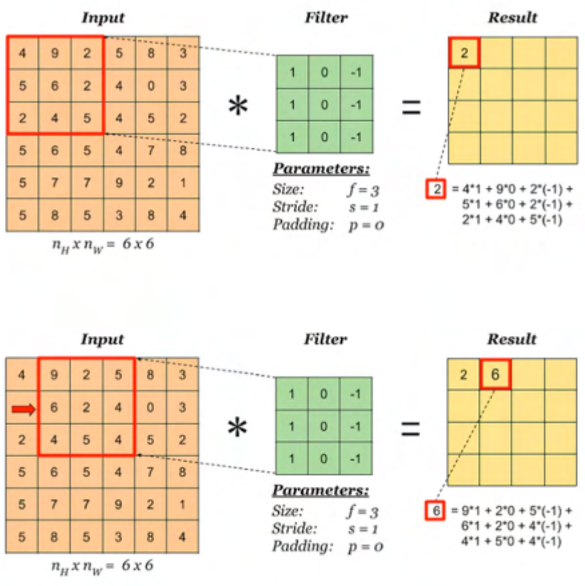


Figure 2-6 : Example of Convolution Operation (*Introduction to Convolutional Neural Networks Architecture*, n.d.).

2.3.2 Pooling Layer

In a single convolution layer of CNN, multiple convolution filters generate multiple feature maps resulting in a higher dimensional stack. The task of the pooling layer is to reduce the dimension of the feature maps by aggregating and decreasing the dimensions of the feature maps (Alzubaidi et al., 2021). Two types of pooling are used in CNNS, max pooling and average pooling. In max pooling, the input pixel with maximum value is taken, whereas, in average pooling, the average value of the input pixels is taken. Pooling is done to reduce the complexity of feature maps, improving CNN's efficiency. However, pooling operation results loss of some information but highly decreases the computational power required to process large datasets.

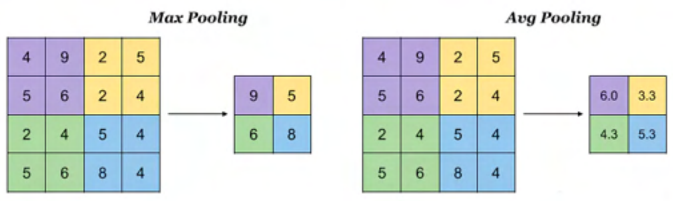


Figure 2-7 : Max Pooling and Average Pooling (*Introduction to Convolutional Neural Networks Architecture*, n.d.).

2.3.3 Fully Connected Layer

The tasks of convolution and pooling layers are to extract features and reduce the dimensions of feature maps where the features are not yet classified. The task of the fully connected layer is to perform the classification process. This layer is often found near the end of each CNN architecture and serves as the CNN’s classifier (Alzubaidi et al., 2021). Flattened output from the convolution layers and pooling layers are fed to the fully connected layer where all the input units are connected to all the activation units of the consecutive layer (thus the name "full connected"). The final fully connected layer gives the probabilities for each label, hence finalising the classification.

2.4 Semantic Segmentation and Object Detection

This thesis is focused on applying deep learning based computer vision methods to extract features from map scans. Object detection and semantic segmentation are the computer vision techniques employed in the study. In object detection, the CNN algorithm finds the objects within the images and their location using bounding boxes (Alzubaidi et al., 2021). Multiple objects in a single image were identified and defined with a bounding box consisting of four parameters, the centre X, Y coordinates, width and height. So, in object detection, no information about the object's shape is given as the output.

In contrast, semantic segmentation classifies each pixel of an image to a particular class outputting a pixel-wise mask. Semantic segmentation methods are used in applications where a more granular understanding of the boundary and shape of the object is necessary.

Object Detection

- **Input** is a RGB (3 channel) or greyscale (single channel) image matrix
- **Output** is a collection of bounding boxes defined by centre x, y coordinates, width and height

Semantic Segmentation

- **Input** is a RGB (3 channel) or greyscale (single channel) image matrix
- **Output** is an image mask with the same width and height as the input image, where the class label is assigned to each pixel

Both object detection and semantic segmentation are complicated tasks, but deep learning-based computer vision systems have succeeded in both. However, there is a substantial difference in performance between the two procedures. Semantic segmentation is more challenging (Chiang et al., 2020b) than object detection because semantic segmentation provides a more in-depth comprehension of the image, whereas object detection provides abstract output. Furthermore, generating a training dataset for semantic segmentation is a complex and time-consuming operation because it requires the creation of accurate borders. With the understanding of the requirements and limitations, this study utilises both object detection methods and semantic segmentation.

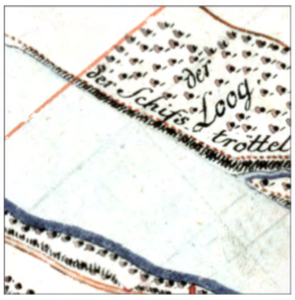
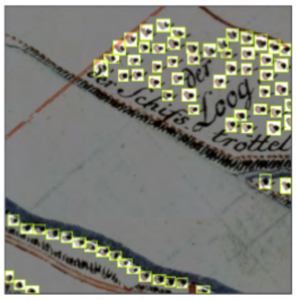
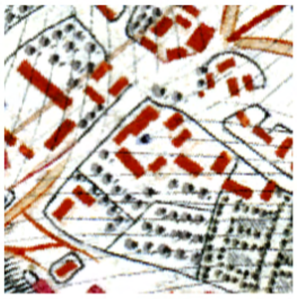
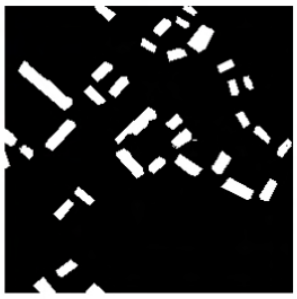
	Input	Output
Object Detection		
Semantic Segmentation		

Figure 2-8 : Comparison of input and output of object detection and semantic segmentation.

3 LITERATURE REVIEW

3.1 Early Days of Digital Map Processing Techniques for Feature Extraction

Since the establishment of digital map archives to preserve historical maps, researchers from various domains have established computational methods to extract geographical features from map scans. Therefore, it was necessary to develop automated or semi-automated strategies to extract the information from historical maps considering the vast amount of map documents scanned and stored in map archives (Chiang et al., 2014). Digital Map processing for information extraction from maps has been developed since the 1980s (Ebi et al., 1994) and has grown in intensity with the improvement of scanning technology, storage capacities, and computational power (Chiang et al., 2014).

In the early days of digital map processing, colour image segmentation was a technique adapted from the image analysis domain in computer science to extract features from map scans. Image segmentation divides images into homogeneous regions or objects (Khattab et al., 2014). With image segmentation, an image is split into more representative sections and is easier to study. In colour image segmentation, it is assumed that homogeneous colours of image pixels corresponding to homogenous segments, representing meaningful objects of the image. Colour image segmentation is an obvious choice of method in digital map processing because the same colour combination represents the same thematic information of the maps. An example of such representation in topographic maps can be taken from USGS topographic product standard written by Allord et al., 2014. According to the USGS standard, green is used for areas of vegetation, blue for water features and brown to represent topographic features. This standard can vary slightly from one topographic map series to the other. But since colour remains the most prominent attribute in describing area features of a topographic map, colour image segmentation can be considered a simple automated way to map processing (Chiang et al., 2014). Histogram-based thresholding (Glasbey, 1993) and colour space clustering (Khotanzad & Zink, 1996) are commonly used colour image segmentation techniques used in digital map processing.

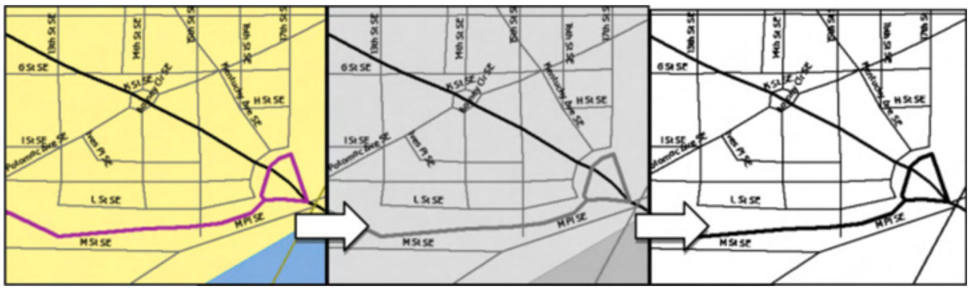


Figure 3-1 : Example of histogram thresholding technique in colour image segmentation to extract roads from a map scan [Left: Original RGB Map Scan; Middle: Conversion to monochrome image; Right: Extracted roads] (Chiang et al., 2014).

However, in practice, applying colour image segmentation methods is highly inefficient on historical maps. It is identified with the study by Chiang et al., 2014 that the colour of historical maps varies drastically mainly due to the manual process of creating the maps and ageing. Colour image segmentation requires homogenous colours in scanned maps belonging to the same class based on the colour characteristic of image pixels. The homogeneous zones that are classifiable must also be spatially contiguous. As a result, colour image segmentation methods fail to extract composite cartographic symbols that combine point, line and area symbols to denote a single class (ex: forest, wetland). Furthermore, as explained in work done by Jiao et al., 2020, colour image segmentation methods are unable to extract marsh regions (figure 3-2) where the area is symbolised by strokes which makes the symbol not area-filling in colour while forming a continuous texture.

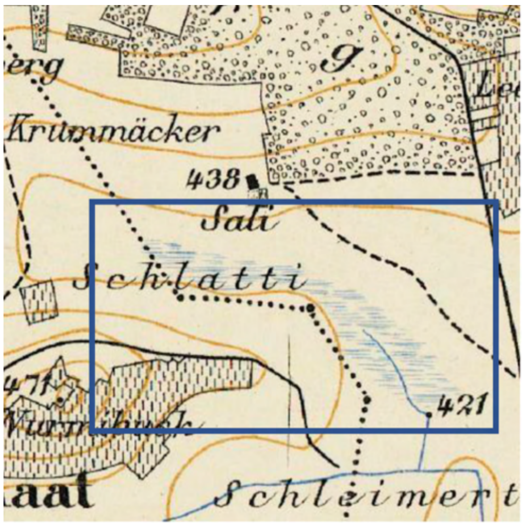


Figure 3-2 : Non-continues cartographic symbol of marsh regions from a Swisstopo map (Jiao et al., 2020a).

Edge detection is another technique used in the early days of digital map processing to extract line symbols like roads, contour lines, railways etc., from map scans. Edge detection is used in image processing to determine the borders of objects inside images. It operates by detecting changes in brightness (Oskoei & Hu, 2010) which usually works by applying filters such as Sobel, Canny, and Prewitt (Basu, 2002) applied to the converted greyscale image of a map. However, as explained in the study by Chiang et al., 2014, map scans with low contrast or unconnected features have limitations in using the above-mentioned edge detection techniques to extract features. Also, noise is induced in the results due to the small kernel size of the applied local differential filters.

Another approach often employed in computer vision applications that paved the way to digital map processing is template matching. The book by Brunelli, 2009 extensively analyses template matching techniques and their applications. Traditional template matching algorithms compare raw pixel values inside a searching window to identify a match between a template and a target image. The objective of template matching is to find similar objects in the template inside the target image. When it comes to digital map processing, template matching is successfully used to extract point symbols from map scans when the symbol is consistent in shape and size throughout the map (Xia et al., 2022). However, classic template matching techniques cannot handle complicated scenarios effectively when the transformation between the template and the target picture is non-rigid or involves overlaps, which is common in maps (Chiang et al., 2014).

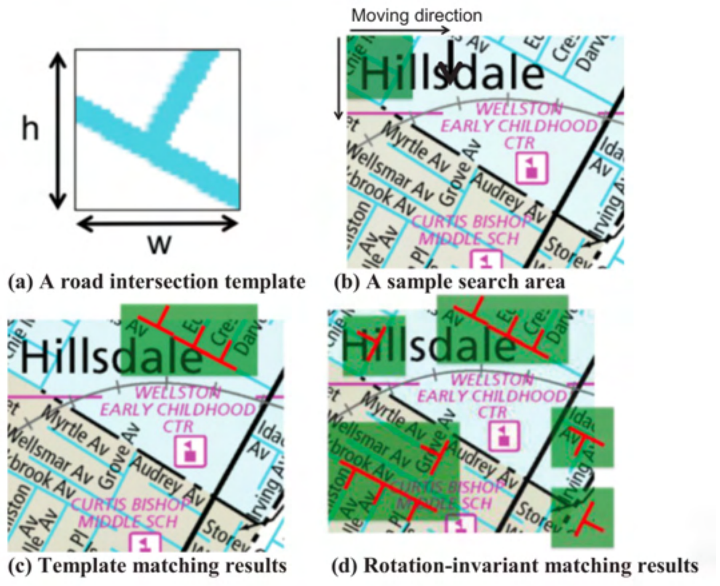

Moving direction
h
w
(a) A road intersection template
(b) A sample search area
(c) Template matching results
(d) Rotation-invariant matching results

Figure 3-3 : Use of template matching to extract road intersections (Chiang et al., 2014).

3.2 Influence of GIS in Digital Map Processing

With the establishment of Geographic Information Systems (GIS), the interest in extracting data from map scans has increased drastically (Drolias & Tziokas, 2020) as GIS is a versatile tool to visualise, analyse, and store the data. GIS provides a centralised system to georeference, digitise, classify and post-process the extracted data. The most straightforward strategy to use GIS tools for digital map processing is to extract the features manually by one or more users who digitise (redraw the map features in vector format) and assign one or more labels to each feature. Manually digitising the content of a historical map is a complex, time-consuming and costly task. An example given in the book by Budig, 2018 shows that it takes an expert 15 to 30 hours to extract and georeference building footprints of a single map. Another study by Chiang et al., 2014 provides an example of labelling a nautical chart just by drawing bounding boxes would take around 6 hours to complete. Modern GIS applications also offer crowdsourcing capability, where experts can work collaboratively on a single task. However, considering the factors of complexity and time, a manual digitisation approach with GIS tools would be acceptable only when processing a small number of map sheets. It will be practically impossible to manually digitise the vast amount of historical maps available in today's archives.

Modern GIS solutions facilitate tools for manual digitisation. Also, they are equipped with versatile image processing tools which can be utilised in digital map processing. The work done by Gobbi et al., 2019 showcases the use of Object-Based Image Analysis (OBIA) tools in the GRASS GIS package to extract features from historical maps. The authors also utilise GIS tools to preprocess the map scans, which remove unwanted texts, symbols and boundary lines. The procedure was tested with three different historical maps with different characteristics (one cadastral map and two forest maps) and reached an average classification accuracy of 97%. The object-based image analysis technique combines region-based segmentation with rule-based machine learning. OBIA differs from traditional pixel-based classification because it first breaks the image into objects (group of pixels), followed by rule-based classification that utilises the geometric and colour information of the objects. The classification ruleset needs to be defined by an expert, and it is subjective to input data. The study by Gobbi et al., 2019 shows the potential of object-based image analysis applied in digital map processing, obtaining a high accuracy when map symbolisation is simple, but this is also because of the manual definition of the classification rule set by an expert for each map sheet.

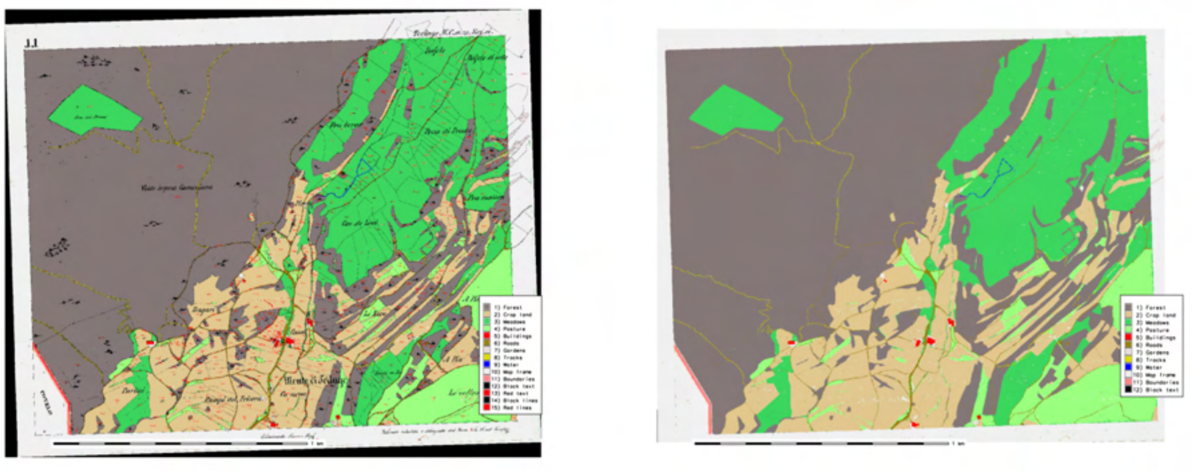


Figure 3-4 : Application of object-based image analysis on cadastral map sheet [left: input map sheet; right: output] (Gobbi et al., 2019).

The power of GIS in digital map processing lies not only in feature extraction but also in establishing an end-to-end pipeline. The study by Drolias & Tziokas, 2020 has established such pipeline using open source GIS tools (QGIS, GRASS, and SAGA) to extract building footprints. The overview of the pipeline is shown in the figure 3-5. First, the scanned map is manually georeferenced using control points. The feature extraction was done next by applying a manually defined ruleset to perform a binary classification by thresholding. This method extracts a single feature class, in this case, buildings. The extracted budlings are still in raster format and are then converted to vector using the raster to vector conversation tool available in GIS software. The main advantage of such a pipeline is that the whole process can be programmed using GIS tools and executed automatically only with a few inputs from the user. The main disadvantage is that the input parameters to this pipeline change from one map scan to another, in particular in the stage of feature extraction.

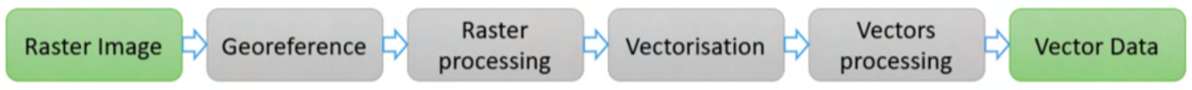


Figure 3-5 : Overview of a GIS-based pipeline for Digital Map Processing (Drolias & Tziokas, 2020)

3.3 Deep Learning Based Feature Extraction in Digital Map Processing

In recent years, deep learning based supervised learning algorithms have demonstrated their supremacy in automated feature extraction from historical maps. Several studies have proven that deep learning is the way forward to digital map processing outperforming the traditional and GIS-based approaches. Recent research has moved away from laboriously creating tailor-made machine learning rulesets and toward the use of deep learning (DL) techniques to extract features from historical maps (Chiang et al., 2014; Drolias & Tziokas, 2020; Ignjatić et al., 2018; Jiao et al., 2020b; Uhl et al., 2020). It has been identified that convolution neural networks are versatile and the most suitable technique for feature extraction by today's standard. The advantages of deep CNNs described in the literature are scalability (Petitpierre et al., 2021) and the ability automatically learn and extract complex symbols (Xia et al., 2022) in challenging environments (Uhl et al., 2020) such as in maps with different quality as well as variation of cartographic symbols.

Even though the deep learning approach solves most of the limitations of classical feature extraction methods, it has challenges in digital map processing. The study done by Chiang et al., 2020 on training deep learning models for feature extraction from historical maps gives a comprehensive analysis of the subject discussing strengths and weaknesses. This study focuses more on a challenging aspect of deep learning, which is the requirement of an extensive training dataset. The authors have identified the lack of cartographic training data as the most limiting factor in applying CNN methods in feature extraction in digital map processing.

All deep learning methods including CNNs, require a large amount of training data to train a model successfully. Since there are no publicly available machine learning datasets for historical maps, the training data need to be created from scratch, which involves a lot of manual work. To overcome the issue of limited training data, the study by Chiang et al., 2020 adopts a technique called transfer learning. Transfer learning is a machine learning technique in which a model trained on one task is repurposed for another related task where the pre-trained model serves as the starting point to train the new model (Pan & Yang, 2010). The authors have compared four deep learning convolution neural networks (CNNs) to extract railroads from the United States Geological Survey (USGS) historical topographic maps. The CNN models used in this study are PSPNet, FCN-VGG16, FCN-GoogLeNet, and FCN-ResNet. The transfer learning strategy is used where PSPNet was pre-trained using the PASCAL VOC 2012 dataset and the rest with the ImageNet dataset. The reason for choosing two different pre-trained datasets is not explained. This study's highest achieved accuracy was 62.22% using PSPNet architecture. The lowest accuracy was 4.689% using the FCN-VGG16 architecture, followed by FCN-GoogLeNet and FCN-ResNet. The authors of the study mention that the achieved calcification accuracy would still be insufficient to obtain useable railroad data from historical maps. More refined approaches are necessary to extract features from historical map scans properly.

Table 3-1 : Accuracy comparison of different deep learning architectures in the study done by Chiang et al., 2020

Deep Learning Architecture	Transfer Learning Dataset	Accuracy
FCN-VGG16	ImageNet	4.689%
FCN-GoogLeNet	ImageNet	11.79%
FCN-ResNet	ImageNet	23.09%
PSPNet	PASCAL VOC 2012	62.22%

It is assumed that the source and destination tasks are comparable in transfer learning. However, in this study, the domains of the pre-trained models used as a starting point and the final model are very different. The domain of the pre-trained models used in the transfer learning approach (both ImageNet and PASCAL VOC) represent day-to-day objects, but the domain of the final models is cartographic symbols. Due to this reason, the transfer learning approach does not yield a better classification accuracy compared to a model trained from scratch, but it may trigger faster training.

Another approach to overcome the limiting factor of the unavailability of cartographic training data for digital map processing is to use a technique in machine learning called weak supervision. Weak supervision uses high-level or less accurate supporting data to provide labels for a significant quantity of unsupervised data, allowing for the generation of a large amount of training dataset much faster than creating it manually (Chen, 2019).

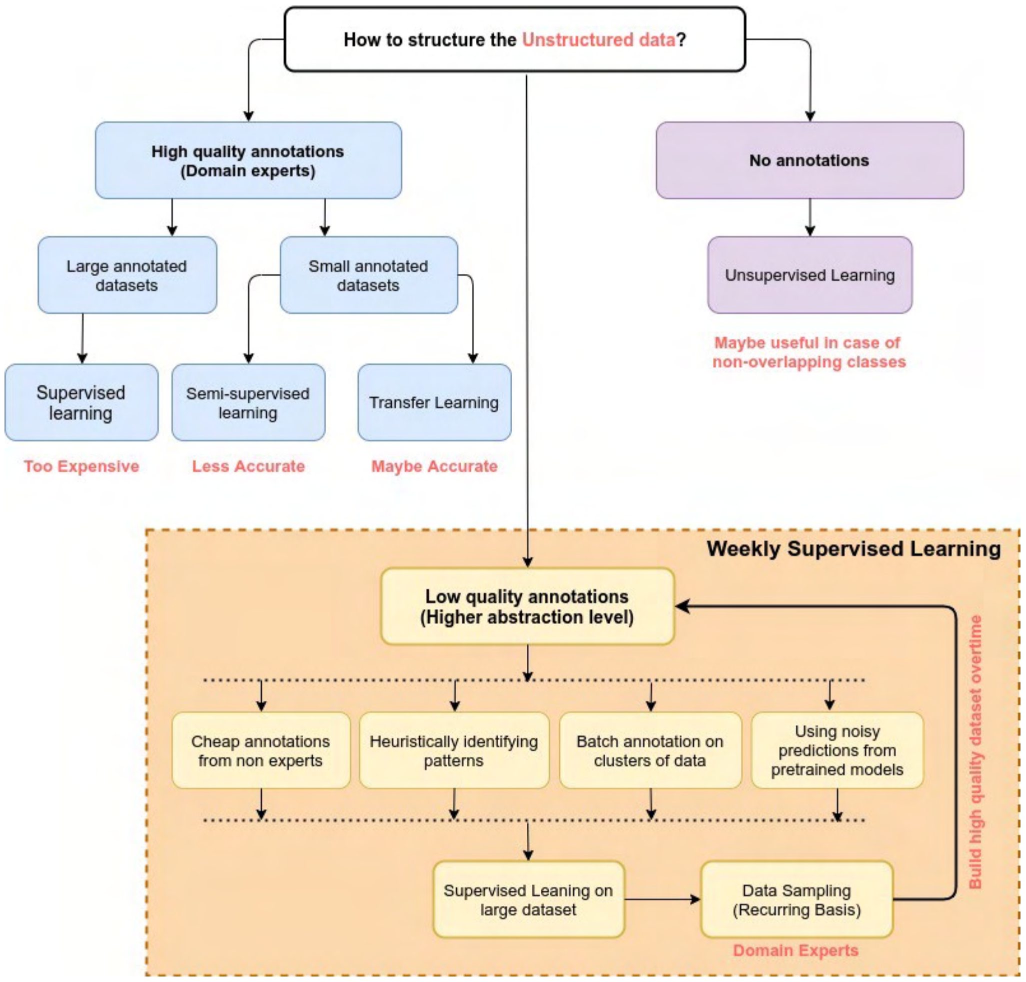

How to structure the Unstructured data?
High quality annotations (Domain experts)
No annotations
Large annotated datasets
Small annotated datasets
Unsupervised Learning
Maybe useful in case of non-overlapping classes
Supervised learning
Semi-supervised learning
Transfer Learning
Too Expensive
Less Accurate
Maybe Accurate
Weekly Supervised Learning
Low quality annotations (Higher abstraction level)
Cheap annotations from non experts
Heuristically identifying patterns
Batch annotation on clusters of data
Using noisy predictions from pretrained models
Supervised Leaning on large dataset
Data Sampling (Recurring Basis)
Domain Experts
Build high quality dataset overtime

Figure 3-6 : Various approaches compared to weak supervision for structuring the unstructured data to train deep learning models (Sharma, 2020).

The study by Uhl et al., 2020 uses a weakly supervised deep learning-based classification to extract human settlement patterns from USGS historical topographic map series. The authors face the lack of training data to be fed to a convolution neural network; so, as a solution, they have used a weakly supervised approach where current geospatial data is used as ancillary data to gather training samples. The method described was inspired by an application of deep learning in remote sensing, where the training data is efficiently generated using supporting geospatial data such as land cover maps. However, the authors describe that using such ancillary data in the context of historical maps to create training data only results in approximated boundaries due to temporal change, map distortions, high positional error and inaccurate georeferencing present in the historical maps. Since the training data generated with ancillary data only provides low-quality training labels, a weakly supervised training method was adopted to train CNN.

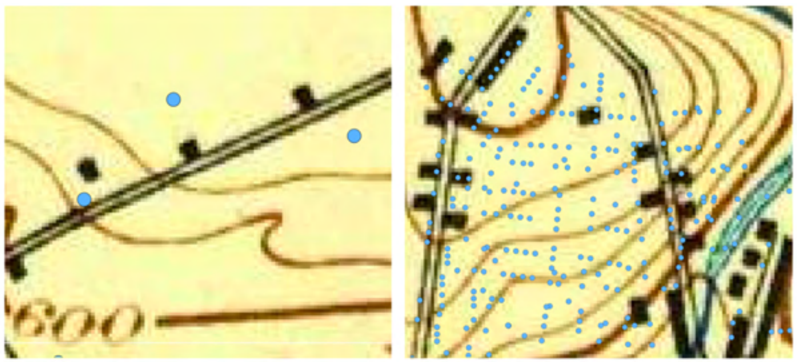


Figure 3-6 : Examples of spatial offset (left) and temporal offset (right) between building locations in historical maps and current building locations (blue dots) (Uhl et al., 2020).

The study was based on USGS topographic maps (1893-1954) and the ancillary spatial data used to create training samples are accurate settlement locations in 2016. First, the ancillary spatial data is preprocessed, and training data is made at the map patch level for settlement symbols using the preprocessed ancillary data. Second, the CNN model was trained using weakly supervised approach for patch-based classification to identify settlement patches, followed by pixel-level semantic segmentation to extract settlement areas. Three widely used CNN architectures (LeNet, AlexNet, VGGNet-16) were tested in this study to compare the performance in patch-based classification to assign patches of the map to settlement class or other land use. The highest classification accuracy was achieved with VGGNet-16, with an overall accuracy of 0.92. Since VGGNet-16 is the best-performing architecture in the image classification step, the authors have chosen VGGNet-16 to perform pixel-level semantic segmentation to extract building footprints. Semantic segmentation was performed using VGGNet-16 in a weakly supervised manner where the CNN is patch-level trained to output pixel-level segmentation. Despite the high accuracy in patch-level classification, the precision of pixel-based semantic segmentation drops to a level of 0.29. This study shows that weakly supervised CNNs can be applied successfully in a patch-based classification where a map is divided into small patches. The target is to tag each class to each patch (ex. urban, non-urban). But weakly supervised deep learning method does not yield acceptable performance in pixel-level classification for semantic segmentation since the model is trained using data with low granularity than the output.

Another interesting finding of the study by Uhl et al., 2020 is that the feature extraction accuracy increases with the graphical quality of the maps. This is because older maps generally have degraded graphical quality due to ageing compared to newer maps, and there is a trend of increasing accuracy over the time of the creation of the map.

The performance of semantic segmentation is highly dependent on the quality of the training dataset. Therefore, selecting proper deep learning architecture also plays a significant role in successfully applying semantic segmentation. The study by Ekim et al., 2021 and Heitzler & Hurni, 2020 elaborate on the above factors. The study by Heitzler & Hurni, 2020 used U-Net CNN architecture to extract building footprints from the Swiss Siegfried map resulting in a remarkable precision of 0.985. The high-quality training dataset is a major contributing factor to the high accuracy achieved in this study compared to the rest. Manual creation of a training dataset is a costly and time-consuming task, but the studies by Ekim et al., 2021; Heitzler & Hurni, 2020 proves that using a manually created high-quality training dataset will produce a high-quality deep learning model.

3.4 Selection of CNN Architectures

The focus of this study is to evaluate different deep learning architectures to extract area features from historical map scans. Two strategies are used in this study to extract area symbols which are explained in detail in next chapters. The semantic segmentation approach is taken to extract features where the area of the symbol has topographic meaning, and the object detection approach is taken when only the location of the symbol is important.

Since Convolution Neural Network methods are the state-of-the-art deep learning technology for image processing, CNN architectures were chosen for each approach to compare the performance in semantic segmentation and object detection. Many CNN architectures are available today, developed by scientists improving the existing architectures and inventing new methods to overcome limitations. Selection among these architectures must be made with a clear understanding of the application and the available resources. In this study, 3 CNN architectures were selected based on a literature review considering their performance in digital map processing or similar tasks such as feature extraction from remote sensing data.

3.4.1 Semantic Segmentation Architectures

UNet

The researchers Ronneberger et al., 2015created the UNet architecture for biomedical image segmentation in 2015. It is currently one of the methods used the most frequently for semantic segmentation tasks. It is a fully convolutional neural network built with a smaller training sample size in mind. A contracting path and an up-sampling path define the ability of the UNet architecture to perform semantic segmentation. The contracting path has pooling and convolution layers which generate an abstract representation of the input image. These abstract representations are combined by the up-sampling path that generates pixel-level inference.

UNet is the architectural choice for the previous work carried out by the historical map semantic group, which became the foundation of this research. It was selected as the first choice for semantic segmentation in this study because UNet has proven to work well with historical map data for feature extraction. There are several studies demonstrating this fact. Heitzler & Hurni, 2020 have used UNet architecture to successfully extract building footprints from the Swiss Siegfried map series with an average accuracy of 88.2%. In ICDAR 2021 Competition on Historical Map Segmentation, a team using a modified version of UNet won the segmentation task by achieving 74.1% accuracy.

ResNet

ResNet stands for Residual Network, which is the winning architecture of the ILSVRC 2015 competition. ResNet was introduced in the paper "Deep Residual Learning for Image Recognition" by He et al., 2016 achieving the objective of designing a very deep convolution neural network free of vanishing gradient problem. The basic concept of solving the vanishing gradient problem is introducing bypasses to the computations. ResNet is a feedforward CNN architecture with the bypass connections named "residual connections".

ResNet holds the first place for the SpaceNet benchmark of 2022, which is for building detection from satellite images with an achieved accuracy of 78.48%. The study was done by Chiang et al., 2020 compared different CNN architectures in challenging environments to extract rail roads from map scans. In the results, the ResNet architecture outperformed the competitors by achieving twice the accuracy in comparison, but all the compared architectures haven't reached acceptable performance due to data limitations. Finally, the study by Petitpierre et al., 2021 uses ResNet architecture for generic semantic segmentation of historical maps of Paris. The CNN model could classify buildings with very high accuracy of 91% and road networks with 75%.

InceptionResNet

InceptionResNet CNN architecture was developed by Szegedy et al., 2016 as an upgraded version of the original Inception architecture. Original Inception architecture was developed by Google (Szegedy et al., 2014) and was the winner of the 2015 ImageNet challenge with an error rate of 6.67%. The improvement made to the original architecture is intended to minimise the computational cost with less effect on the depth of the architecture. This improvement was driven by the introduction of residual blocks and bringing them together with inception blocks. The developers of InceptionResNet have demonstrated that introducing residual blocks to the original architecture will significantly increase the processing speed while achieving similar results.

InceptionResNet is successfully used by Xu et al., 2019 to develop a feature reconstruction model for building segmentation using remote sensing data. They have compared the developed model performance with UNet architecture and have achieved a 3.7% improvement. The study by Erdem & Avdan, 2020 shows promising performance in InceptionResNet architecture to extract buildings from satellite images with 86% accuracy.

3.4.2 Object Detection Architectures

YOLOv5

YOLO stands for You Only Look Once, which is a CNN architecture that made a turning point in deep learning for object detection. It is designed to be simple yet effective in object detection, so it is most popular in real-time object detection applications. The architectural implementation of YOLO broke the traditional CNN implementation at its invention (Du, 2018) by combining two separate neural network processes into one process. In YOLO, detection and classification are done simultaneously, whereas the models before YOLO use two neural networks to perform the task in two steps and combine it at the end. The YOLO algorithm uses the concepts of residual blocks, bounding box regression, and intersection over union (IOU) (Redmon et al., 2016) to train and perform predictions by dividing an image into smaller grids and classifying each grid.

YOLOv5 is used by the researchers Ding & Zhang, 2021 to detect buildings from remote sensing imagery with a detection accuracy of 88.5. A similar study was carried out by Kim & Hong, 2021 have achieved an accuracy between 88% and 98% in various scenarios of satellite images.

In this research, we have used the fifth iteration of YOLO for object detection tasks. The architectures chosen are YOLOv5_m, YOLOv5_l and YOLOv5_X. The main architectural difference here is the depth of the neural network. YOLOv5_X is a deeper network with 86.7M parameters, while YOLOv5_l and YOLOv5_m having 46.5M and 21.2M parameters.

4 DATASET

4.1 Mileage Sheets of Saxony

This study is based on the dataset Mileage Sheets of Saxony (or "Sächsische Meilenblätter" in German), a series of historical topographic maps created between 1780 and 1825. The Saxon mile sheets were compiled for the Saxony region of Germany with a military topographic survey. The survey was based on a national geodetic triangulation starting from a baseline between Pirna and Königstein. The baseline distance was measured to a precision of 1cm using a cubit rod, whereas angle measurements of the triangulation made use of the theodolite. By adapting to the natural orientation of the baseline, the survey and the resulting maps were not aligned to the north but rotated about 42 degrees to the west. So, the mileage sheets point roughly in a northwest direction. Each map sheet is about 56.6 × 56.6 cm in size, corresponding to one Saxon Square Mile (approximately 6.8km × 6.8km) hence the name "Mileage Sheets" of Saxony. The original series consists of 445 individual map sheets. Two additional copies were made during the survey in addition to the original map series named Dresden copy which is named after the place where it was kept. The first copy, named Berlin copy, also called the royal specimen, was intended for the king, and the second copy is named Freiberg copy. The maps were created manually and drawn with ink in black, red, blue and brown. (Walz, 2002; Witschas, 2002).

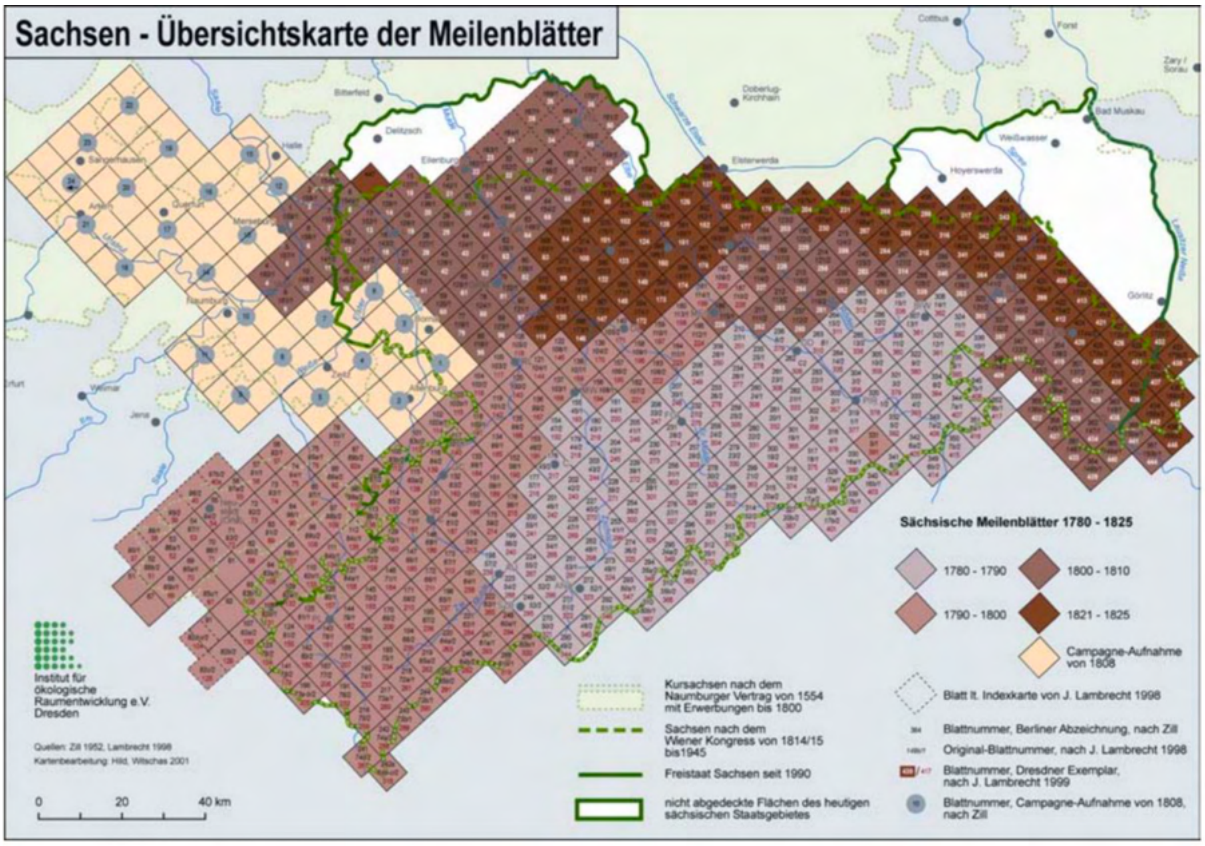

Sachsen - Übersichtskarte der Meilenblätter
Sächsische Meilenblätter 1780 - 1825
1780 - 1790
1800 - 1810
1790 - 1800
1821 - 1825
Campagne-Aufnahme von 1808
Kursachsen nach dem Naumburger Vertrag von 1554 mit Erwerbungen bis 1800
Sachsen nach dem Wiener Kongress von 1814/15 bis1945
Freistaat Sachsen seit 1990
nicht abgedeckte Flächen des heutigen sächsischen Staatsgebietes
Blatt lt. Indexkarte von J. Lambrecht 1998
Blattnummer, Berliner Abzeichnung, nach Zill
Original-Blattnummer, nach J. Lambrecht 1998
Blattnummer, Dresdner Exemplar, nach J. Lambrecht 1999
Blattnummer, Campagne-Aufnahme von 1808, nach Zill
Institut für ökologische Raumentwicklung e.V. Dresden
0
20
40 km

Figure 4-1 : Overview and the Saxon mile sheets topographic map series (Witschas, 2002).

The maps have a high degree of information and were created at a relatively large survey scale of 1:12,000. Even though the maps were primarily made for military purposes, it was also intended to meet several other needs, such as building roads and canals, mining, and public administration. The primary content of the maps shows borders, buildings, road networks, and water bodies, including streams, forests and meadows. The relatively large scale of 1:12,000 allowed representation of individual buildings and even the courtyards when scale allows. The maps show detailed information on the smallest features such as footpaths, drainage ditches, gardens, stone blocks, etc. However, no legend explains the symbology used in the map sheets. A clear understanding of the cartography symbology in the map series is required in this study. A description of the majority of map symbols used has been published by the Saxonian agency "Geobasisinformation und Vermessung" and has been used to interpret the topographic maps.

In Saxon mile sheets, hatching was used to designate the relief. Hachures are small lines drawn to represent slopes. The lines are drawn thicker for steeper slopes, while for a gentler slope, they are drawn thinner. This is noteworthy because it is an early method of depicting relief, but it also visually dominates the map. It is a challenging aspect of this study because symbols overlapped by hachures are hard to distinguish even to the human eye. By 1870, hatching had been replaced by contour lines in so-called "equidistant maps," which significantly updated the content of the Saxon mile sheets.

Virtual Map Forum of the Saxon State Library (https://kartenforum.slub-dresden.de/) provides public access to the high-quality scanned maps of the Saxon mile sheets. Virtual Map Forum serves as a web-based interface to access the digital map archive of the Saxonian State Library, which has more than 20,000 historical maps preserved in digital format (Mendt, 2014). Georeferenced map sheets of the Saxony topographic survey were made available through the web client, where users can filter, visualise and download. In addition to publishing the georeferenced maps on the virtual map forum's web page, standardised spatial data infrastructure interfaces (WMS/WCS/CSW) are made available to the public. Researchers can use Web Mapping Service (WMS) and Web Coverage Service (WCS) to easily integrate historical maps into local or web-based geographic information systems. However, vectorisation of the map content is not available for the maps; hence the services are only provided with the scanned raster images. Due to this reason, only visualising the maps in GIS systems is possible, but to perform spatial analysis, it is required to extract the data locked in the map scans.

The Saxon mile sheets provide a sound cartographic basis for the historical landscape of Saxony in the 18th century. The historical map series allows for the investigation of landscape change over the last 200 years and offers adequate resolution in geometry and content for the research. However, historical maps should only be used in analysis with caution because survey techniques, accuracy, map details, and production methods are significantly different to today's standards hence inducing some limitations such as the shift in buildings (Haase et al., 2007).

4.1.1 Selected Data Set and Study Area

The original map series of the Mileage Sheets of Saxony consist of 445 map sheets. In this study, ten map sheets from the Mileage Sheets of Saxony were selected considering the challenging aspects of deep learning in feature extraction from historical maps. Studies have shown that the quality of the map scan affects the quality of the feature extraction. Historical maps have different graphical qualities due to ageing and the manual process of creating the maps where the scans of the Mileage Sheets of Saxony show no exception. As shown in the figure 4-2, the graphical quality changes from one sheet to another. In this study, the variation factor in the graphical quality of historical maps was considered when selecting the dataset. Available map sheets were visually observed, and ten maps were selected concerning varying graphical quality. All the maps were downloaded in the highest available resolution in GeoTIFF format that were available to the public through the Virtual Map Forum of the Saxon State Library.

Each map scan has a width of 12500 pixels, a height of 8000 pixels, and an approximate spatial resolution of 1.1 m/pixel. The dimensions and the spatial resolution changes from one map scan to another. All the selected map scans were already georeferenced using DHDN geographic coordinate system.

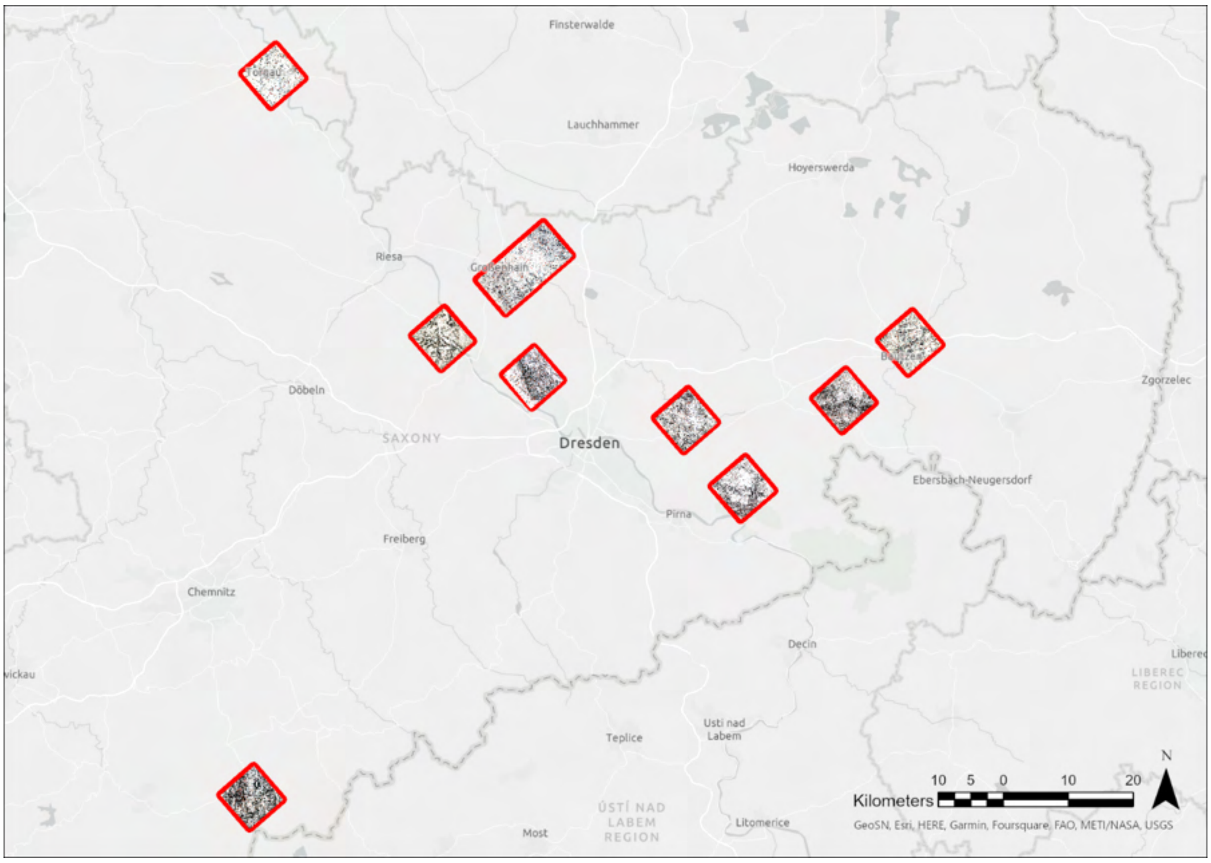


Figure 4-2 : Spatial distribution of the selected map sheets.

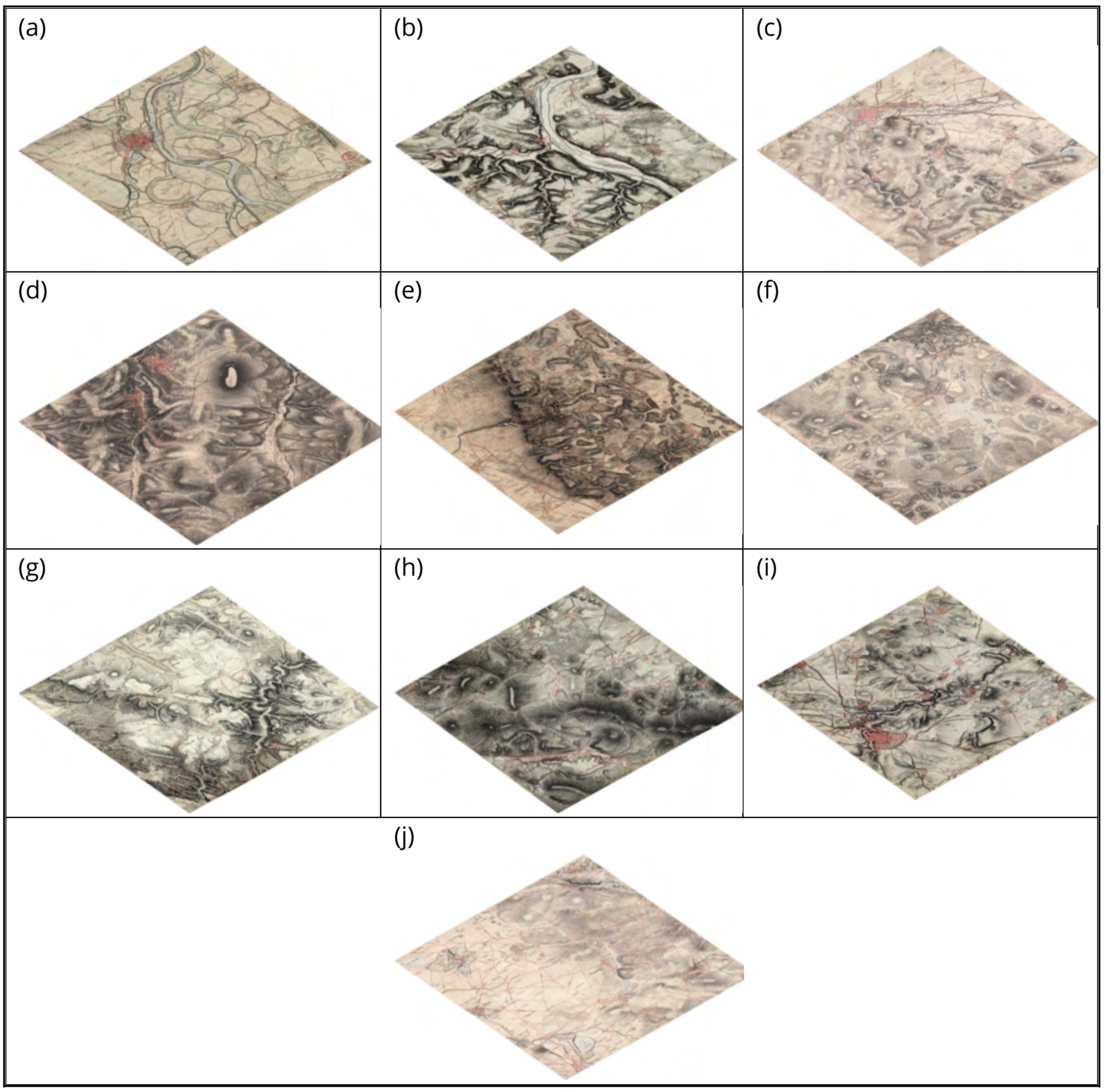


Figure 4-3 : Selected Map Sheets [index, location, sheet number in Virtual Map Forum] (a) Torgau, No 050; (b) Zehren, No 161; (c) Groszenhain, No 163; (d) Annaberg, No 181; (e) Coswig, No 211; (f) Arnsdorf, No 285; (g) Hohnstein, No 321; (h) Gauszig, No 324; (i) Bautzen, No 326; (j) Lampertswalde, No 164.

4.2 Data Pre-Processing

Pre-processing of the dataset was carried out with one goal in mind. To make the selected map images consistent in terms of colour and map geometry. Pre-processing is necessary to make a consistent dataset in the context of historical map processing to train deep learning models.

There are two main issues to be addressed.

1. Variation of the graphical quality in different map sheets.
2. Geometric error in map symbols

4.2.1 Colour Homogenization

The dataset is chosen to represent a general scenario of historical maps where the graphical quality of the maps is not consistent. Specifically, to the Mileage Sheets of Saxony, the period to create all the maps took nearly fifty years, starting in the late 18th century. All the maps created are subjected to technological limitations of the time and subjective variations due to the manual cartographic process. Different cartographers have worked on making this topographic map series, so as a result, the cartographic style and colour vary. The variation of the colour is also a result of ageing. Since the map sheets are more than two centuries old and since all the paper maps are not ageing the same way, variation in the graphical quality of the scans is unavoidable.

Colour homogenisation is carried out to decrease the variation of the graphical quality of the selected map sheets. The working group developed an in-house algorithm to homogenise the colour of maps. The algorithm uses the principle of histogram equalisation to balance the hue between all the map images and increase the contrast. The figure 4-4 shows the output of two map scans with varying graphical quality before and after colour homogenisation.

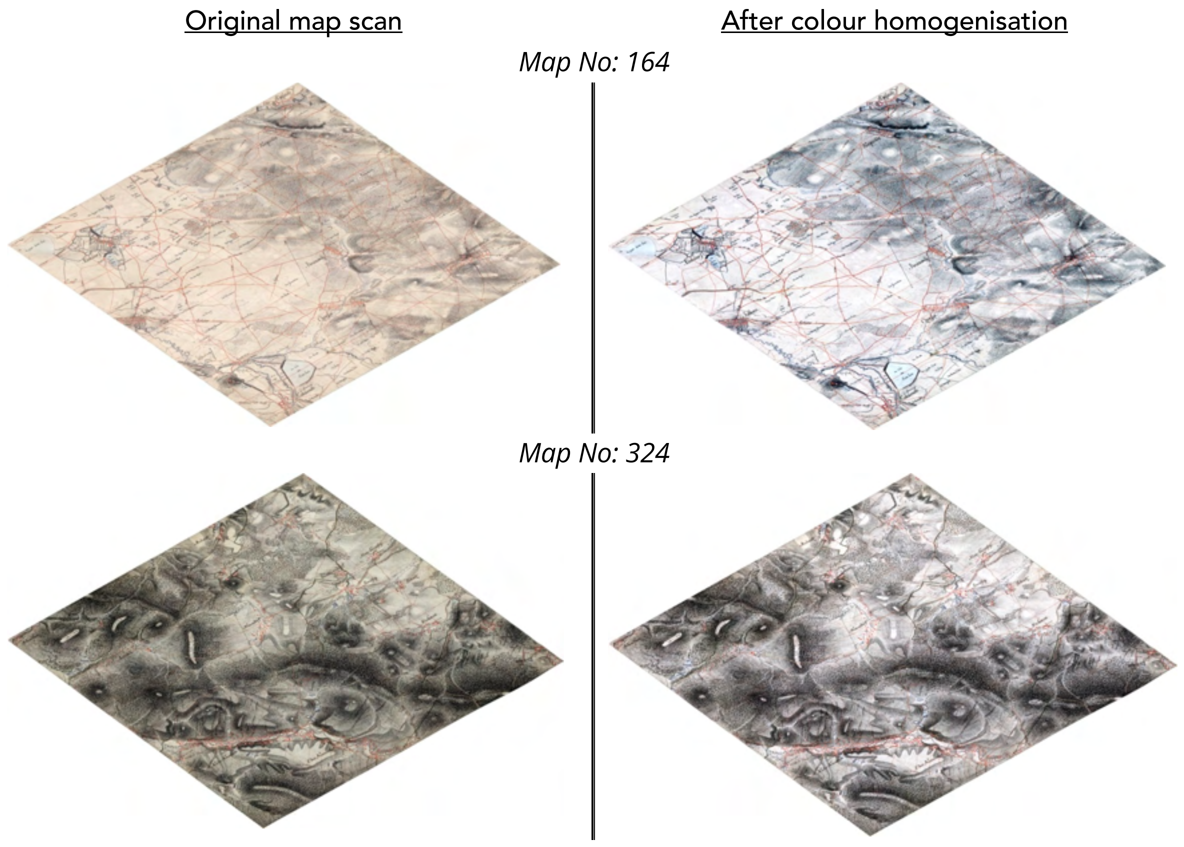


Figure 4-4 : Comparison of visual variation of the original map scans and colour homogenised maps.

4.2.2 Re-Projection of the Dataset

The dataset of this study downloaded from Virtual Map Forum are georeferenced scans of the Mileage Sheets of Saxony. The scanned copies of the original map sheets did not initially contain digital geographic coordinate information. Instead, they were just scanned images of map sheets. Later, the Saxon State and University Library Dresden (SLUB) performed georeferencing of the scanned map sheets to DHDN coordinate system (Mendt, 2014). Georeferencing is the process of assigning geospatial coordinates to raster data based on the defined coordinate system. This process involves finding a set of control points on the scanned map with known geographical coordinates and transforming the image based on the identified control points. This transformation permanently affects the image pixels based on the selected coordinate reference system, which results in re-sizing and re-shaping the original image.

The original map sheets of the Mileage Sheets of Saxony are square in shape with a height and width of 56.6 cm. However, digital georeferenced maps provided by the Virtual Map Forum do not preserve the square shape of the maps (figure 4-4), which is caused by georeferencing to the DHDN geographic coordinate system. This induces an error in the shapes of cartographic symbols that adversely affect the end goal of digital map processing.

Let's take an example from the building symbols. Small buildings are represented in the maps as red colour rectangles or as a composition of rectangles depending on the actual shape of the building. However, due to the geometric change that occurred at georeferencing, the building symbols have a skewed shape. Figure 4-5 is an exaggerated representation of the effect of georeferencing with the DHDN coordinate system on building symbols.

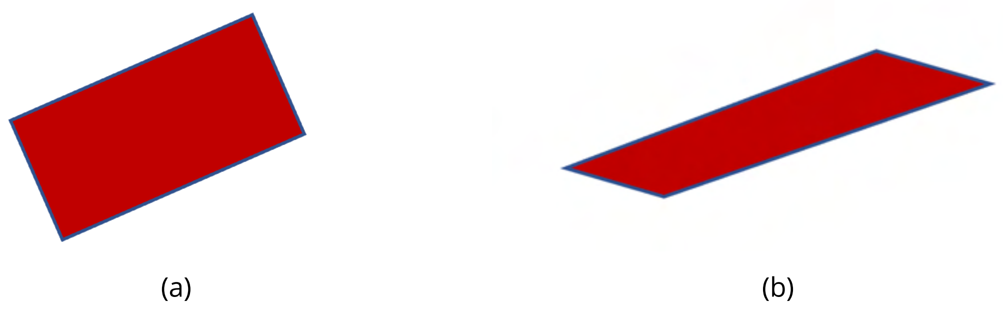
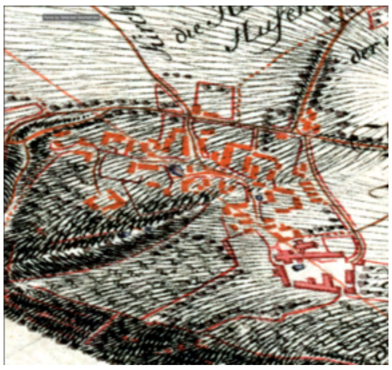
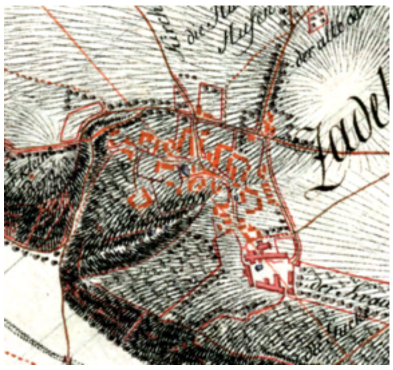
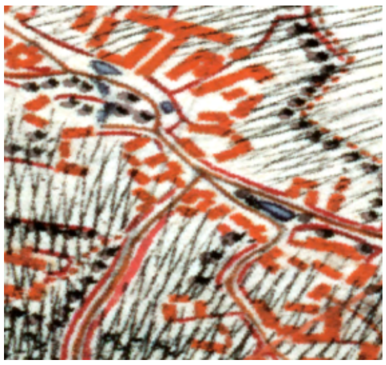
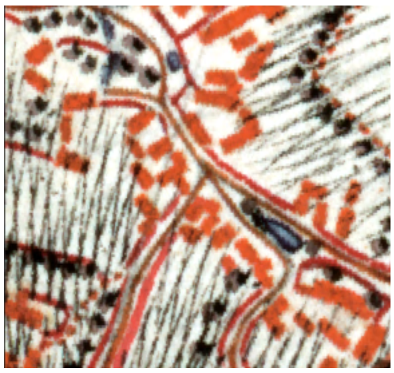


Figure 4-5 : Comparison of the geometric shape of buildings in (a) original maps and (b) georeferenced map scans available in the Virtual Map Forum.

Accurate geometric representation of the cartographic symbols is important because the end goal of digital map processing is to create meaningful vector data from the map scans that can be used in the geospatial analysis (Chiang et al., 2014). Creating vector data for building symbols in a scanned map can be broken into two general steps. The first is to extract the buildings as raster masks, and the second is to convert the raster masks to vectors. The rectangular shape of the building symbols becomes essential in the second step. When converting the extracted masks to vectors, a ruleset can be developed to detect the corner points and enforce the perpendicularity of the edges. Heitzler & Hurni, 2020 have successfully used such an approach to vectorise buildings. To use such an algorithm, the building symbols of the input need to have the characteristics mentioned above.

The geometric error in cartographic symbols can be corrected by reprojecting the dataset to the appropriate conformal projection system. Conformal projection is defined in cartography as projection systems which preserve local angles (Grafarend & Krumm, 2014). As a result, the shapes of the features *appear* to be true. Explained with a real-world example of a right-angle road intersection mapped in a conformal projection system, the line symbols representing the road will cross at a 90-degree angle.

Considering the importance of the correct geometric shape of the map symbols, a workflow is developed and implemented in Python to reproject the dataset to the intended shape of the original maps. Universal Transverse Mercator (UTM) was selected as the target coordinate system to reproject the maps. UTM is a conformal map projection system based on Transverse Mercator, where the earth is divided into 60 zones, each zone with a 6-degree longitudinal width (Field, 2018). Depending on the location of the earth, an appropriate zone needs to be selected when projecting data into UTM coordinate system. The python program written for reprojecting the dataset automatically identifies the appropriate UTM zone by calculating the centre coordinate of each map sheet. Each map sheet is individually analysed by the program. The only user input required is the folder path containing the map rasters. Python GDAL (Geospatial Data Abstraction Library) library was used in the program to open, read and reproject the raster images. The results of reprojecting the data can be seen in the figure 4-6. The complete workflow of pre-processing is provided in the figure 4-7.

Input Data Original Coordinate System (DHDN)	**Output** Reprojecting to UTM Coordinate System
Effect on building symbols	
Scale: 1:2000 	
Scale: 1:1000 	

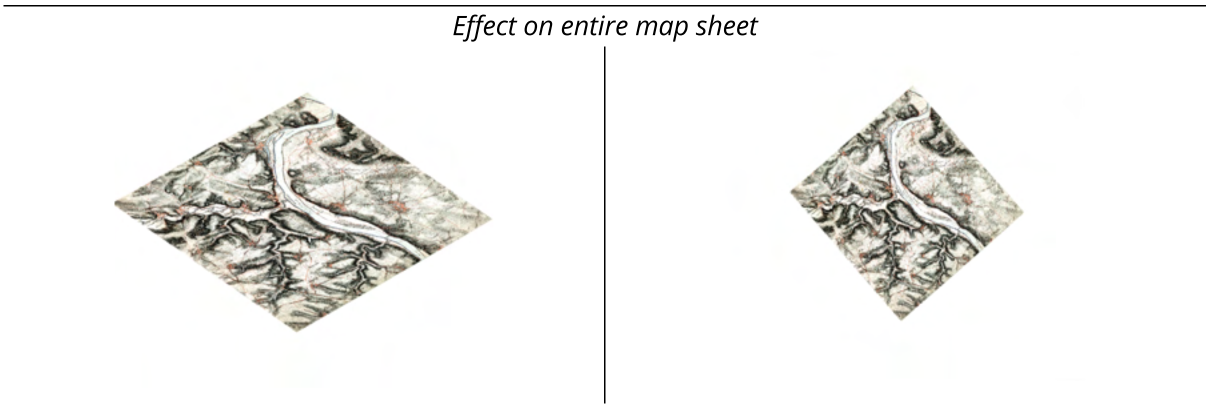


Figure 4-6 : Output of reprojecting the map scans to the UTM coordinate reference system.

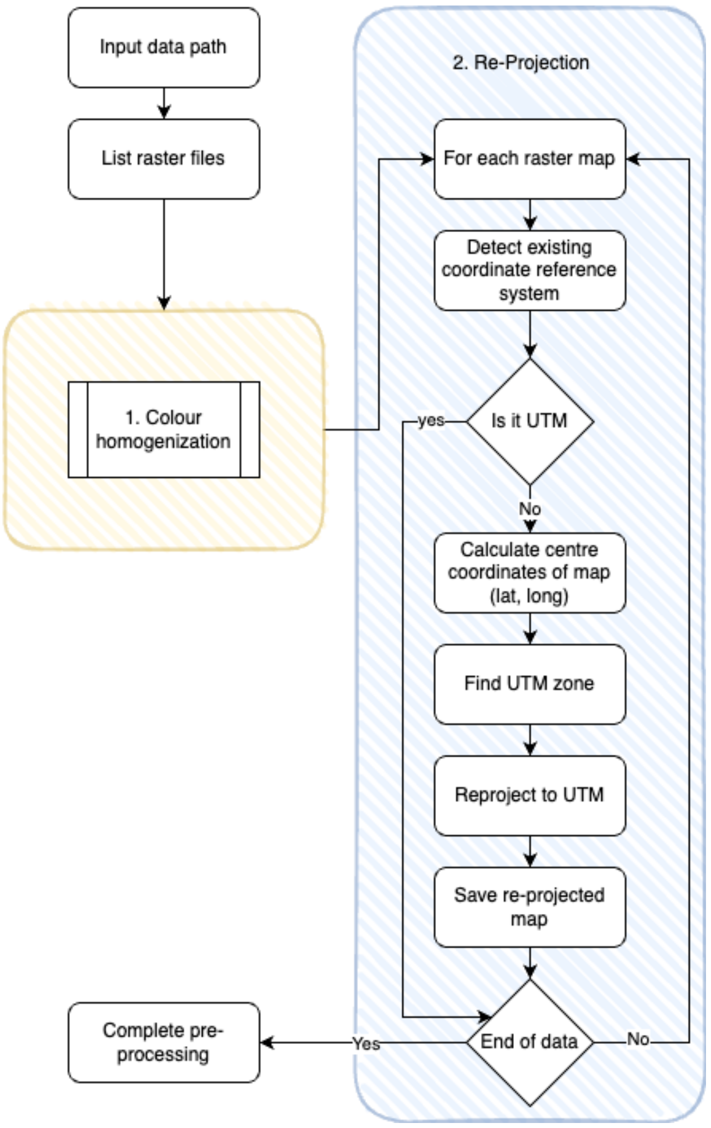


Figure 4-7 : Data pre-processing workflow.

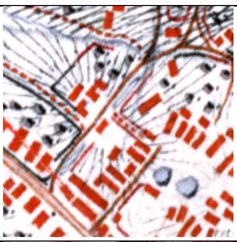
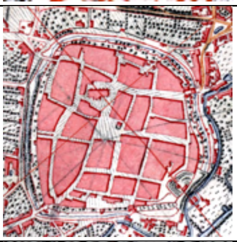
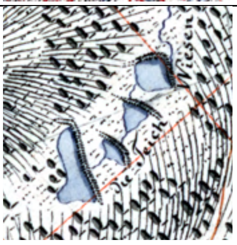
5 METHODOLOGY

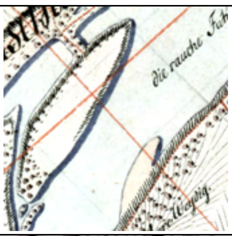
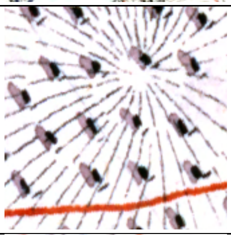
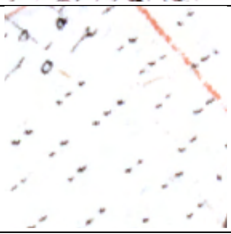
5.1 Selection of Map Features

In the mileage sheets of Saxony, a very detailed representation of the geographic features can be seen on the maps. To fully use the high degree of information available in the Saxony milage sheets, it is required to define subcategories of the symbols as different classification classes. Merging the classes in the classification (ex: small buildings + complex buildings as a single building class) is not a good option because the symbology of the sub-classes is significantly different in terms of size, shape, colour and texture. It will make it difficult for the deep learning algorithm to learn the ruleset to map the semantics between significantly different symbology. This hypothesis is backed up by a study done by Uhl et al., 2020 as the authors did not achieve a good performance in their deep learning model to extract large buildings when small and large building symbols were taken as a single classification class. The authors have suggested considering small buildings, large buildings and building blocks as separate classes as a solution to increase the accuracy of the model.

The scope of this research is limited to extracting three main geographical feature categories represented by area symbols, buildings, vegetation and water bodies. According to the symbology variation, six classification classes were selected, as shown in the table 5-1.

Table 5-1 : Selected classification classes and relevant symbology.

Geographical feature Class	Classification class	Symbol	Description
Buildings	Small buildings		Red rectangles or composition of red rectangles
	Building complex		Light red area with a red border (according to the actual size and shape of the building complex)
Water bodies	Lakes		Blue coloured area with a black dash line

	Rivers		Light blue area
Vegetation	Forest		Cluster of single black tree symbols
	Meadow		Cluster of black dots or small black lines

5.2 Feature Extraction Strategy

Extracting features from map scans using deep learning can be identified as a computer vision task of convolution neural networks. There are four main computer vision methods in deep learning: image classification, object detection, semantic segmentation and instance segmentation. In image classification, the deep learning model categorises images into different classes. Object detection identifies intended features in images, usually by using bounding boxes. In semantic segmentation, pixelwise classification is performed, where each pixel is assigned a class. Semantic segmentation produces numerous image regions referred to as image segments. Finally, instance segmentation goes one step further by performing pixelwise classification with localising objects in the image that belongs to the same class. (An overview of semantic segmentation and object detection was given in chapter 2).

In this study, the feature extraction strategy was selected based on the characteristics of symbology in Saxony mileage sheets. Buildings and waterbodies are symbolised by area symbols on the maps. The symbol's location, area and shape have a geographical meaning in the real world as it corresponds to the actual location, area and shape. In order to extract all this information, a pixelwise classification is necessary. Semantic segmentation is used to extract buildings and water bodies to fulfil this requirement.

Regarding the symbology of vegetation class, both forest and meadow symbols are composite symbols. The Forest symbol comprises a cluster of tree symbols (a point symbol), and the meadow symbol by a cluster of black dots or lines. In the symbology of forest and meadow, the individual point symbols do not have a geographical meaning regarding their area and shape. The geographical area of the symbol is visualised by clustering individual symbols. Due to this reason, forest and meadow areas were extracted with an object detection approach where a deep learning algorithm only detects the location of the individual symbols. Detected locations are used to reconstruct the area using a clustering algorithm. As a result, it is possible to generate vectorised output which makes it one step further to the end goal of digital map processing. The workflow of the two selected feature extraction strategies is shown in the figure 5-2.

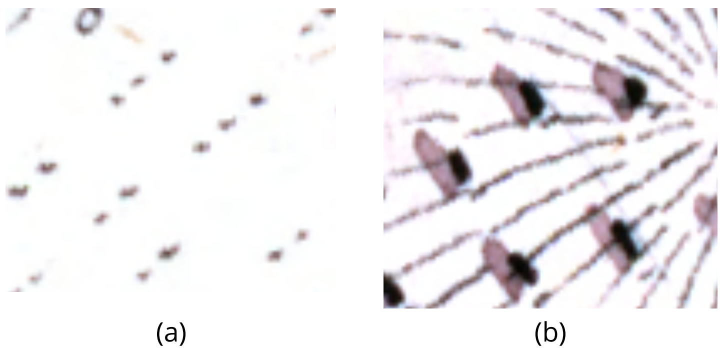


Figure 5-1 : Composite symbols used to represent (a) meadows and (b) forests.

5.3 Creating Training Dataset

A high-quality training dataset is the key to successfully training any deep learning model. Unfortunately, creating a training dataset for deep learning from scratch is a costly, time-consuming and challenging task. Several literatures (Chiang et al., 2020a; Uhl et al., 2020) in the context of historical map processing have explained that the major limitation of using deep learning technology in feature extraction is the lack of publicly available training datasets. Researchers (Chiang et al., 2020a; Uhl et al., 2020) have attempted to solve this issue of deficiency in training data with alternative approaches like transfer learning and weakly supervised learning, but the end results were insufficient to apply in practice. Considering the facts, it was decided to create the required training dataset manually. The created training dataset is only valid for maps of the Saxony milage sheets topographic map series.

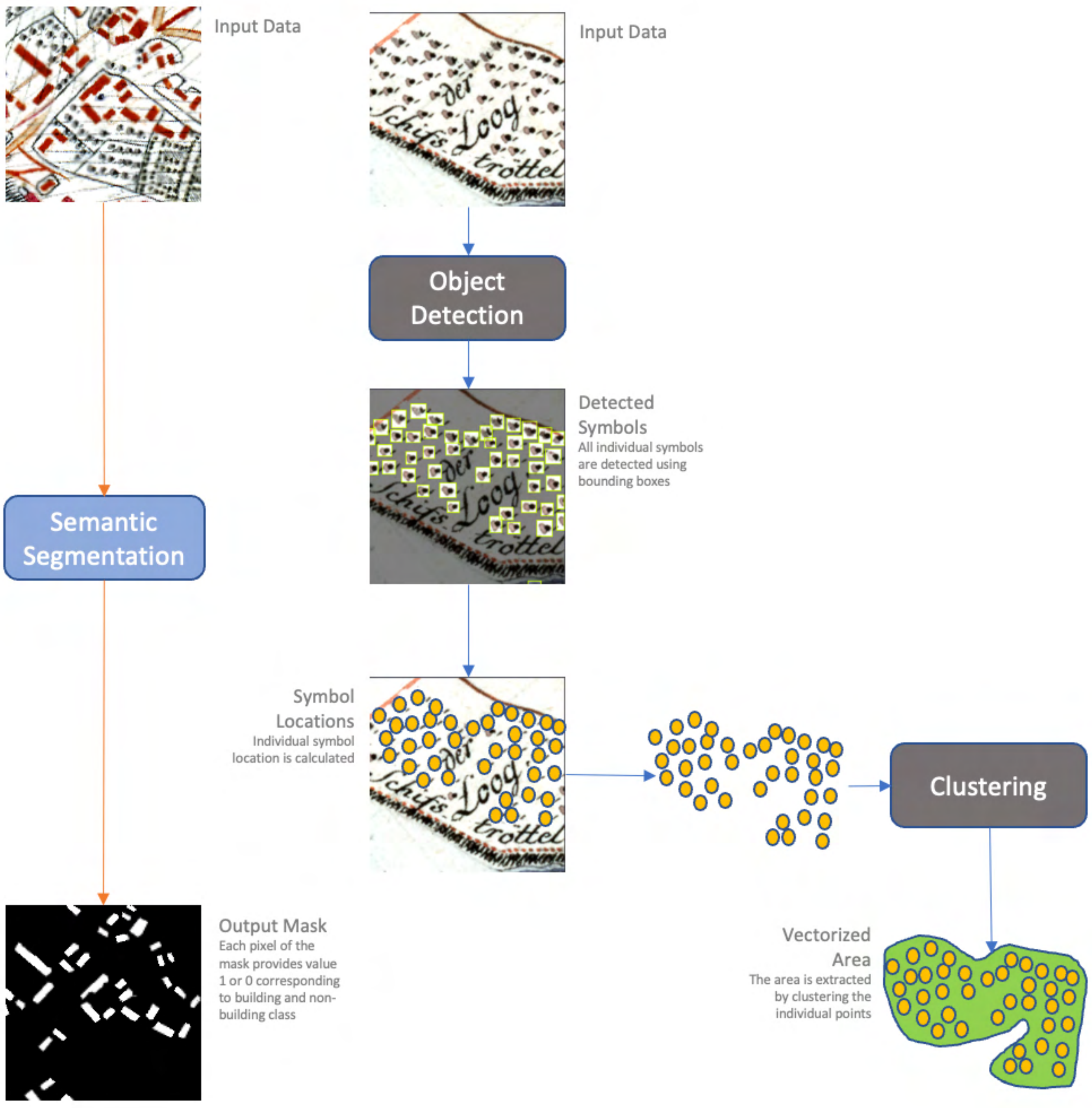
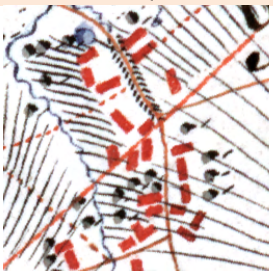
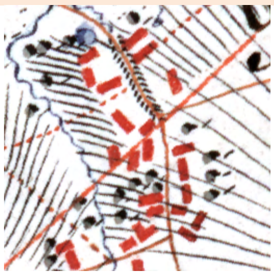
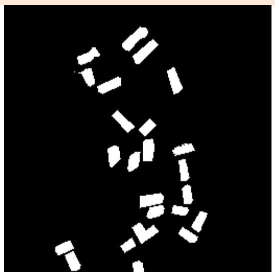

Input Data
Input Data
Object Detection
Detected Symbols
All individual symbols are detected using bounding boxes
Semantic Segmentation
Symbol Locations
Individual symbol location is calculated
Clustering
Output Mask
Each pixel of the mask provides value 1 or 0 corresponding to building and non-building class
Vectorized Area
The area is extracted by clustering the individual points

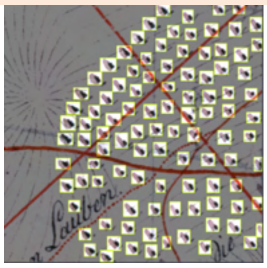
Figure 5-2 : Semantic segmentation and object detection workflows for feature extraction.

The process of creating the training dataset can be broken into several steps.

1. Selection of candidate training patches from the map scans for each feature class.
2. Extraction of the training patches from the map sheets.
3. Manually annotate the extracted patches to create an initial training dataset.
   a. For semantic segmentation: accurate boundaries of the features are annotated, which were then converted to raster masks.

Input	Training data		
			
RGB image patch	RGB image patch	+	Binary Mask

   b. For object detection: accurate bounding boxes are annotated.

Input	Training data		
			
RGB image patch	RGB image patch	+	Coordinates of bounding box

4. Data augmentation.
5. Splitting the training dataset.

The selected map sheets from the Milage sheets of Saxony are the primary data source for creating the training dataset. Since all the maps are georeferenced, they can be treated as any other geospatial data source in raster format (ex. satellite image, aerial image) in GIS software. Selection of the candidate patches was made, taking advantage of functions available in GIS software. The locations of the candidate training patches were randomly selected from all ten maps, and the centre coordinates along with the attributes were recorded in the shapefile (shapefile is a form to store geospatial data).

A script was written using Python to extract the training patches from map scans. The script takes the locations saved in the shapefile, classification class name, and required patch size as the input parameters. Then, it automatically extracts the training patches and appropriately names each image patch so it can be identified later in the map scans.

The extracted training patches are just raw RGB images. They do not contain any labelled information so that the deep learning model can understand the features in the images. Therefore, image labelling must be performed to add context to the raw training images. In the machine learning field, the process of creating the labelled data for images is called annotating. The labelled data created for semantic segmentation are binary feature masks. For object detection, the labels are bonding boxes drawn around map features.

In order to annotate raw training patches, a simple GUI was developed using Python. The tool was developed with simplicity in mind and to speed up the tedious process of creating a training dataset. At the initialisation, the user must select the annotation mode and the classification class. After that, the software automatically verifies and loads the image patches so the user can navigate through and annotate.

When creating labels for semantic segmentation, annotation can be done in two ways with the developed software. The first method is to draw polygons manually, marking the boundaries of the features in images. The second method uses the assistive tools programmed into the software to draw the polygons semi-automatic way. A tool named "magic tool" is implemented where this tool can be used to automatically draw polygons over the image features, which can significantly speed up the process of annotation. The magic tool works with the following principle using two well know algorithms in image processing and GIS named flood fill (Y. He et al., 2019), and Ramer-Douglas-Peucker (Douglas & Peucker, 1973). Both algorithms are implemented using the Scikit-Image, an open-source image processing library for Python. When the user clicks on an image with the "magic tool", the corresponding pixel is taken as a seed point. The flood fill algorithm then detects pixels similar to the seed point. The detected pixels are then converted to a closed polygon by creating contours around the pixels. The Ramer-Douglas-Peucker algorithm finally simplifies the polygon. The manual polygon creation tool and the "magic tool" can be used interchangeably as the user desires. After annotating all features of an image patch, the user can submit the annotations and the software automatically creates a binary mask with an appropriate naming convention.

To create labelled data for object detection, a third-party framework called Roboflow-Annotate was used. Using the framework, it is possible to load the raw training patches and annotate bounding boxes of image features by drawing rectangles. The created bounding boxes are saved in a text file recording the centre x, y coordinates, width and height of the bounding boxes.

When creating a deep learning training dataset for feature extraction, there is no hard and fast rule to decide the amount of required training data. Training a deep learning model is an iterative process. The training data amount is increased if the result of the model indicates a lack of training data.

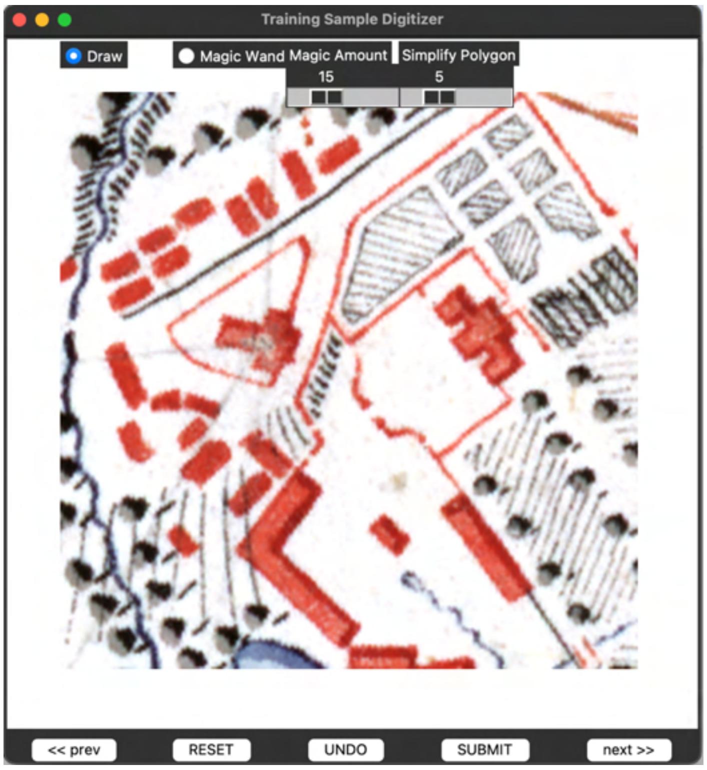


Figure 5-3 : The user interface of the annotating tool.

5.4 Data Augmentation

Since the training dataset is created manually, it takes a vast amount of time to generate adequate training data to train and evaluate deep learning models. Using the developed annotating tool takes approximately 5 minutes to create a single semantic segmentation training sample. Considering the time required to manually create a sufficient amount of training data, a method called data augmentation (Shorten & Khoshgoftaar, 2019) is used to increase the quantity of training data artificially. Many deep learning developers use data augmentation as a low-cost and successful strategy to lessen reliance on the manual preparation of a large amount of training data (Taylor & Nitschke, 2018). Data augmenting expands the size of the existing training dataset, allowing the creation of new training data without manually creating it. The methods used in this study to augment the training data are 1. Rotate 2. Geometric transformation 3. Rescaling 4. Adding Gaussian noise 5. RGB shift. Data augmenting was implemented using the Python library named Albumentations and integrated directly into the developed deep learning pipeline.

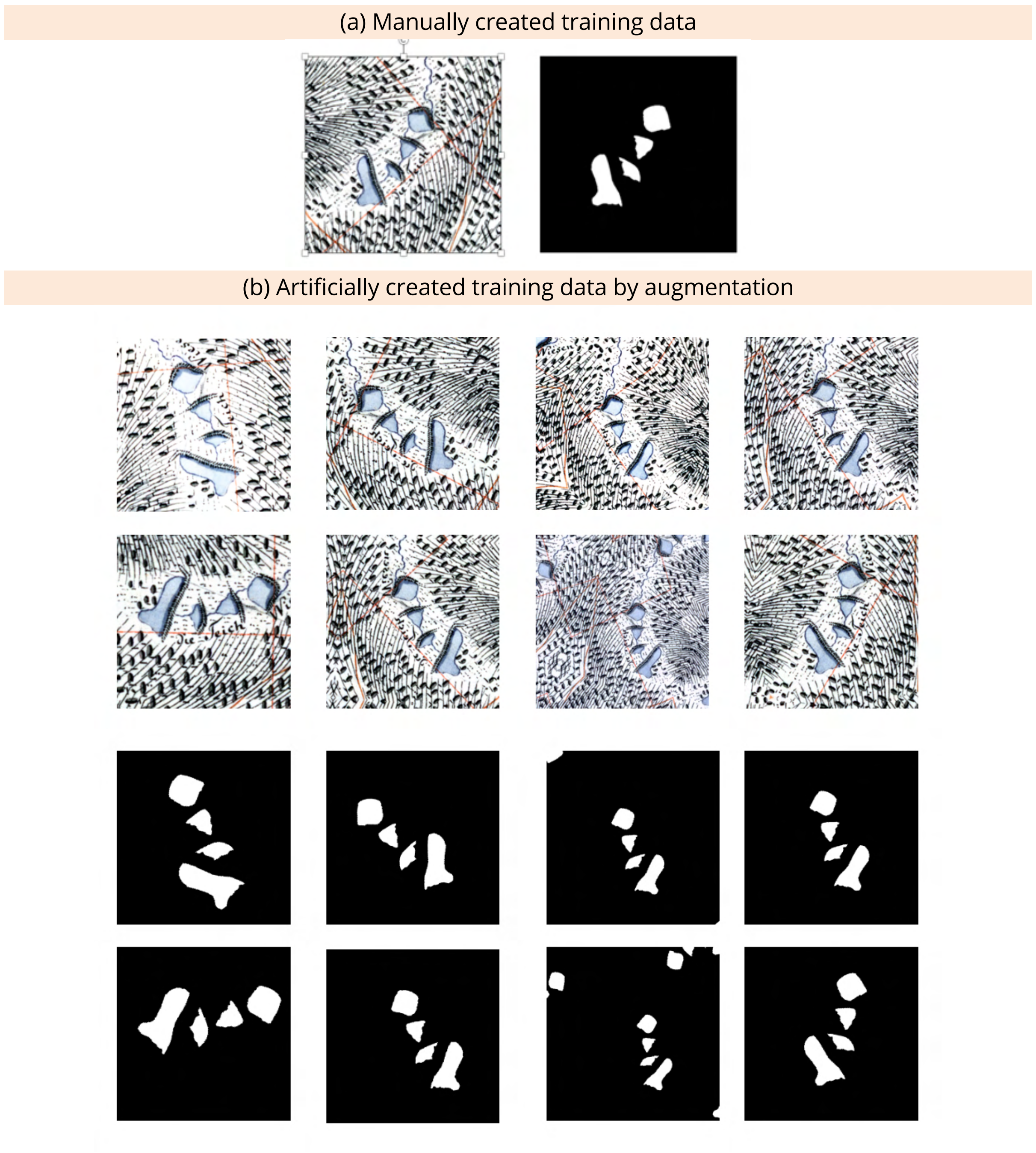

(a) Manually created training data
(b) Artificially created training data by augmentation

Figure 5-4 : Data augmentation applied on a data sample for lakes class.

5.5 Splitting the Training Data

In order to successfully train a deep learning model, the accuracy of the model needs to be evaluated during the training to monitor overfitting and at the end of the training to evaluate the final model performance. This evaluation must be done with a dataset unseen by the model in the learning phase. For a successful evaluation, the best practice is to split the training data into three categories. By convention, the categories are named Train, Dev, and Test sets.

Train Set: Training a deep learning model is an iterative process where the model needs to go over the same dataset multiple times (epochs), finetuning its parameters. This is referred to as learning in deep learning terminology. Train set is training data that is used to train and make the model learn the features in the data. In each epoch, the train set is fed to the neural network repeatedly, and the model continues to improve by adjusting model weights and biases.

Dev Set: During the training process, the Dev set (development set) is used to produce validation metrics such as validation-mAP (mean average precision) and validation-loss after each epoch. These measures are used to determine whether the model performs well or if the model overfits/underfit the Train data set. In addition, the model's hyper-parameters are adjusted based on the performance of the Dev set.

Test Set: After the completion of training, the model is tested using a different set of data called the test set. Since the test set's data were never seen by the model before, the final model performance is unbiased. In other words, performance in the test set answers the question, how well does the model perform in the real world?

In this study, the training data for each class was split into Train, Dev, Test sets with 70%, 20%, and 10% ratios, respectively. There is no general rule to calculate the allocation percentages to split the training data. Split percentages are decided based on the availability of training data. A significantly large portion of the training data is assigned to the Train set because the model is learned with the Train set. Splitting the training data was performed randomly, and this step is directly integrated into the deep learning pipeline.

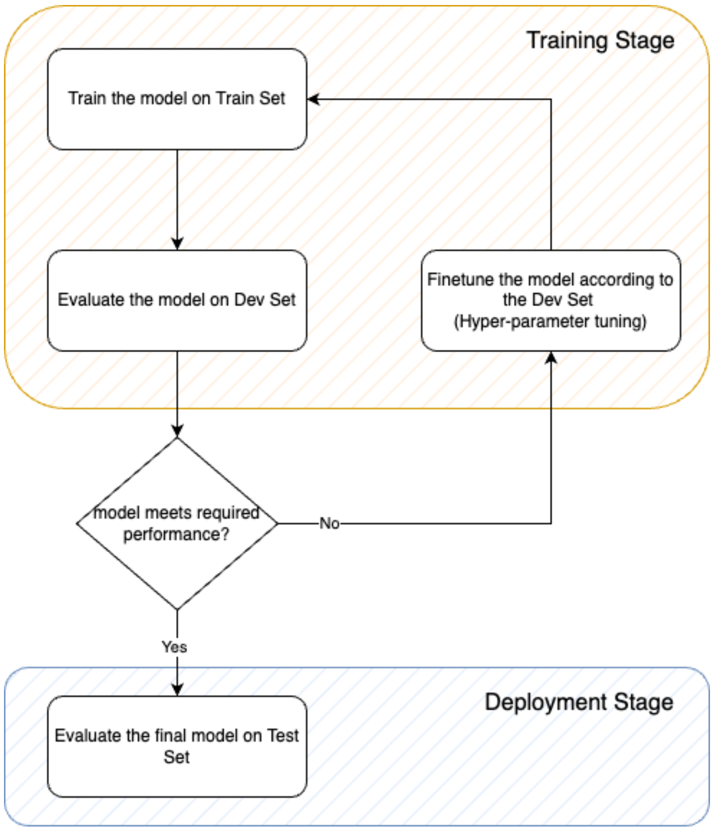

Training Stage
Train the model on Train Set
Evaluate the model on Dev Set
Finetune the model according to the Dev Set (Hyper-parameter tuning)
model meets required performance?
No
Yes
Deployment Stage
Evaluate the final model on Test Set

Figure 5-5 : Workflow with Train/Dev/Test data sets.

5.6 Implementation of Deep Learning Pipeline

The Deep learning pipeline to extract map features is the essential software created in this study. The deep learning pipeline is implemented with Python to automate training a deep learning model and evaluate its performance. The developed pipeline automates multiple steps such as input data validation, data augmentation and loading, training of deep learning model, model validation and generating visual outputs. As this study compares different deep learning architectures, the selected architectures are integrated into the pipeline. For object detection YOLO V5m, YOLO V5l and YOLO V5x architectures are integrated, and for semantic segmentation, UNet, InceptionResNetV2 and ResNet.

The backbone of the developed deep learning pipeline is the PyTorch (Paszke et al., 2019) machine learning framework (https://pytorch.org/). Since its introduction in 2017 by the Facebook AI Research team, PyTorch has become a widely popular and efficient framework for developing deep learning models. This open-source machine learning library is intended to boost the flexibility and speed of deep neural network research and implementation. PyTorch is one of the industry's most popular libraries among artificial intelligence researchers and practitioners (Paszke et al., 2019).

In addition to the PyTorch library, several key python libraries are used to develop the pipeline. Albumentations (https://albumentations.ai/) is the library used to implement data augmenting. In order to handle the vast image data and perform image visualisation Python Image Library (PIL) (https://pillow.readthedocs.io/en/stable/). The NumPy library (https://numpy.org/), the library of choice to most Python programmers for numerical computing, is used for the computation of matrices and tensors, which is required on numerous occasions in the pipeline.

In general, training and implementing deep learning models are computationally intensive processes. Therefore, it is recommended to the reader to use a high-performance PC with a dedicated Nvidia GPU (Graphical Processing Unit) or a cloud computing platform such as Google Colab (https://colab.research.google.com/). Google Colab is a free Jupyter notebook "like" environment that runs entirely in the cloud on a virtual machine. The free tier is equipped with Nvidia GPU, which makes it possible to train the deep learning models used in this study with adequate performance. The pipeline is designed to work with both PC and Google Colab environments. This study used Google Colab to perform all the heavy computing required in the pipeline.

5.7 Evaluation Methods

When training deep neural networks, descriptive evaluation measures must be used to capture the model performance. Therefore, in addition to presenting visual results, the performance of all the models used in this study was evaluated using widely used evaluation criteria. Following is a description of the evaluation methods implemented in the pipeline.

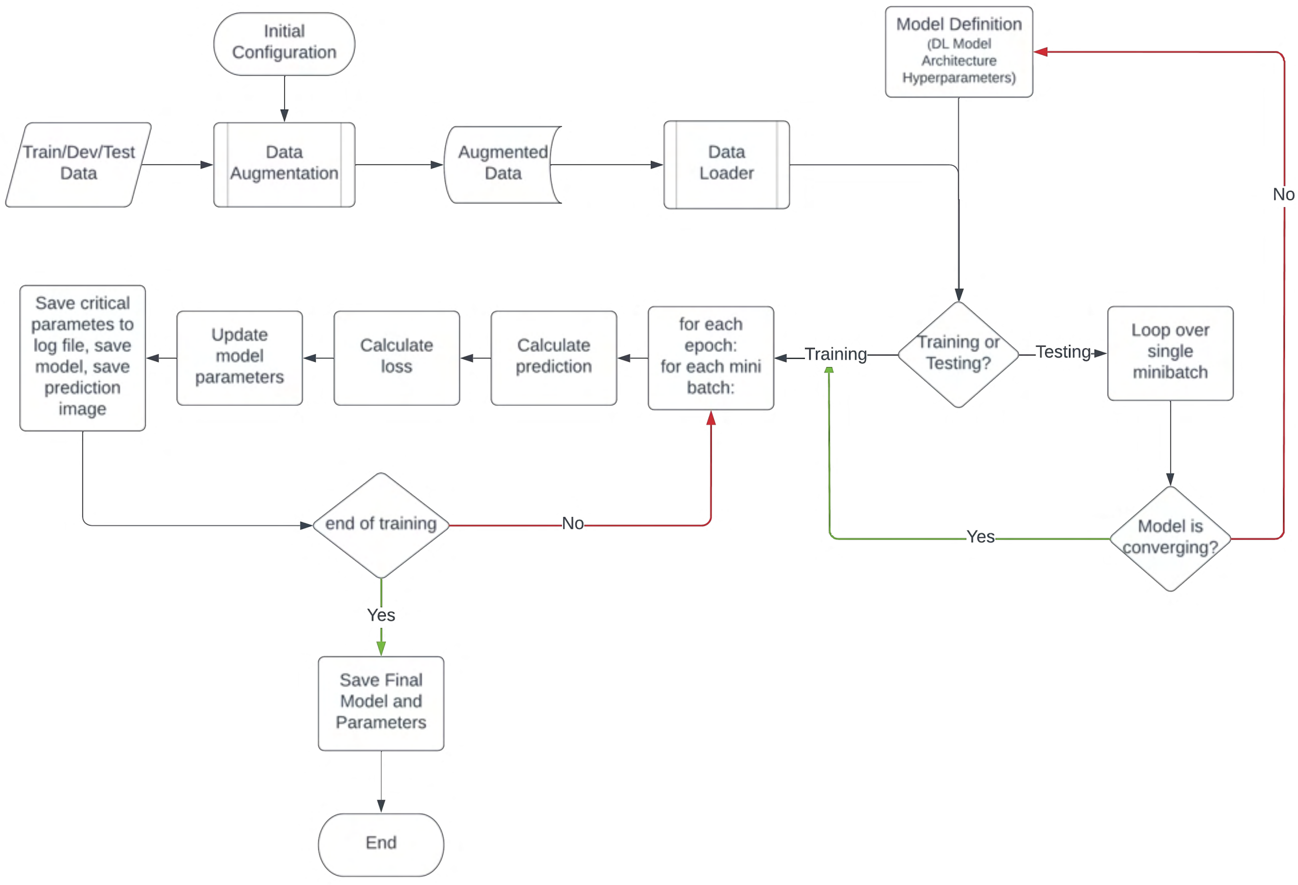


Figure 5-6 : An overview of the deep learning pipeline

5.7.1 Pixel Accuracy

Pixel Accuracy is the percentage of pixels in images that are correctly classified. In this study, pixel accuracy is only used to evaluate semantic segmentation models. Since a single class classification approach is taken in the study for semantic segmentation, the resulting binary mask is evaluated against the ground truth. In the binary mask, true positive (TP) represents a pixel that is correctly classified, and true negative (TN) represents a pixel correctly classified as not belonging to the class (correctly classified negative case). Furthermore, false positives (FP) and false negatives (FN) are the number of misclassified positive and negative cases.

Equation 4: Pixel Accuracy

$$Pixel\ Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

5.7.2 F1 score

F1 score is defined as the harmonic mean between precision and recall. Precision is the fraction of true positives in an image classified correctly among all the positive predictions. On the other hand, recall is the fraction of true positives in an image classified correctly among all the positive ground truths. So, precision represents the purity of the classification, whereas recall represents completeness. The F1 score considers both precision and recall and harmonises the two measures.

Equation 5: F1 Score

$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

$$F1\,Score = \frac{2 \times Precision \times Recall}{Precision + Recall}$$

5.7.3 IoU (Intersection over Union)

In the context of semantic segmentation, Intersection over Union measures the fraction of overlap between the predicted mask and the ground truth.

Equation 6: Intersection over Union

$$IoU = \frac{Prediction\ mask \cap Ground\ truth\ mask}{Prediction\ mask \cup Ground\ truth\ mask}$$

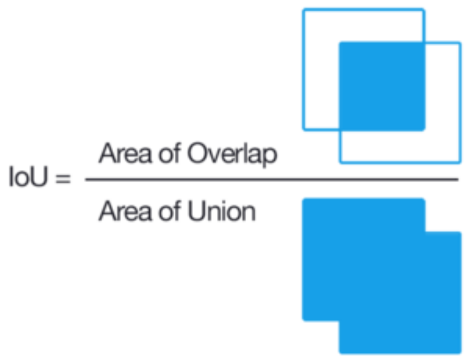


Figure 5-7 : Illustration of Intersection over Union. Figure by (Tiu, 2020).

In object detection, the predictions are made in terms of a bounding box and a class label.

The IoU in object detection is measured by the area of overlap between the actual and predicted bounding boxes. Furthermore, in object detection, precision and recall are calculated based on the IoU value for a given IoU threshold. The prediction becomes true or false based on the IoU threshold that determines how much area the predicted and actual bounding box should overlap. In the pipeline, IoU is not used to evaluate object detection because a better matrix named mean average precision is implemented, which uses the IoU value as its main parameter.

5.7.4 mAP (Mean Average Precision)

Mean Average Precision is a metric used to evaluate object detection specifically, and It is defined as the area under the precision-recall curve (Yohanandan, 2020). mAP value is based on several sub-matrices such as confusion matrix, IoU, precision and recall (mAP is not the average value of precision measures). The higher the mAP, the more accurate the detections.

5.8 Hyperparameter Tuning

Hyperparameters are variables whose values govern the learning process of a deep neural network. The learning algorithm uses hyperparameters when learning, but they are not part of the produced model. They can be recognised as top-level parameters that regulate the learning process. The values for the hyperparameters need to be set before starting the training process because hyperparameters are external to the model and cannot be changed during the training process.

Hyperparameter tuning is the process of selecting the appropriate hyperparameters for the model, and it is a highly iterative process. At the end of the training process, the model results are examined and based on the results, the hyperparameters are tuned, and the model is re-trained. The process continues till the resulting model performance is acceptable to the needs. There is no straightforward way to decide the hyperparameters correctly at the initial stage of developing a deep learning model. There are several advanced automated methods to speed up hyperparameter search (Feurer & Hutter, 2019), such as grid search or random search to find the initial learning rate and Bayesian optimisation. Implementing these techniques requires extensive knowledge, which is beyond the scope of this study. Manual search is used in the pipeline to find appropriate hyperparameters.

The following are the hyperparameters that were used in the pipeline.

5.8.1 Learning Rate

Learning rate defines how quickly the neural network updates its model parameters (weights, biases and kernel values). The learning process is slowed by selecting a low learning rate, yet it converges smoothly. Also, there is a risk of selecting a low learning rate because overfitting can occur. A faster learning rate accelerates learning, but if it is too large, the model will not converge. Due to this reason, the selection of a proper initial learning rate is crucial. A common practice when training a deep neural network is starting with a large learning rate and automatically reducing it when the model progresses using a learning rate decay function. Comparatively simple learning rate decay functions were used in this study. For semantic segmentation tasks, a function named step decay (learning rate is reduced by a constant amount after a certain number of epochs) is used. The default cosine learning rate decay function (starting with a large learning rate and rapidly decreasing to a minimum value before increasing again) is used for object detection. The initial learning rate was decided with a trial-and-error method.

5.8.2 Loss Function

The learning process tries to find the optimum model parameters that generalise the input data to a complex ruleset to make predictions on the unseen data. This generalisation process involves learning the weights and biases (or kernel values in CNNs) of all computing units, which was done by measuring the model's error. The model parameters are then adjusted to minimise the error. The loss function measures this error.

5.8.3 Optimisation Algorithm

The objective of the loss function is to provide a way to calculate the model error, which can then be used to update the model parameters. In order to update the model parameters, the value of the loss function needs to be minimised. Therefore, the objective of the optimisation algorithm is to minimise the loss. The definition of the loss function significantly affects the speed of training the model as well as the accuracy of the model.

Two commonly used optimisation algorithms are used in the deep learning pipeline used in this study.

Stochastic Gradient Descent (SGD): In this technique, the parameters are adjusted after each training sample. The training data is randomly sampled in each epoch before training. This technique is more memory-efficient and much faster than Batch Gradient Descent (model parameters are only updated once after considering all training data) for large-sized training datasets.

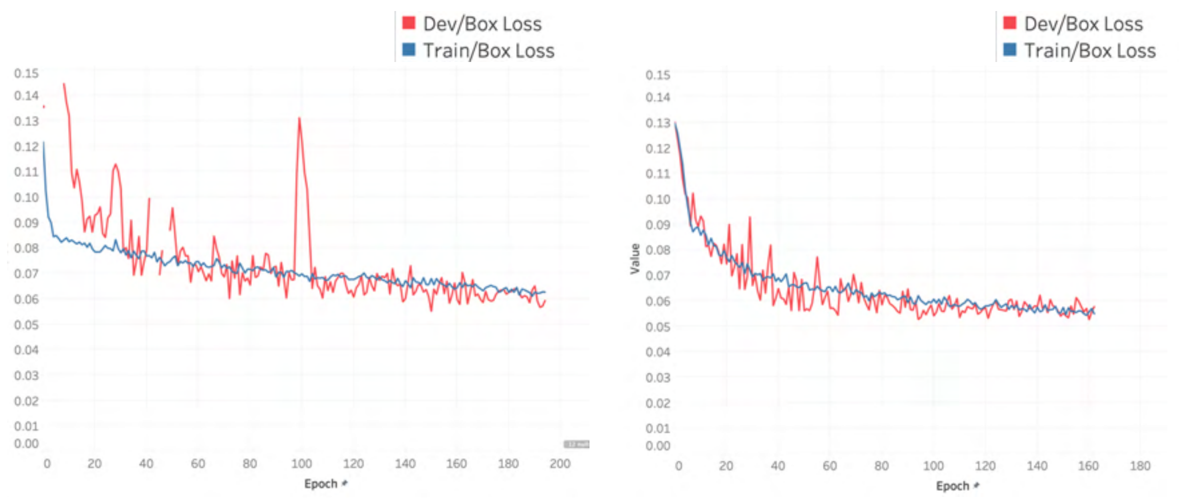
Adaptive Moment Estimation (Adam): This optimisation approach is an extension of the optimisation technique used in stochastic gradient and developed to update network weights during training. Unlike SGD, which maintains a single learning rate, Adam optimiser modifies the learning rate for each network weight individually. The process of Adam is to calculate an adaptive learning rate for each parameter in the model.

The Adam optimiser has various advantages over other optimisation algorithms. Adam is explicitly developed for deep learning. It has been adopted as a benchmark for deep learning papers and is suggested as the default optimisation approach. Furthermore, the algorithm has a faster running time, consumes less memory, and requires less tuning than any other optimisation technique. It also tries to compensate for the effects of selecting a large learning rate.

Considering the advantages of the Adam optimisation algorithm, it is selected as the default optimiser in the pipeline. It shows a high level of performance in all deep learning architectures used in this study except for YOLOv5. In YOLO, SGD shows superior performance where Adam optimiser suffered with high fluctuations of the loss value between the performance of the Train set and Dev set (figure 5-8). This was an indicator that the model was not converging and generalising well. Compared to Adam, SGD generalises the data better, and performance improvement has been observed in YOLO.

5.8.4 Number of Epochs

It is the number of iterations the optimisation algorithm runs over the whole training dataset. The number of epochs was decided for each model based on observing the convergence of the model. The error in the Training set and Dev set indicates convergence. The number of epochs was increased in case of errors indicating the possibility of improving the model by increasing the number of iterations. If both Train and Dev set errors have reached their minimum and maintain their minimum at successive epochs, it indicates convergence. If the errors are still decreasing, the model can be improved by increasing the number of iterations.



(a) (b)

Figure 5-8 : Variation of loss values in YOLOv5 in extracting tree symbols (a) with Adam optimiser (b) with SGD optimiser.

5.8.5 Mini Batch Size

Mini batch size denotes the number of samples taken at a time during the training process. Since deep learning is a computationally intensive process, it is required to manage memory efficiently. Therefore, dividing the samples into mini-batches and feeding them to the network one at a time is one way to reduce the computational requirements.

During the workflow, the mini-batch size is decided by the size of the training images and by the demand of the deep learning architecture. Due to the efficient model architecture of YOLO, it is much faster compared to semantic segmentation architectures, regardless of the number of model parameters.

Table 5-2 : Number of model parameters in different architectures used in the study.

Task	Architecture	Number of Model Parameters
Semantic segmentation	UNet	45.6M
	InceptionResNetV2	54M
	ResNet	23M
Object Detection	YOLOv5-m	21.2M
	YOLOv5-L	46.5M
	YOLOv5-X	86.7M

In this study, the implemented deep learning pipeline was deployed in the Google Colab cloud computing environment for training and testing the models. In the free tier, Google Colab provides a virtual machine with 16gb of RAM (Random Access Memory) and 16gb of GPU memory. Therefore, the training image size and mini-batch size were selected to fit the memory availability.

There are several other essential hyperparameters in a deep learning model. Hyperparameters such as the choice of activation function, number of hidden layers, number of computations units in each layer, kernel size, and pooling size are based on the architecture of the deep neural network. Changing these parameters will change the neural network architecture. As well-established pre-programmed CNN architectures are used in this study, changing the aforementioned hyperparameters was not required.

6 RESULTS AND DISCUSSION

After the training phase, the test set data were used to evaluate the model's performance qualitatively and quantitatively using the evaluation metrics implemented in the pipeline. Successfully training a deep neural network is an iterative process. As explained in the previous chapter, hyperparameter tuning and solving bias and variance errors required repeatedly running the training process after tweaking the settings. The results presented in this section of the thesis are based on the best-obtained models after many iterations.

In this study, it is proposed to evaluate deep learning architectures to extract six symbol classes from historical maps. However, due to time limitations and the more extensive scope of the study, the models were trained and evaluated only for four classes, i.e. small buildings, lakes, rivers, and forests. A similar approach can be used to train and evaluate extraction for building complex and meadow classes.

6.1 Results of Semantic Segmentation

The models for extracting small buildings were trained with the highest number of training samples compared to other classes. All three models indicated exceptional performance when classifying small buildings in terms of accuracy. Even though all three models show nearly 100% pixel accuracy, the F1 score and IoU are better measures because the class imbalance in the training data is not affecting F1 score and IoU. The highest F1 Score and IoU were obtained by the ResNet model outperforming other architectures only with a slight margin in terms of accuracy.

No significant overfitting is shown in the results (figure 6-2) for all three models. In the Inception-ResNet model, there is a slight deviation of the Dev set performance and Train set performance, indicating overfitting after the 35th epoch. This separation can be neglected because the model achieved convergence in the early stage of training and maintained it for an extended period of epochs.

However, when considering the training performance, ResNet could complete it in just over 30 minutes, far sooner than its competitors. This is due to its superior architectural design that reduces complexity while maintaining good performance. Furthermore, because ResNet is lightweight in terms of memory usage, it was possible to use a minibatch size of 64. It means at each step in each epoch, the pipeline feeds 64 samples to the neural network, effectively reducing the time to complete a single epoch and speeding up the training.

The cartographic symbol for small buildings class in the maps is visually simple. Therefore, it can be assumed that the deep learning architectures used here achieved high performance in extracting small building symbols due to the simplicity of the task and primarily due to having a good training dataset.

Few misclassifications appeared in the results and were identified with visual analysis. The visual analysis was performed by comparing the prediction outputs of the Test set. It was observed that all three models have some difficulty distinguishing between the symbols for rocks and small buildings. However, ResNet architecture showed the lowest number of misclassifications in contrast

InceptionResNet indicating the most. It was also observed that some building complexes are misclassified as small buildings in all three models. However, this error was most prominent in the ResNet model and minor occurrences in the InceptionResNet model.

After observing the misclassifications, it was found that there are very few images in the training data that contain rock symbols. An approach to solve the misclassification is by including more samples of rocks in the training data and making them true negatives. To solve the misclassifications between small buildings and building complexes, a multi-class semantic segmentation approach can be taken where single model is trained for both classes.

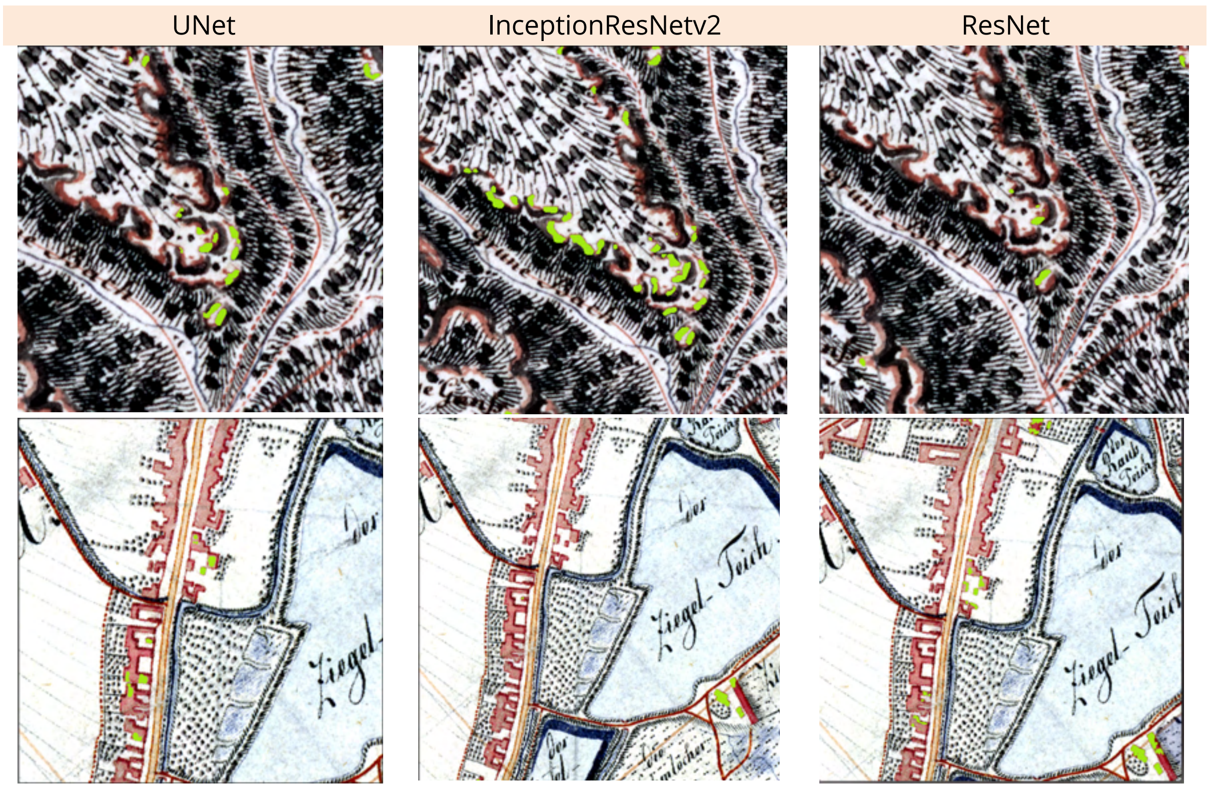


Figure 6-1 : Misclassifications of the buildings class.

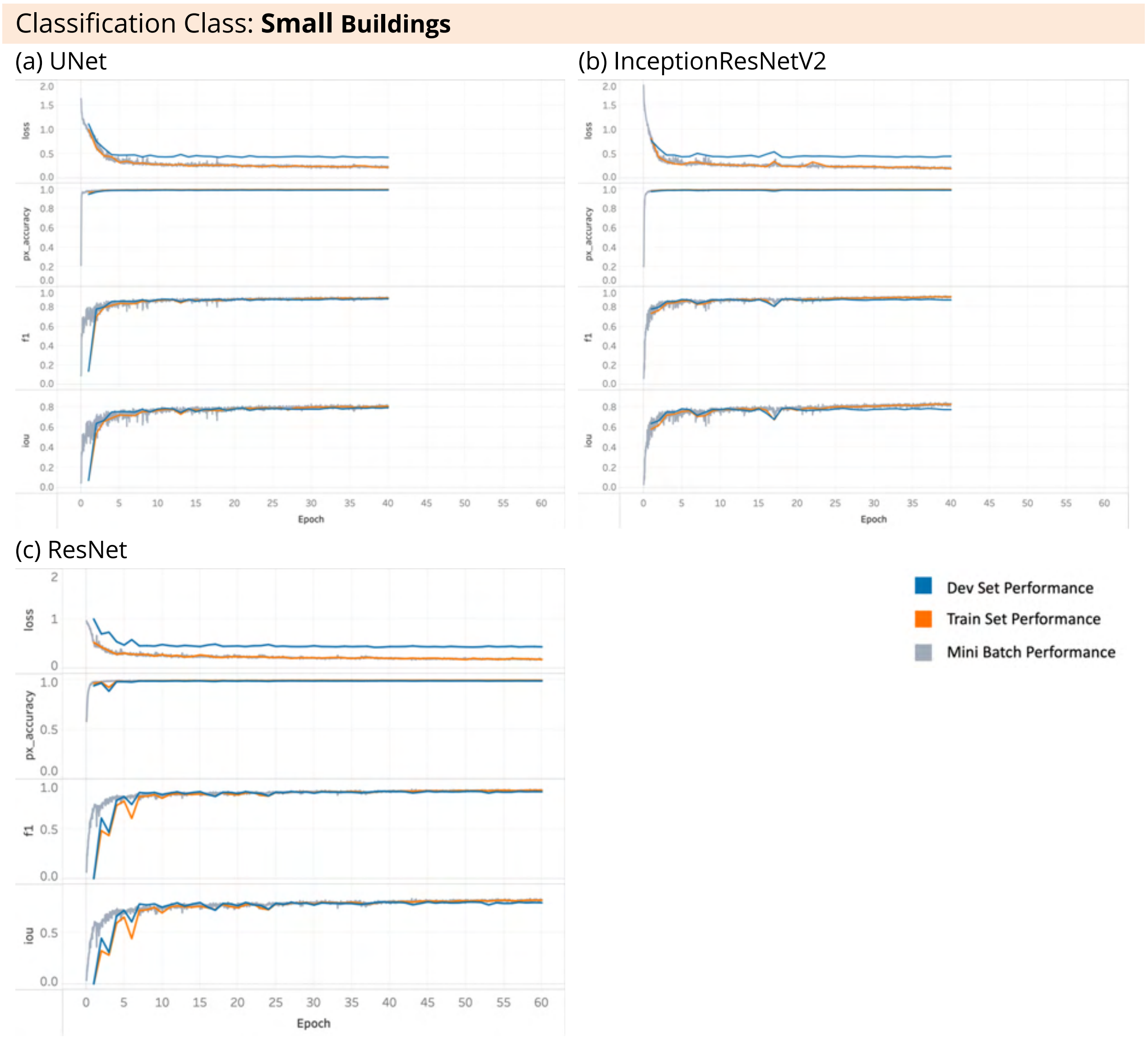


Figure 6-2 : Visualisation of the model performance in extracting small buildings.

Table 6-1 : Model and accuracy information: **Small Buildings.**

Architecture	**UNet**	**InceptionRes-Netv2**	**ResNet**
	Input Parameters		
Input image size (px)	256	256	256
Number of images in Train Set	1664	1664	1664
Number of images in Dev Set	608	608	608
	Hyper Parameters		
Number of epochs	40	40	60
Mini batch size	32	32	64
Initial learning rate	0.001	0.001	0.003
Learning rate decay	Step	Step	Reduce on Plateau
Optimiser	Adam	Adam	Adam
Loss function	IoU+BCE loss	IoU+BCE loss	IoU loss

Accuracy Parameters			
Best epoch	25	25	**40**
Pixel accuracy	0.985	0.984	***0.985***
F1-Score	0.878	0.876	***0.884***
IoU	0.782	0.779	***0.792***
Performance Parameters			
Time to compute one epoch	00:02:21	00:01:40	**00:00:39**
Total time to train	01:34:15	01:06:47	**00:38:31**

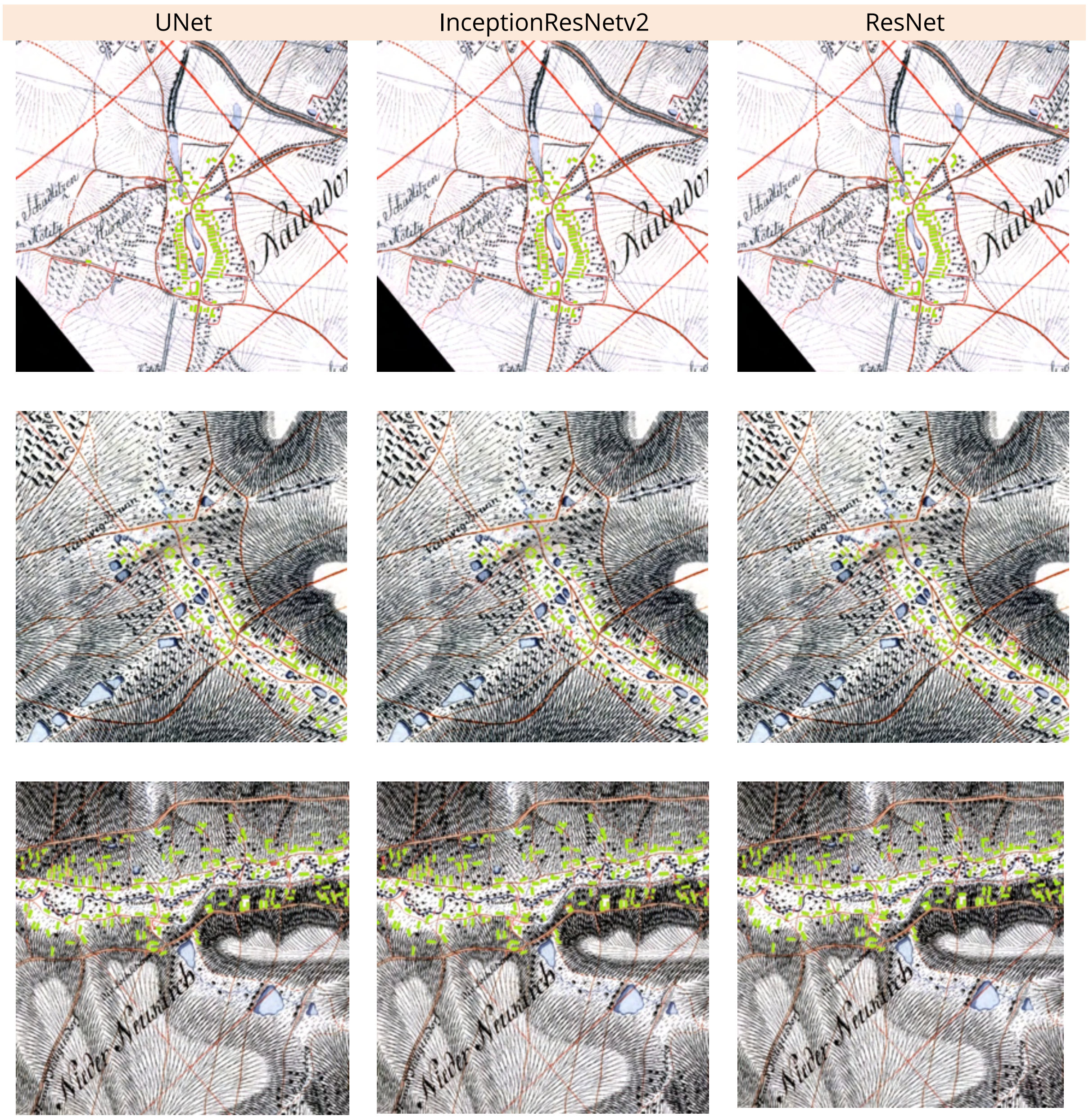


Figure 6-3 : Extraction results of small buildings.

The classification of lakes and rivers shows a different scenario compared to the buildings' results. The computation was intensive due to the larger image size used to accommodate the significant size variation in lake and river symbols. Image size of 512px was used in both cases, making the training images four times larger than the training images of buildings. So more computation time was required even though rivers and lake classes use a smaller number of training data. In both river and lake classes, the ResNet architecture shows superior performance in terms of accuracy matrices and performance time (table 6-2 and table 6-3). By visual analysis of the test set results, it was confirmed that the ResNet model produced significantly better results in extracting lakes and rivers. There were only a few misclassifications where parts of large rivers were classified as lakes and vice versa. In UNet and InceptionResNet, the misclassification rate was higher.

The poor performance in UNet and ResNet can be explained by visualising the accuracy matrices. When classifying lakes, UNet shows the worst performance indicated by high fluctuations of the loss, IoU and F1 score (figure 6-5) that indicate difficulty for the model to converge. However, all three models show significant separation between the Train set and Dev set, which indicates overfitting. This is explained by the fact that lakes and river classes have significantly less training data. The figure 6-5 and figure 6-7 show that all three models fit well into the Train set but cannot fit into the untrained Dev set. The significantly low number of training data is at the heart of this problem. To overcome this problem, creating and including more training data is required, which is a highly time-consuming task. In addition, variance reduction techniques such as Drop Out can be used, but it requires modification of the deep learning architecture, which is beyond the scope of this study.

By analysing the model's performance and visualising the results, it can be confidently stated that the ResNet model provides better performance even in the challenging circumstances of insufficient training data. Furthermore, the performance of all three models will significantly increase when more training data is included.

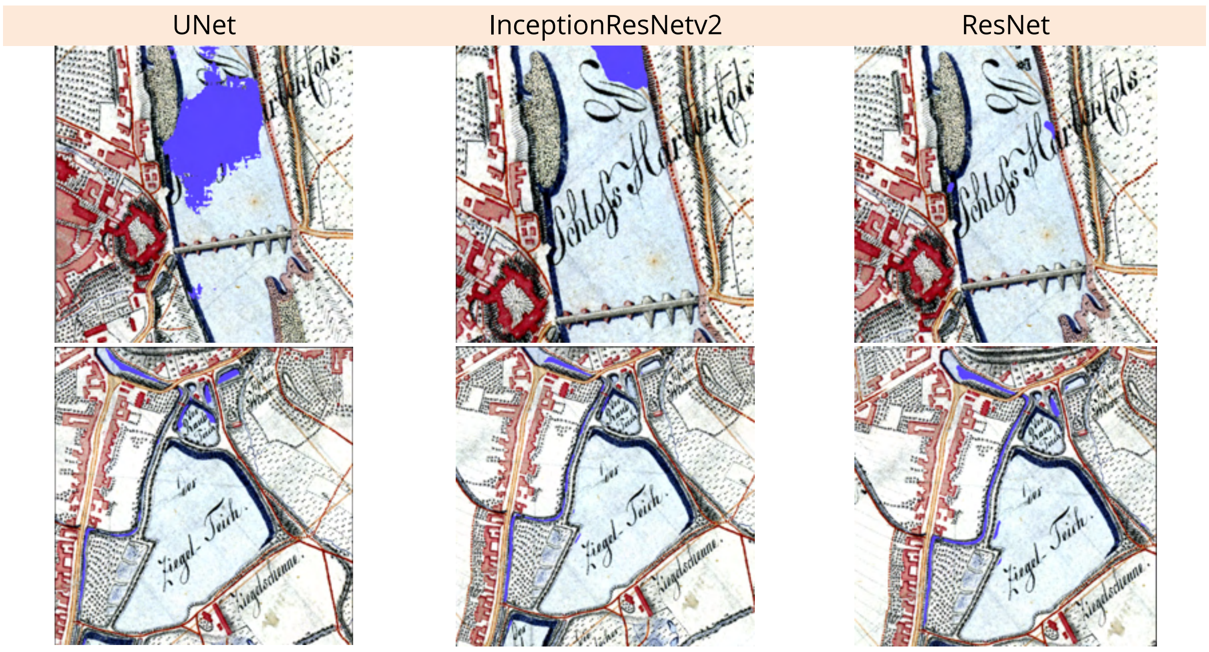


Figure 6-4 : Misclassifications in Lakes (top) and River (bottom) classes.

Classification Class: **Lakes**

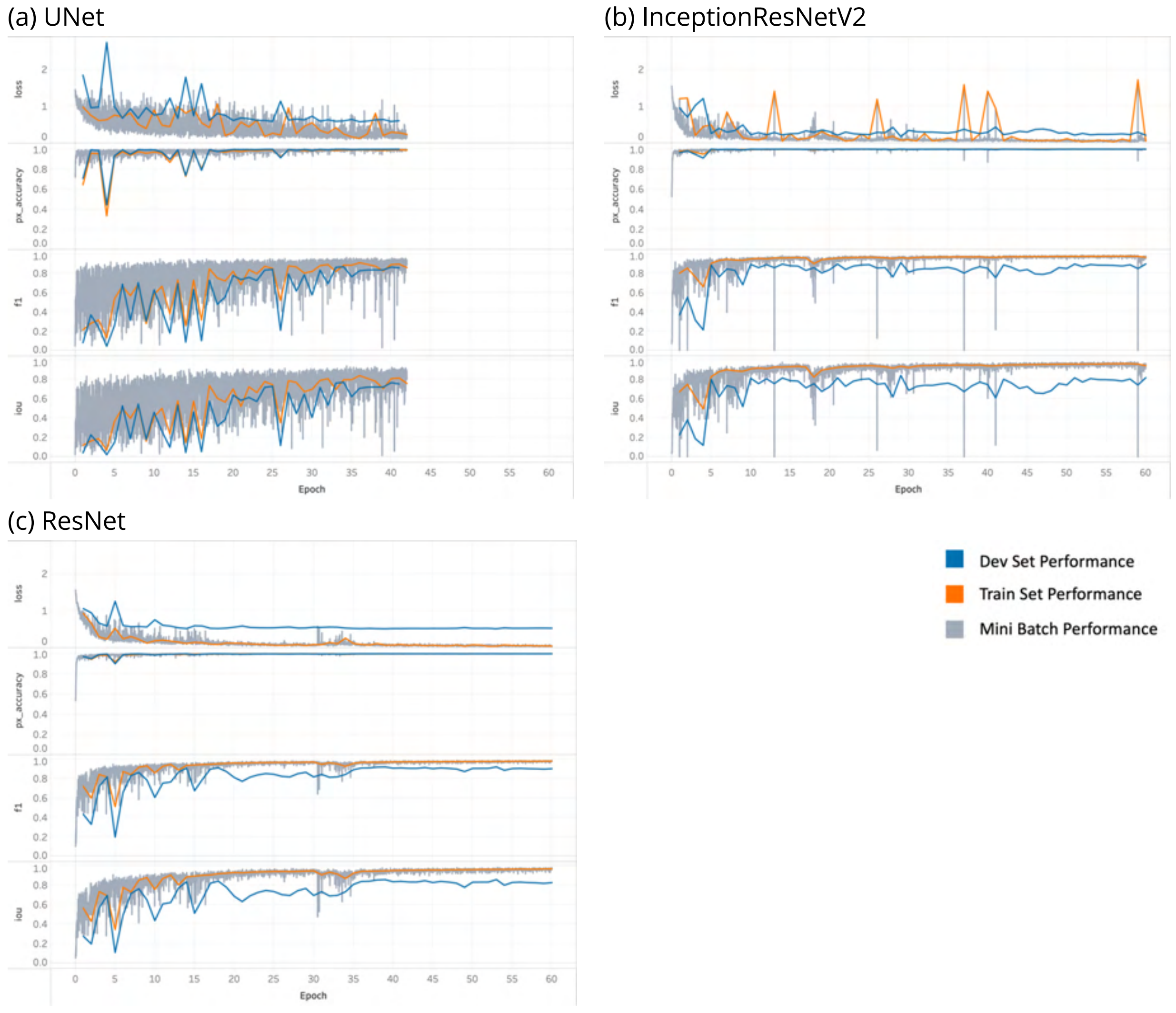


Figure 6-5 : Visualisation of the model performance in extracting lakes.

Table 6-2 : Model and accuracy information: **Lakes class.**

Architecture	**UNet**	**InceptionRes-Netv2**	**ResNet**
Input Parameters			
Input image size (px)	512	512	512
Number of images in Train Set	800	800	800
Number of images in Dev Set	104	104	104
Hyper Parameters			
Number of epochs	*41	60	60
Mini batch size	8	12	16
Initial learning rate	0.0007	0.001	0.0007
Learning rate decay	Step	Step	Step
Optimiser	Adam	Adam	Adam
Loss function	IoU + BCE	IoU + BCE	IoU + BCE
Accuracy Parameters			
Best epoch	40	55	40

Pixel accuracy	0.996	0.996	0.997
F1-Score	0.865	0.885	0.909
IoU	0.763	0.794	0.833
Performance Parameters			
Time to compute one epoch	00:03:30	00:02:38	00:01:41
Total time to train	03:30:08	02:38:00	01:41:18

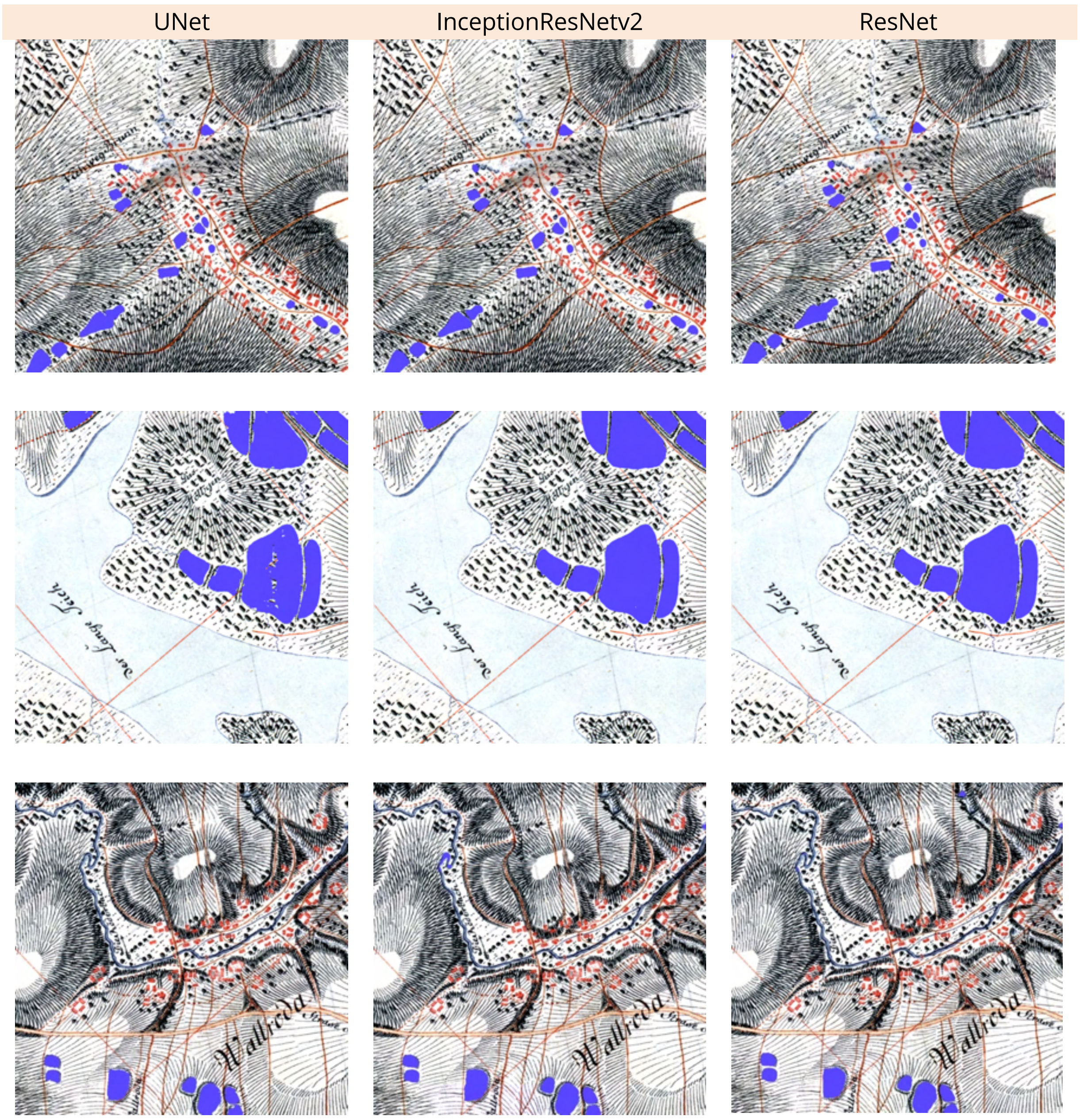


Figure 6-6 : Extraction results of lakes.

Classification Class: **Rivers**

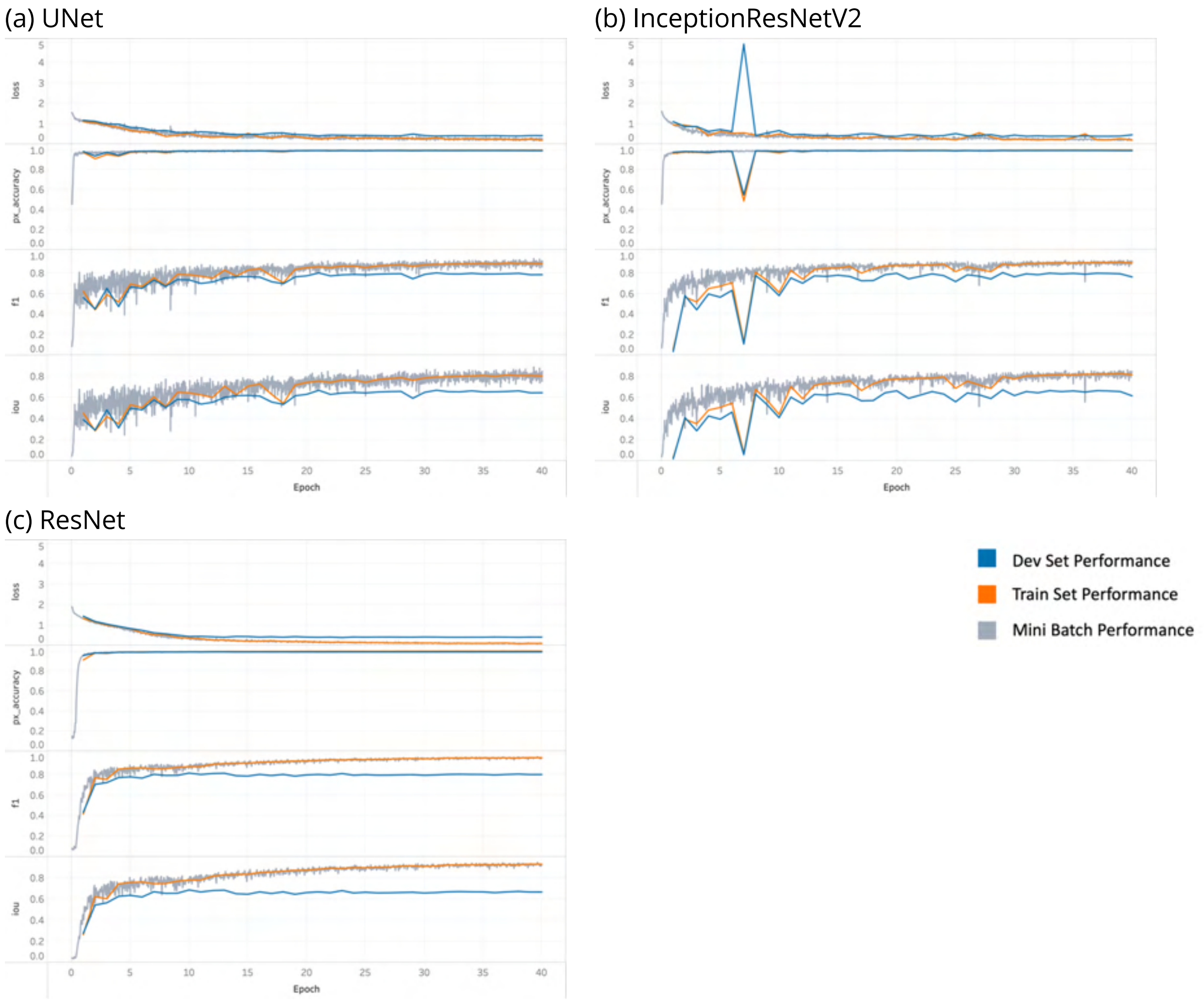
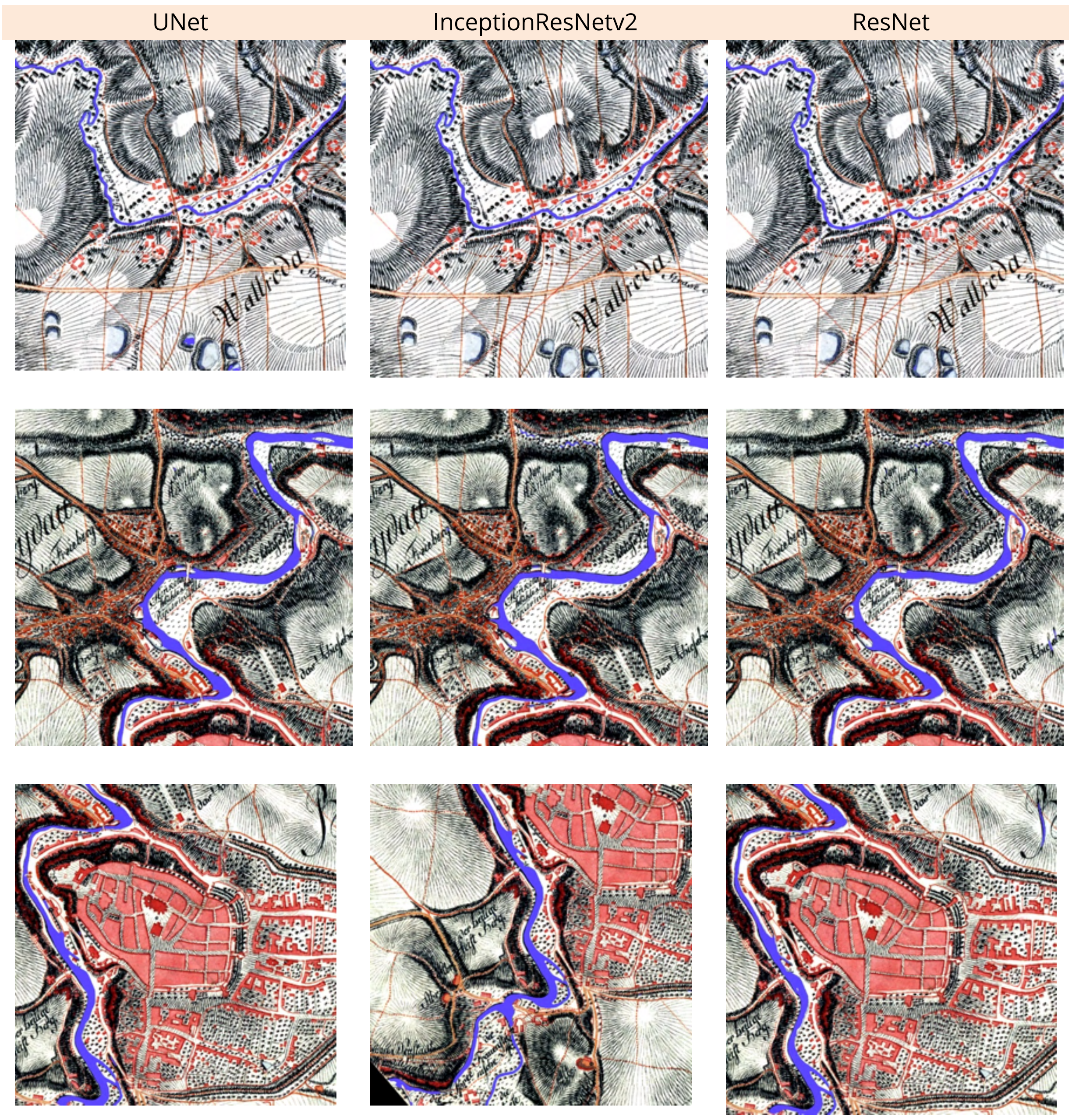


Figure 6-7 : Visualisation of the model performance in extracting rivers.

Table 6-3 : Model and accuracy information: River class.

Architecture	**UNet**	**InceptionRes-Netv2**	**ResNet**
Input Parameters			
Input image size (px)	512	512	512
Number of images in Train Set	352	352	352
Number of images in Dev Set	88	88	88
Hyper Parameters			
Number of epochs	40	40	40
Mini batch size	8	12	16
Initial learning rate	0.0005	0.001	0.0005
Learning rate decay	Step	Step	Step
Optimiser	Adam	Adam	Adam
Loss function	IoU+BCE	IoU+BCE	IoU+BCE
Accuracy Parameters			
Best epoch	35	20	10
Pixel accuracy	0.987	0.988	0.988

F1-Score	0.789	0.796	0.813
IoU	0.651	0.661	0.685
Performance Parameters			
Time to compute one epoch	00:02:07	00:01:19	00:00:45
Total time to train	01:24:42	00:53:00	00:30:01



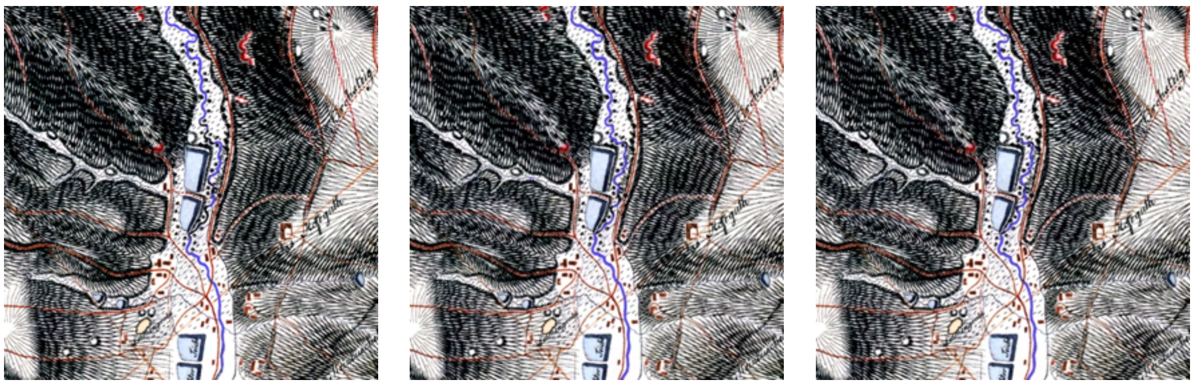


Figure 6-8 : Extraction results of rivers.

6.2 Results of Object Detection

An object detection approach is proposed in this study to extract forest areas from the map scans. The first step is to detect the individual tree symbols and extract the centre coordinates, which was done using deep learning. The second step is to use a clustering algorithm to cluster the extracted points and polygonise the cluster to extract the forest area.

The family of algorithms used to extract tree symbols is YOLO. YOLOv5_m, YOLOv5_l, and YOLOv5_x are evaluated in the task. The significant architectural difference between these three models is their size. YOLOv5_l has the smallest model parameters, whereas YOLOv5_x has the largest.

The object detection performance of all three architectures shows remarkable accuracy and faster training time. To train the models, 213 images were used in the Train set with approximately 80 annotations per image, whereas 31 images were used in Dev set with 86 annotations per image. It creates a total number of 17102 training data labels and 2661 evaluation labels. This large amount of training data is the key to successfully training the models. Training data preparation for object detection is much simpler and less time-consuming than creating a training dataset for semantic segmentation. To prepare a semantic segmentation dataset, the object's boundaries need to be accurately marked as a polygon which is then converted to a binary mask. However, in object detection, the training data is prepared by making the objects in the images with a simple bounding box.

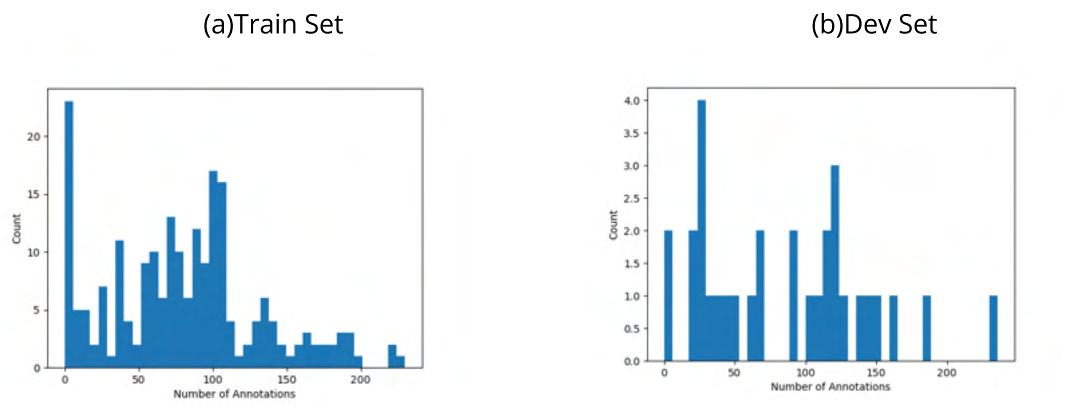


Figure 6-9 : Distribution according to the number of annotations in the training data of the forest class.

Compared to semantic segmentation, object detection is computationally less expensive. Each model took less than half an hour to successfully build a fully trained neural network to extract tree symbols from the maps. All the models converge well to the data, and there is no indication of overfitting (figure 6-11). Regarding accuracy matrices, all three models are on par with each other. When considering the accuracy values, the YOLOv5_l model is marginally ahead of its competitors. On the other hand, YOLOv5_m has fewer model parameters to train, and it was a few minutes faster to complete training.

When visually observing the test set results, it was found that there is a negligible number of misclassifications in all the models. For example, the meadow symbol is misclassified in a few locations as tree symbols in YOLOv5_m and YOLOv5_x models, but the misclassifications were even less in YOLOv5_l. However, the error of omission is less in the YOLOv5_l model. Both scenarios can be seen in the figure 6-10.

YOLOv5_x is a larger model (with deeper neural network architecture) than the rest. As it has more model parameters to learn, it can learn complex data patterns. The tree symbol in the Saxony mileage sheet has many variations in shape, colour, and size. YOLOv5_x can leverage the availability of a high number of parameters to learn such varying and complex patterns of data. Due to this reason, YOLOv5_x shows overall superiority compared to the other models.

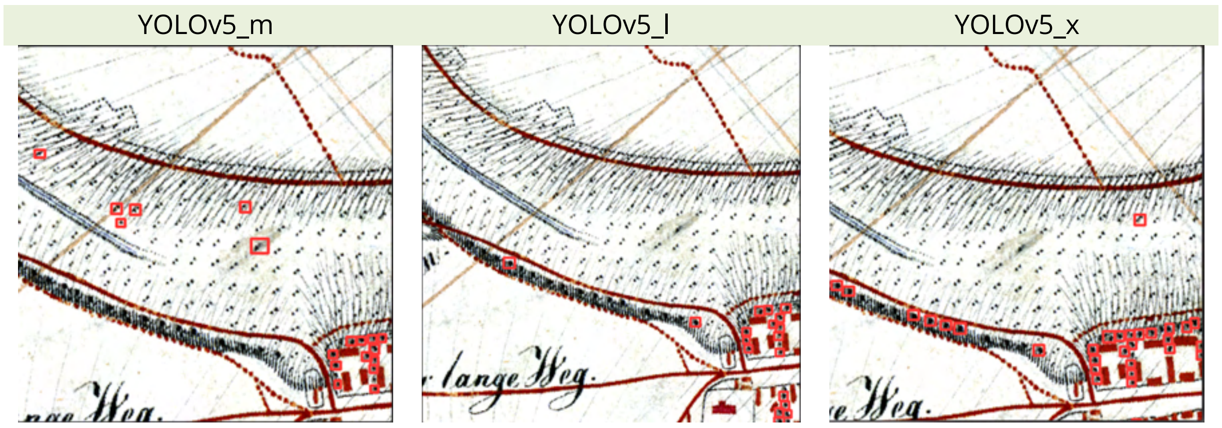


Figure 6-10 : Misclassification in extracting tree symbols.

Classification Class: **Forest**

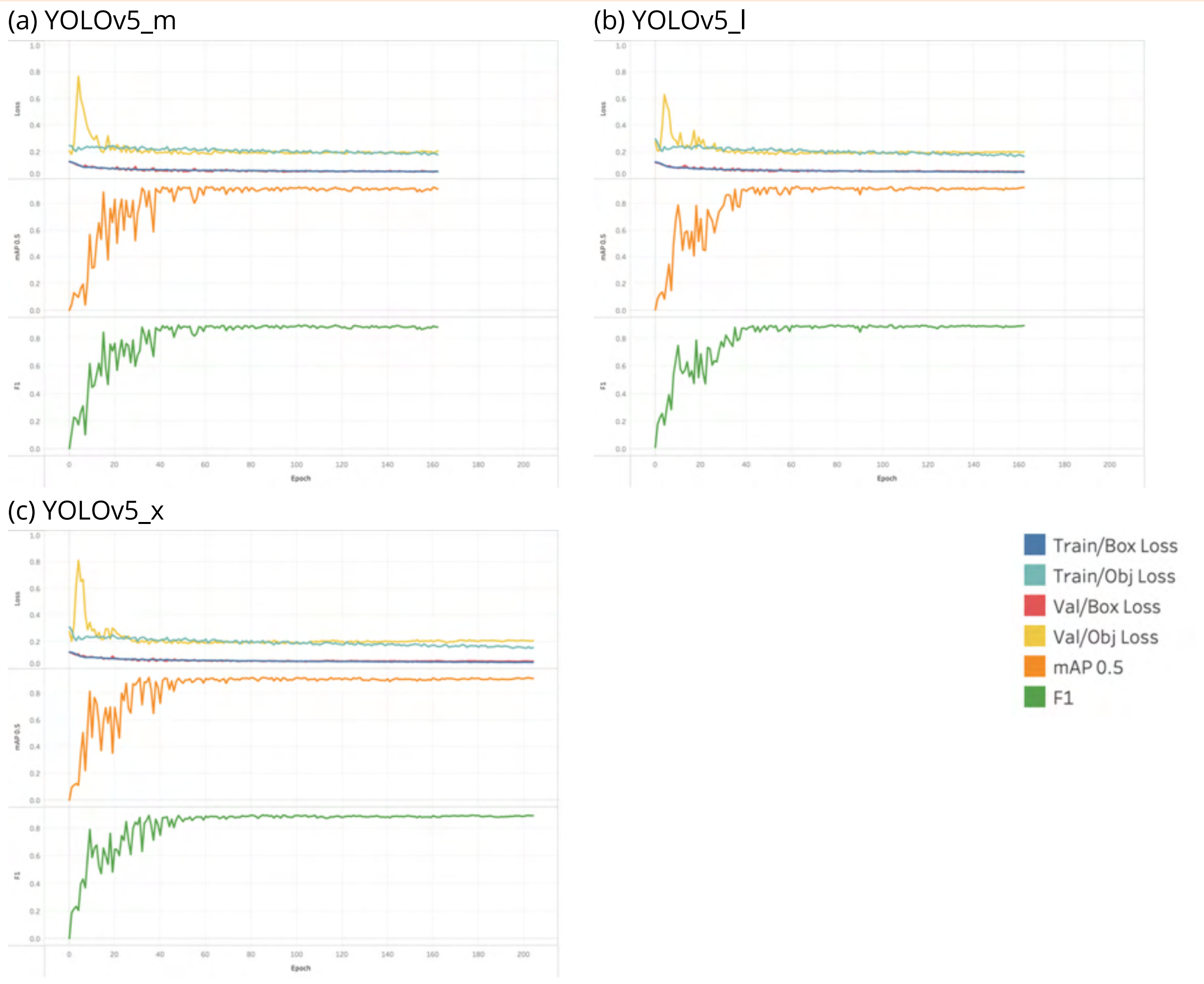


Figure 6-11 : Visualisation of the model performance in extracting tree symbols.

Table 6-4 : Model and accuracy information: **Forest class.**

Architecture	**YOLOv5_m**	**YOLOv5_l**	**YOLOv5_x**
Input Parameters			
Input image size (px)	416	416	416
Number of images in Train Set	213	213	213
Number of images in Dev Set	31	31	31
Average annotations per image in Train Set	80	80	80
Average annotations per image in Dev Set	86	86	86
Total annotations in Train Set	17102	17102	17102
Total annotations in Dev Set	2661	2661	2661
Hyper Parameters			
Number of epochs	200	200	200
Mini batch size	32	32	32
Initial learning rate	0.01	0.01	0.01
Learning rate decay	Constant	Constant	Constant
Optimiser	SGD	SGD	SGD
Loss function	CIoU	CIoU	CIoU
Early stopping after (epochs)	100	100	100
Accuracy Parameters			

Best epoch	100	119	101
F1-Score	0.891	0.896	0.894
mAP @0.5	0.917	0.92	0.916
Performance Parameters			
Time to compute one epoch	00:00:27	00:00:31	00:00:38
Total time to train	00:17:46	00:20:53	00:25:31

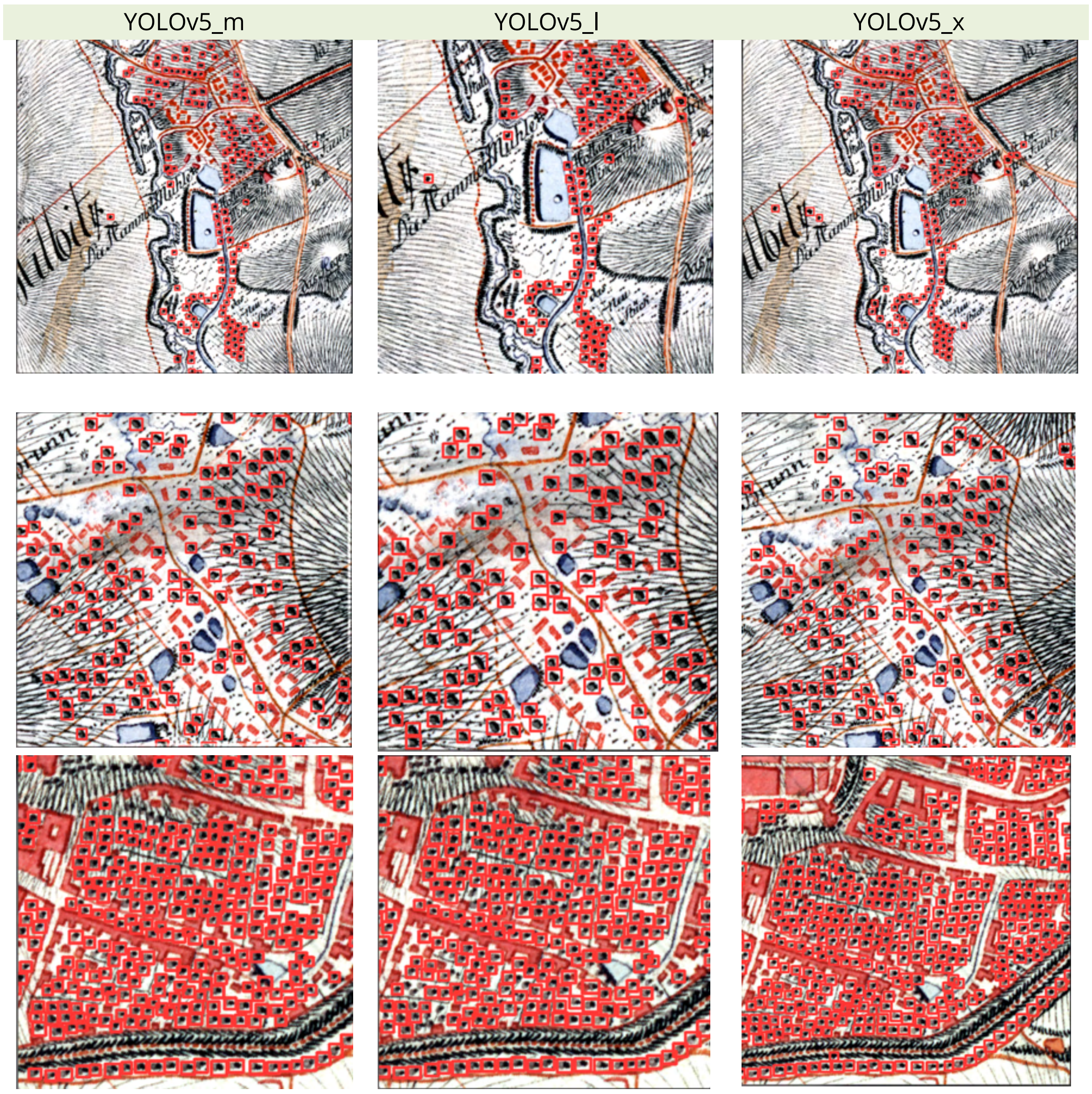


Figure 6-12 : Extraction results of individual trees.

The next step in object detection workflow the extract forest areas from the map is to perform clustering based on the extracted locations of the tree symbols. As a proof of concept, a clustering workflow is created for point symbols to convert into area polygons.

1. Extracting centre coordinates of each detected bounding box
2. Perform spatial clustering using the DBSCAN algorithm. The clustering is controlled by two parameters. 1. maximum distance between two samples for one to be considered as in the neighbourhood of the other. 2. minimum number of samples in a cluster
3. converting the clusters to polygons using a concave hull

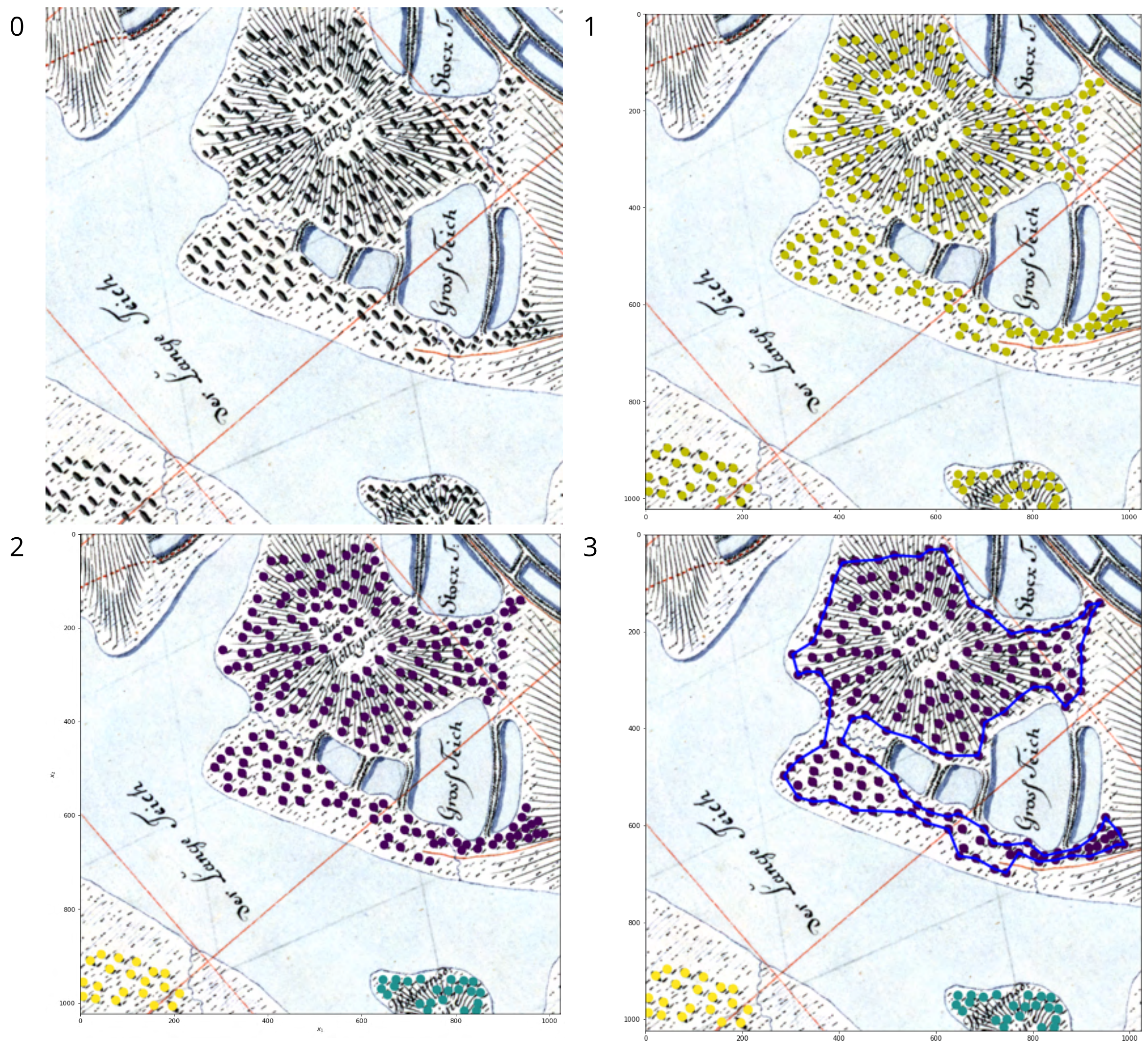


Figure 6-12 : Figure: vectorising workflow to convert detected tree symbols to polygons. 0. Original image 1. Centre coordinates extraction 2. DBSCAN clustering 3. Converting cluster to polygon.

This workflow has many significant limitations, and evaluation methods are not implemented in this study. The first limitation is calculating DBSCAN parameters. The distance between tree symbols varies throughout map scans. The second is separating tree symbols belonging to the forest class and individual trees. These questions need to be answered to effectively complete the workflow of forest area extraction, which is left for future research due to time limitations.

6.3 Limitations and Future Work

6.3.1 Lack of Training Data

Data are the foundation of every deep learning project, so the issue of data scarcity is crucial. The poor performance in many deep learning models is due to the small size of the training dataset. The lack of training data is the main limitation of this study as well. It was identified that object detection models for detecting tree symbols and the semantic segmentation model for building classification works remarkably well compared to the other classification models obtained in this study. The better performance in tree detection and building classification is mainly due to having a high number of training data. However, creating such a large training dataset is tedious and costly.

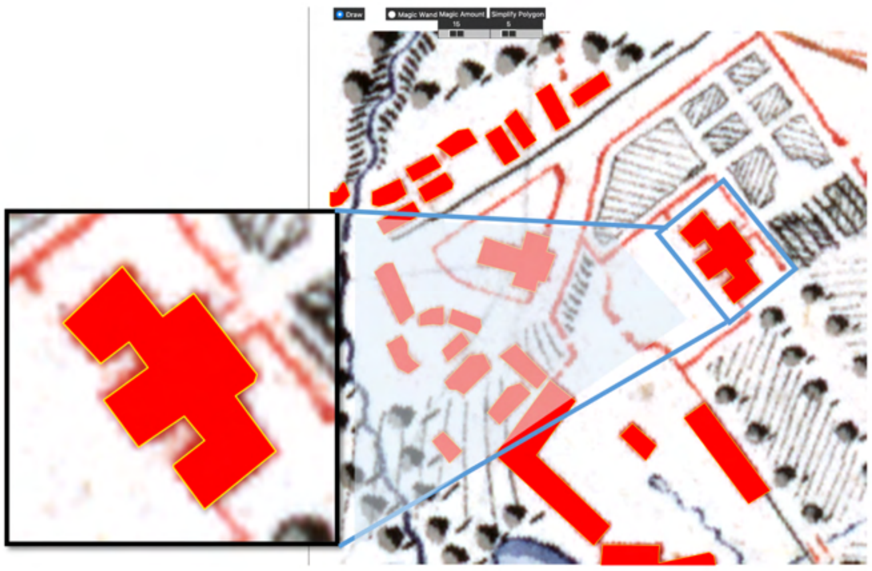
To speed up the process of creating training data for this study, simple yet effective software is developed where a user can load unlabelled training images and manually generate training data by annotating. The developed software is at its early stage with several limitations. The geotagged map scans are first pre-processed in GIS software to extract the candidate training patches. However, there is no link between the GIS process and the annotation software. Currently, it runs by executing several pieces of code independently. As the next step, the GIS process and annotating tool can be harmonised to make it more efficient and effective. Another improvement to the annotating tool is to develop it further to work with multiple users working on a single task.

Crowdsourcing is another approach to creating training data efficiently at a low cost (Sheng & Zhang, 2019). Crowdsourcing is not new in the geospatial domain. OpenStreetMap is the most outstanding example that uses crowdsourced volunteered geographic information (VGI) to build a global geospatial database. In the context of historical maps, the milage sheets of Saxony historical map series, which is the same dataset used in this study was successfully geo-referenced using a crowdsourced approach (Mendt, 2014). However, obtaining error-free crowdsourced training data can be challenging, making it more difficult in a cartographic context as the users need to have a specific understanding of the maps they have to work with.

6.3.2 Lack of Computational Power

Another limitation of this study is the lack of computational power. Training deep learning models demands a lot of computational power, which cannot be fulfilled with a consumer-grade computer. Cloud computing is one solution to these limitations. In this study, the free tier of Google Colab cloud computing service is used, which comes with resource limitations such as limited memory, GPU and time limitations. It was sufficient for the scope of this study, but higher computation power is necessary when bringing this research further to evaluate more sophisticated and new deep learning architectures. As a solution, there are several service providers, such as Google, Amazon, and Microsoft, that provide subscription-based cloud computing services that offer more computational power for AI applications.

6.3.3 Digitisation Error



[✓ Draw] Magic Wand

Magic Amount 15

Simplify Polygon 5

Figure 6-13 : Digitisation error.

This study uses manual digitisation to create semantic segmentation training data. However, it is impossible to perform perfect pixel-level digitisation manually. The effect of this error when training deep learning models is unknown. It can be further examined by synthetically creating training data and evaluating the resulting deep learning models.

7 CONCLUSION

This study demonstrates a deep learning based workflow to extract the data from historical maps. It began with investigating different deep learning architectures and feature extraction strategies that have promising capabilities to employ in digital map processing tasks. A thorough deep learning experiment on comparing different architectures was presented in this study using historical maps of the Saxonian topographic survey. The main goal of the experiment was to find the best performing deep learning architectures for effectively extracting geographic features from historical maps.

A pipeline was developed to train deep learning models, evaluate the performance, and visualise the results to achieve this goal. Simplicity was kept in mind when designing the pipeline so cartographers could quickly adapt it to deep learning research in digital map processing. Several deep learning architectures were compared using the pipeline to extract areal features from historical maps. The results were presented, discussing its strengths, challenges and limitations in extracting area features. To address the issue of lack of training data, a tool was developed to create labelled training data rapidly.

The experiment showcased that selection of a proper deep learning architecture has a significant influence in terms of performance and accuracy, which is an impactful factor when deploying the models in real-world applications. However, this study also demonstrates that solving the fundamental challenges of deep learning, such as scarcity of training data, should be addressed first to unlock the technology's full potential.

With the findings of this study, It can be concluded that deep learning is the technology that can make a change in digital map processing to unlock the vast amount of data hidden in historical map archives.

REFERENCES

Albawi, S., Mohammed, T. A., & Al-Zawi, S. (2017). Understanding of a convolutional neural network. *2017 International Conference on Engineering and Technology (ICET)*, 1–6. https://doi.org/10.1109/ICEngTechnol.2017.8308186

Allord, G. J., Fishburn, K. A., & Walter, J. L. (2014). Standard for the U.S. Geological Survey Historical Topographic Map Collection. In *Standard for the U.S. Geological Survey Historical Topographic Map Collection* (USGS Numbered Series No. 11-B3; Techniques and Methods, Vols. 11-B3, p. 20). U.S. Geological Survey. https://doi.org/10.3133/tm11B03

Alzubaidi, L., Zhang, J., Humaidi, A. J., Al-Dujaili, A., Duan, Y., Al-Shamma, O., Santamaría, J., Fadhel, M. A., Al-Amidie, M., & Farhan, L. (2021). Review of deep learning: Concepts, CNN architectures, challenges, applications, future directions. *Journal of Big Data*, *8*(1), 53. https://doi.org/10.1186/s40537-021-00444-8

Bajcsy, P. (n.d.). *Automatic Extraction Of Isocontours From Historical Maps*.

Basu, M. (2002). Gaussian-based edge-detection methods-a survey. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, *32*(3), 252–260. https://doi.org/10.1109/TSMCC.2002.804448

Brunelli, R. (2009). *Template Matching Techniques in Computer Vision: Theory and Practice*. Wiley Publishing.

Budig, B. (2018). *Extracting spatial information from historical maps: Algorithms and interaction*. Würzburg University Press.

Chen, J. (2019, December 22). Weak Supervision (Part I). *Medium*. https://medium.com/@jameschen_78678/weak-supervision-part-i-aca5b810af72

Chiang, Y.-Y., Duan, W., Leyk, S., Uhl, J. H., & Knoblock, C. A. (2020a). Training Deep Learning Models for Geographic Feature Recognition from Historical Maps. In Y.-Y. Chiang, W. Duan, S. Leyk, J. H. Uhl, & C. A. Knoblock (Eds.), *Using Historical Maps in Scientific Studies: Applications, Challenges, and Best Practices* (pp. 65–98). Springer International Publishing. https://doi.org/10.1007/978-3-319-66908-3_4

Chiang, Y.-Y., Duan, W., Leyk, S., Uhl, J. H., & Knoblock, C. A. (2020b). *Using Historical Maps in Scientific Studies: Applications, Challenges, and Best Practices*. Springer International Publishing. https://doi.org/10.1007/978-3-319-66908-3

Chiang, Y.-Y., & Knoblock, C. A. (2013). A general approach for extracting road vector data from raster maps. *International Journal on Document Analysis and Recognition (IJDAR)*, *16*(1), 55–81. https://doi.org/10.1007/s10032-011-0177-1

Chiang, Y.-Y., Leyk, S., & Knoblock, C. A. (2014). A Survey of Digital Map Processing Techniques. *ACM Computing Surveys*, *47*(1), 1:1-1:44. https://doi.org/10.1145/2557423

Dertat, A. (2017, October 9). *Applied Deep Learning - Part 1: Artificial Neural Networks*. Medium. https://towardsdatascience.com/applied-deep-learning-part-1-artificial-neural-networks-d7834f67a4f6

Ding, W., & Zhang, L. (2021). Building Detection in Remote Sensing Image Based on Improved YOLOV5. *2021 17th International Conference on Computational Intelligence and Security (CIS)*, 133–136. https://doi.org/10.1109/CIS54983.2021.00036

Douglas, D. H., & Peucker, T. K. (1973). Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. *Cartographica: The International Journal for Geographic Information and Geovisualization*, *10*(2), 112–122. https://doi.org/10.3138/FM57-6770-U75U-7727

Drolias, G. C., & Tziokas, N. (2020). Building Footprint Extraction from Historic Maps utilizing Automatic Vectorisation Methods in Open Source GIS Software. *Automatic Vectorisation of Historical Maps: International Workshop Organized by the ICA Commission on Cartographic Heritage into the Digital. Budapest – 13 March, 2020*, 9–17. https://doi.org/10.21862/avhm2020.01

Du, J. (2018). Understanding of Object Detection Based on CNN Family and YOLO. *Journal of Physics: Conference Series*, *1004*, 012029. https://doi.org/10.1088/1742-6596/1004/1/012029

Ebi, N., Lauterbach, B., & Anheier, W. (1994). An image analysis system for automatic data acquisition from colored scanned maps. *Machine Vision and Applications*, *7*(3), 148–164. https://doi.org/10.1007/BF01211660

Ekim, B., Sertel, E., & Kabadayı, M. E. (2021). Automatic Road Extraction from Historical Maps Using Deep Learning Techniques: A Regional Case Study of Turkey in a German World War II Map. *ISPRS International Journal of Geo-Information*, *10*(8), 492. https://doi.org/10.3390/ijgi10080492

Erdem, F., & Avdan, U. (2020). Comparison of Different U-Net Models for Building Extraction from High-Resolution Aerial Imagery. *International Journal of Environment and Geoinformatics*, *7*(3), 221–227. https://doi.org/10.30897/ijegeo.684951

Feurer, M., & Hutter, F. (2019). Hyperparameter optimization. In *Automated machine learning* (pp. 3–33). Springer, Cham.

Field, K. (2018). *Cartography.* Esri Press.

Glasbey, C. A. (1993). An Analysis of Histogram-Based Thresholding Algorithms. *CVGIP: Graphical Models and Image Processing*, *55*(6), 532–537. https://doi.org/10.1006/cgip.1993.1040

Glassner, A. (2021). *Deep Learning: A Visual Approach*. No Starch Press. https://books.google.de/books?id=NgTyDwAAQBAJ

Gobbi, S., Ciolli, M., La Porta, N., Rocchini, D., Tattoni, C., & Zatelli, P. (2019). New Tools for the Classification and Filtering of Historical Maps. *ISPRS International Journal of Geo-Information*, *8*(10), 455. https://doi.org/10.3390/ijgi8100455

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning*. MIT press.

Grafarend, E. W., & Krumm, F. W. (2014). *Map projections*. Springer.

Groom, G., Levin, G., Svenningsen, S., & Linnet Perner, M. (2020). Historical Maps – Machine Learning Helps Us over the Map Vectorisation Crux. *Automatic Vectorisation of Historical Maps: International Workshop Organized by the ICA Commission on Cartographic Heritage into the Digital. Budapest – 13 March, 2020*, 91–100. https://doi.org/10.21862/avhm2020.11

Haase, D., Walz, U., Neubert, M., & Rosenberg, M. (2007). Changes to Central European landscapes—Analysing historical maps to approach current environmental issues, examples from Saxony, Central Germany. *Land Use Policy*, *24*(1), 248–263. https://doi.org/10.1016/j.landusepol.2005.09.003

He, K., Zhang, X., Ren, S., & Sun, J. (2016). *Deep Residual Learning for Image Recognition*. 770–778. https://openaccess.thecvf.com/content_cvpr_2016/html/He_Deep_Residual_Learning_CVPR_2016_paper.html

He, Y., Hu, T., & Zeng, D. (2019, June). Scan-Flood Fill(SCAFF): An Efficient Automatic Precise Region Filling Algorithm for Complicated Regions. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*.

Heitzler, M., & Hurni, L. (2020). Cartographic reconstruction of building footprints from historical maps: A study on the Swiss Siegfried map. *Transactions in GIS*, *24*(2), 442–461. https://doi.org/10.1111/tgis.12610

Ignjatić, J., Nikolić, B., & Rikalović, A. (2018). *Deep learning for historical cadastral maps digitization: Overview, challenges and potential*. Václav Skala - UNION Agency. https://doi.org/10.24132/CSRN.2018.2803.6

Introduction to Convolutional Neural Networks Architecture. (n.d.). ProjectPro. Retrieved September 8, 2022, from https://www.projectpro.io/article/introduction-to-convolutional-neural-networks-algorithm-architecture/560

Jiao, C., Heitzler, M., & Hurni, L. (2020a). Extracting Wetlands from Swiss Historical Maps with ConvolutionalNeural Networks. *Automatic Vectorisation of Historical Maps. International Workshop Organized by the ICA Commission on Cartographic Heritage into the Digital 13 March, 2020 Budapest. Proceedings*, 33–38. https://doi.org/10.21862/avhm2020.03

Jiao, C., Heitzler, M., & Hurni, L. (2020b). *Extracting Wetlands from Swiss Historical Maps with ConvolutionalNeural Networks* (p. 38). https://doi.org/10.21862/avhm2020.03

Kang, Y. (2020). *Integrating Artificial Intelligence in Cartography: Using Deep Learning for Map Style Transfer and Map Generalization* [Thesis]. https://minds.wisconsin.edu/handle/1793/81097

Khattab, D., Ebied, H. M., Hussein, A. S., & Tolba, M. F. (2014). Color Image Segmentation Based on Different Color Space Models Using Automatic GrabCut. *The Scientific World Journal*, *2014*, 1–10. https://doi.org/10.1155/2014/126025

Khotanzad, A., & Zink, E. (1996). Color paper map segmentation using eigenvector line-fitting. *Proceeding of Southwest Symposium on Image Analysis and Interpretation*, 190–194. https://doi.org/10.1109/IAI.1996.493751

Kim, J. S., & Hong, I. Y. (2021). Analysis of Building Object Detection Based on the YOLO Neural Network Using UAV Images. *Journal of the Korean Society of Surveying, Geodesy, Photogrammetry and Cartography*, *39*(6), 381–392. https://doi.org/10.7848/ksgpc.2021.39.6.381

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2017). ImageNet classification with deep convolutional neural networks. *Communications of the ACM*, *60*(6), 84–90. https://doi.org/10.1145/3065386

Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., & Zitnick, C. L. (2014). Microsoft COCO: Common Objects in Context. In D. Fleet, T. Pajdla, B. Schiele, & T. Tuytelaars (Eds.), *Computer Vision – ECCV 2014* (pp. 740–755). Springer International Publishing. https://doi.org/10.1007/978-3-319-10602-1_48

Mendt, J. (2014). Virtuelles Kartenforum 2.0. *BIS - Das Magazin der Bibliotheken in Sachsen - Jg. 7. 2014, H. 3*.

Mitchell, T. M. (1997). *Machine Learning*. McGraw-Hill.

Nwankpa, C., Ijomah, W., Gachagan, A., & Marshall, S. (2018). *Activation Functions: Comparison of trends in Practice and Research for Deep Learning* (arXiv:1811.03378). arXiv. https://doi.org/10.48550/arXiv.1811.03378

O'Shea, K., & Nash, R. (2015). *An Introduction to Convolutional Neural Networks* (arXiv:1511.08458). arXiv. https://doi.org/10.48550/arXiv.1511.08458

Oskoei, M. A., & Hu, H. (2010). *A Survey on Edge Detection Methods*.

Pan, S. J., & Yang, Q. (2010). A Survey on Transfer Learning. *IEEE Transactions on Knowledge and Data Engineering*, *22*(10), 1345–1359. https://doi.org/10.1109/TKDE.2009.191

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., … Chintala, S. (2019). PyTorch: An Imperative Style, High-Performance Deep Learning Library. *Advances in Neural Information Processing Systems*, *32*. https://proceedings.neurips.cc/paper/2019/hash/bdbca288fee7f92f2bfa9f7012727740-Abstract.html

Petitpierre, R., Kaplan, F., & di Lenardo, I. (2021). *Generic Semantic Segmentation of Historical Maps*.

Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). *You Only Look Once: Unified, Real-Time Object Detection* (arXiv:1506.02640). arXiv. https://doi.org/10.48550/arXiv.1506.02640

Ronneberger, O., Fischer, P., & Brox, T. (2015). *U-Net: Convolutional Networks for Biomedical Image Segmentation* (arXiv:1505.04597). arXiv. https://doi.org/10.48550/arXiv.1505.04597

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature*, *323*(6088), 533–536. https://doi.org/10.1038/323533a0

Sejnowski, T. J. (2020). The unreasonable effectiveness of deep learning in artificial intelligence. *Proceedings of the National Academy of Sciences*, *117*(48), 30033–30038. https://doi.org/10.1073/pnas.1907373117

Sharma, V. V. (2020, October 18). *Weekly Supervised Learning—Getting started with unstructured data*. Medium. https://towardsdatascience.com/weekly-supervised-learning-getting-started-with-unstructured-data-123354dad7c1

Sheng, V., & Zhang, J. (2019). Machine Learning with Crowdsourcing: A Brief Summary of the Past Research and Future Directions. *Proceedings of the AAAI Conference on Artificial Intelligence*, *33*, 9837–9843. https://doi.org/10.1609/aaai.v33i01.33019837

Shorten, C., & Khoshgoftaar, T. M. (2019). A survey on Image Data Augmentation for Deep Learning. *Journal of Big Data*, *6*(1), 60. https://doi.org/10.1186/s40537-019-0197-0

Szegedy, C., Ioffe, S., Vanhoucke, V., & Alemi, A. (2016). *Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning* (arXiv:1602.07261; Version 2). arXiv. https://doi.org/10.48550/arXiv.1602.07261

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., & Rabinovich, A. (2014). *Going Deeper with Convolutions* (arXiv:1409.4842). arXiv. https://doi.org/10.48550/arXiv.1409.4842

Taylor, L., & Nitschke, G. (2018). Improving Deep Learning with Generic Data Augmentation. *2018 IEEE Symposium Series on Computational Intelligence (SSCI)*, 1542–1547. https://doi.org/10.1109/SSCI.2018.8628742

Tiu, E. (2020, October 3). *Metrics to Evaluate your Semantic Segmentation Model*. Medium. https://towardsdatascience.com/metrics-to-evaluate-your-semantic-segmentation-model-6bcb99639aa2

Uhl, J. H., Leyk, S., Chiang, Y.-Y., Duan, W., & Knoblock, C. A. (2020). Automated Extraction of Human Settlement Patterns From Historical Topographic Map Series Using Weakly Supervised Convolutional Neural Networks. *IEEE Access*, *8*, 6978–6996. https://doi.org/10.1109/ACCESS.2019.2963213

Walz, U. (2002). *Historische Kartenwerke in Sachsen als Grundlage für Untersuchungen zur Landschaftsentwicklung* (pp. 113–118).

Witschas, S. (2002). Erinnerung an die Zukunft—Sächsische historische Kartenwerke zeigen den Landschaftswandel. *KN - Journal of Cartography and Geographic Information*, *52*(3), 111–117. https://doi.org/10.1007/BF03544891

Xia, X., Heitzler, M., & Hurni, L. (2022). CNN-BASED TEMPLATE MATCHING FOR DETECTING FEATURES FROM HISTORICAL MAPS. *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, *XLIII-B2-2022*, 1167–1173. https://doi.org/10.5194/isprs-archives-XLIII-B2-2022-1167-2022

Xu, K., Nie, L., Zhang, Z., Yu, W., Zhang, Y., Chen, W., Li, S., Zhou, X., Deng, S., Yu, P., Fan, Y., Zhang, H., & Bouillon, V. (2019). IRSNET: An Inception-Resnet Feature Reconstruction Model for Building

Segmentation. In T. Gedeon, K. W. Wong, & M. Lee (Eds.), *Neural Information Processing* (pp. 41–48). Springer International Publishing. https://doi.org/10.1007/978-3-030-36802-9_5

Yohanandan, S. (2020, June 9). *MAP (mean Average Precision) might confuse you!* Medium. https://towardsdatascience.com/map-mean-average-precision-might-confuse-you-5956f1bfa9e2